Programmation Visual Basic Application sur Excel

Samir DELIMI

Samir.delimi@umontpellier.fr

"Tout le monde est un génie. Mais si on juge un poisson sur sa capacité à grimper à un arbre, il passera sa vie à croire qu'il est stupide..."

A l'issue de ce chapitre,

• vous serez capable :

- ❑ d'interpréter un programme répondant à un problème de gestion ;
- ❑ de rédiger ou compléter le code d'une fonction ou d'une procédure ;
- ❑ de corriger ou modifier un programme afin de l'adapter à un nouveau problème de gestion ;
- ❑ d'utiliser et enregistrer une macro-commande.

• vous saurez identifier :

- ❑ la programmation au sein d'un tableur ;
- ❑ le modèle d'objets associé à un tableur ;
- ❑ les familles d'instruction ;
- ❑ l'affectation d'objets, de variables et de paramètres ;
- ❑ les instructions d'entrée, de calcul, de cumul et de sortie ;
- ❑ les tests (structures alternatives) simples et imbriqués ;
- ❑ les boucles (structures itératives) ;
- ❑ les fonctions ;
- ❑ les procédures.

A l'issue de ce chapitre,

• vous serez capable :

- ❑ d'interpréter un programme répondant à un problème de gestion ;
- ❑ de rédiger ou compléter le code d'une fonction ou d'une procédure ;
- ❑ de corriger ou modifier un programme afin de l'adapter à un nouveau problème de gestion ;
- ❑ d'utiliser et enregistrer une macro-commande.

• vous saurez identifier :

- ❑ la programmation au sein d'un tableur ;
- ❑ le modèle d'objets associé à un tableur ;
- ❑ les familles d'instruction ;
- ❑ l'affectation d'objets, de variables et de paramètres ;
- ❑ les instructions d'entrée, de calcul, de cumul et de sortie ;
- ❑ les tests (structures alternatives) simples et imbriqués ;
- ❑ les boucles (structures itératives) ;
- ❑ les fonctions ;
- ❑ les procédures.

© DCG Vuibert

La programmation au service du tableur

La programmation au service du tableur

© DCG Vuibert

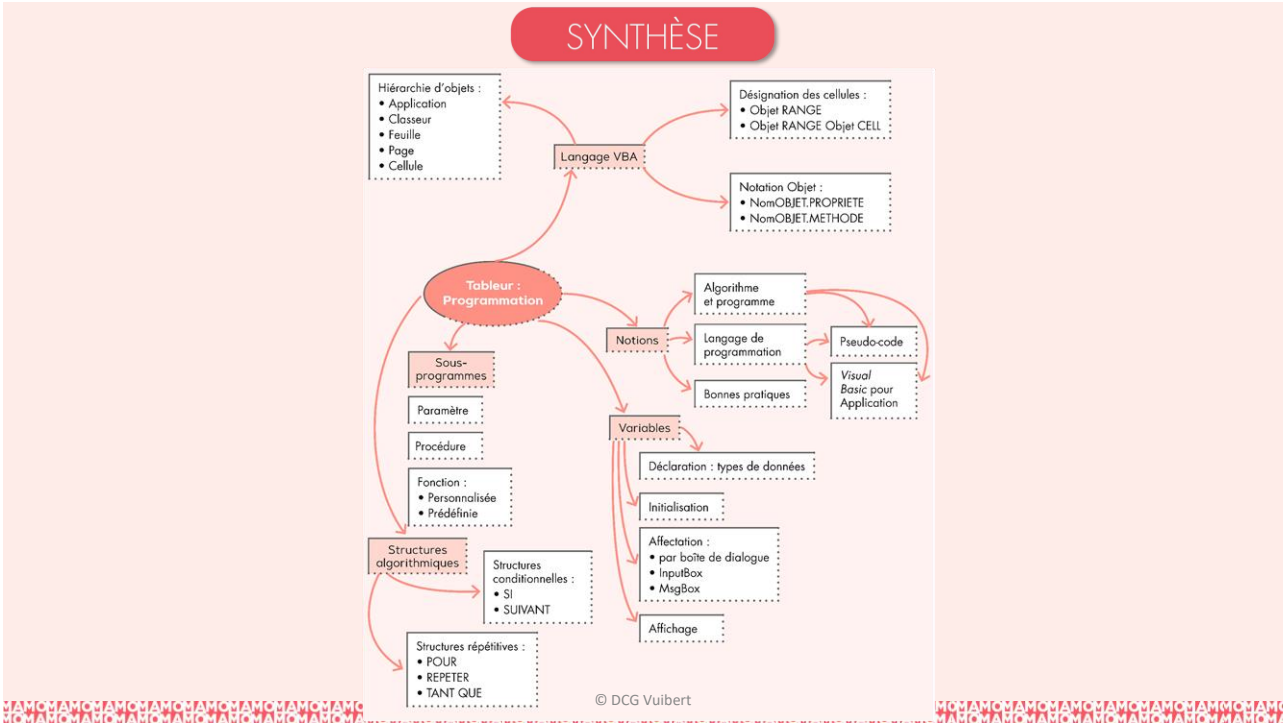
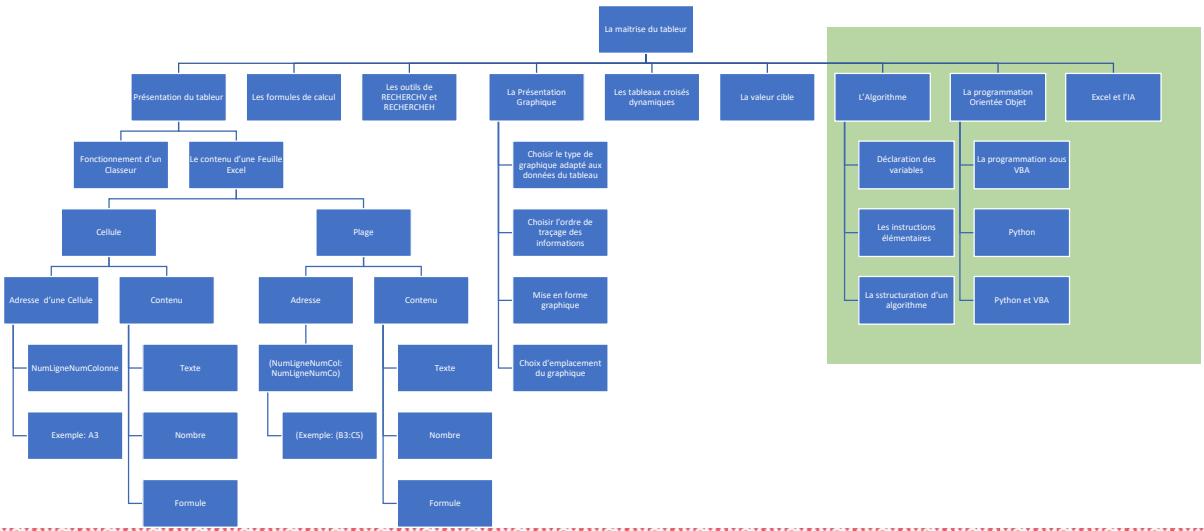
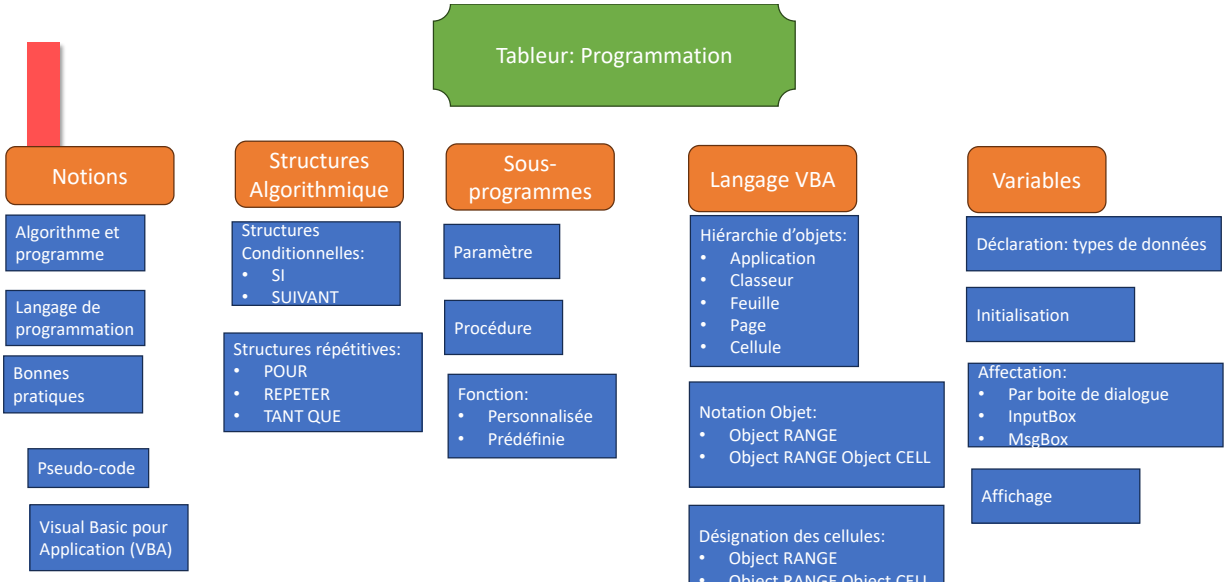


Schéma de synthèse du tableau Excel





Le pseudocode est une méthode permettant de décrire l'algorithme d'un programme à l'aide d'un langage naturel, sans se soucier des détails syntaxiques propres à un langage de programmation spécifique.
VBA: Langage de programmation utilisé dans Microsoft Excel pour automatiser des tâches, créer des fonctions personnalisées et interagir avec les données dans les feuilles de calcul.

pour intégrer le pseudocode dans VBA pour Excel, utilisez des commentaires explicatifs, des noms de variables significatifs et découpez éventuellement votre algorithme en fonctions ou sous-procédures pour une meilleure lisibilité et compréhension du code.

Objectifs du Cours

- À l'issue de ce cours, vous serez capable :
  - d'interpréter un programme répondant à un problème de gestion
  - de rédiger ou compléter le code d'une fonction ou d'une procédure;
  - de corriger ou modifier un programme afin de l'adapter à un nouveau problème de gestion;
  - d'utiliser et enregistrer une macro-commande.
- vous saurez identifier :
  - la programmation au sein d'un tableur;
  - le modèle d'objets associé à un tableur ;
  - les familles d'instruction;
  - l'affectation d'objets, de variables et de paramètres;
  - les instructions d'entrée, de calcul, de cumul et de sortie
  - les tests (structures alternatives) simples et imbriqués ;
  - les boucles (structures itératives);
  - les fonctions ;
  - les procédures

Mettez-vous en situation

- Mettez-vous en situation
  - Vous venez d'intégrer le cabinet ComptaPro, en tant que jeune diplômé(e) en Gestion.
  - Plusieurs missions de démarrage vous sont confiées et notamment celle d'aider certains clients sur tableur.
  - En effet, se nombreux clients sollicitent le cabinet pour automatiser leurs feuilles de calcul sous le tableur Excel.
  - Cette automatisation permettrait d'effectuer des traitements sur tableur impossibles à mettre en œuvre avec de simples formules.
  - Mais comment écrit-on un programme pouvant être exécuté sous un tableur? Quelles instructions permettent de contrôler les saisies effectuées par l'utilisateur, d'afficher un message ou de manipuler les cellules d'une feuille ?

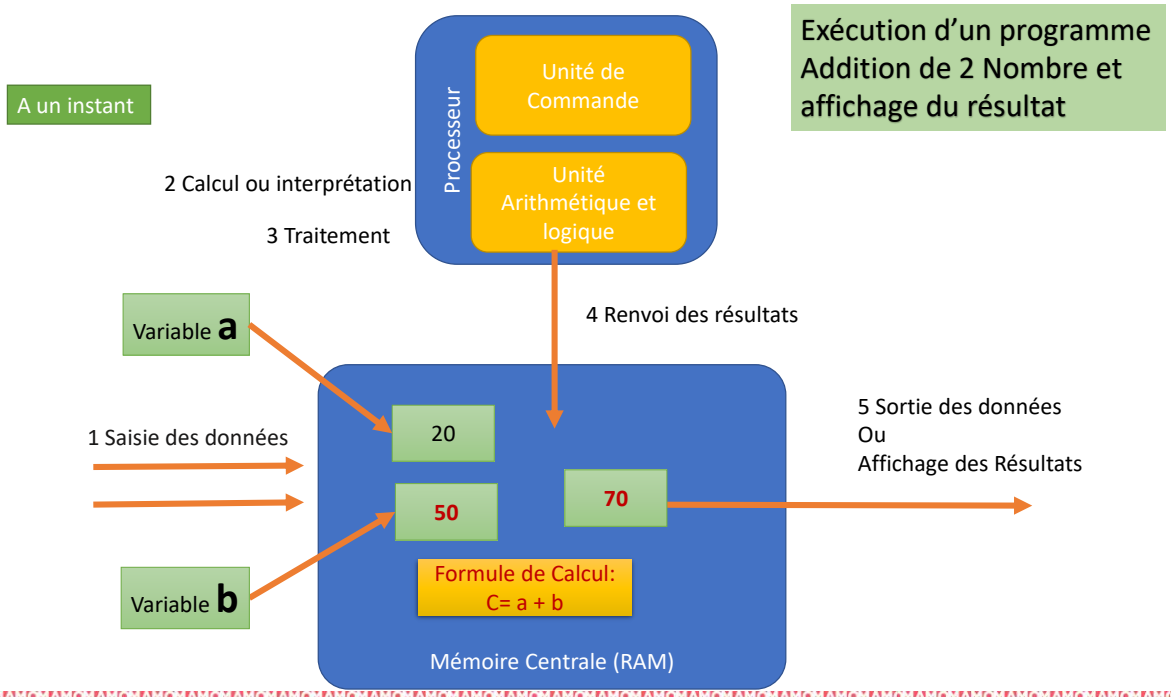
Cours 1

Avant de commencer
Rappel du fonctionnement de la mémoire et les variables

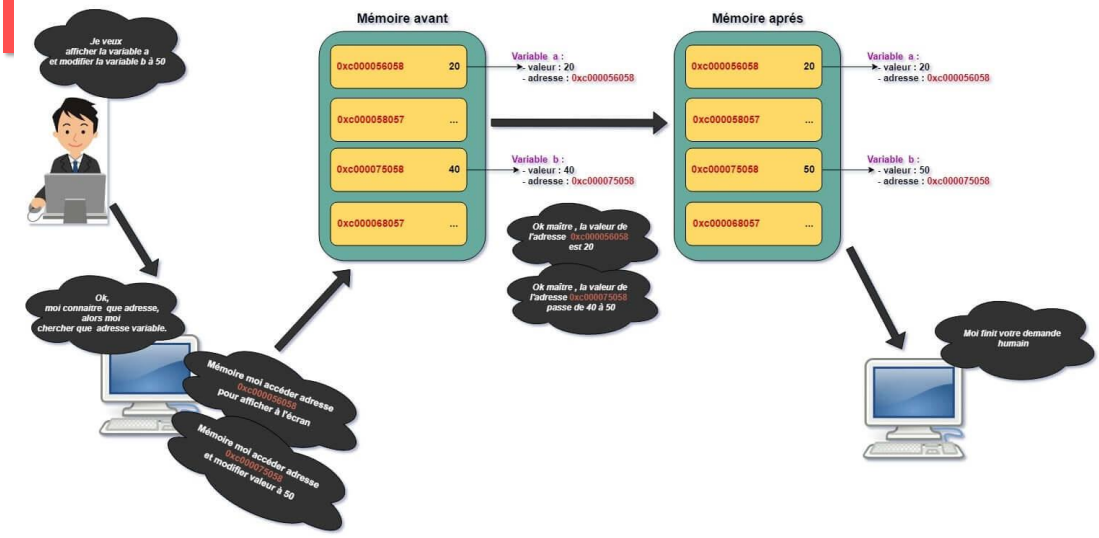
Indispensable pour comprendre les notions de base de la programmation

Fonctionnement de la mémoire et les variables

- Vous ne vous êtes jamais demandé comment votre machine manipule les valeurs de vos variables ?
  - **C'est grâce aux adresses de vos variables !**
- une adresse comme une adresse postale ?
  - Oui tout à fait, toute variable manipulée dans un programme est stockée quelque part dans la mémoire de votre machine.
  - Cette **mémoire** est constituée **d'octets** qui sont identifiés de manière univoque par un **numéro** qu'on appelle **adresse**
- Donc si votre machine veut manipuler vos variables, elle aura besoin au préalable de connaître où se situe votre variable grâce à son adresse mémoire qui lui indiquera la position de la donnée dans la mémoire vive (la RAM de votre ordinateur).
  - Par exemple si on prend le fonctionnement de la poste, le facteur a besoin de l'adresse de destination pour savoir chez qui il doit renvoyer la lettre.
  - Ici le facteur c'est votre ordinateur et l'adresse destination correspond à l'adresse de la variable que l'ordinateur souhaite chercher.



Fonctionnement de la mémoire et les



I. Notions liées à la programmation

1. Notions liées à la programmation:
A- La programmation au service du tableur

- La programmation au service du tableur consiste à écrire des programmes à l'intérieur des tableurs, de manière à automatiser des tâches (de contrôle, de calcul) et à permettre l'interaction avec l'utilisateur (demande de valeurs et/ou affichage de messages d'alerte ou d'information).

Cours 2

ALGORITHMIE - PROGRAMMATION

Généralités sur la programmation

Algorithmie vs. Programmation

Algorithmie et algorithmes

- Solution « informatique » relative à un Problème;
- Un algorithme est une suite finie et non ambiguë d'opérations ou instructions permettant de résoudre un problème ou d'obtenir un résultat.
- Suite d'actions (instructions) appliquées sur des données;
- 3 étapes principales :
  1. Saisie (réception) des données
  2. Traitements
  3. Restitution (application) des résultats

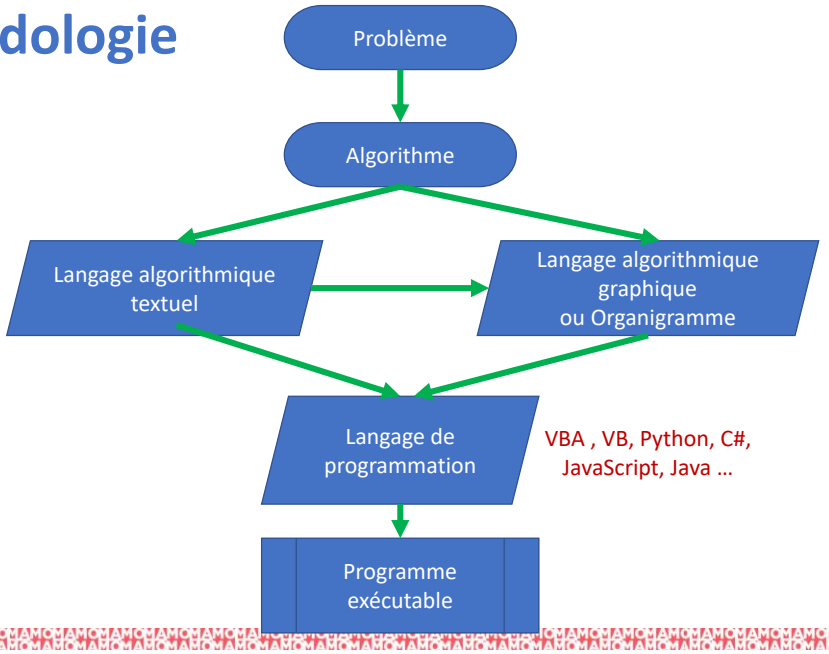
Programme

- Une **traduction** d'un algorithme en un langage interprétable par un ordinateur
- **Transcription** d'un algorithme avec une syntaxe prédéfinie;
- Visual Basic pour Applications;
- Même principes fondamentaux que les autres langages objets (Java, C#, etc.);
- VBA agit en interaction avec les fonctions prédéfinies disponibles dans la suite Office.

Etapes de la conception d'un programme (une application)

1. Déterminer les besoins et fixer les objectifs : que doit faire le logiciel, dans quel cadre va- t-il servir, quels seront les utilisateurs types ? On rédige un cahier des charges avec le commanditaire du logiciel (Remarque : commanditaire = maître d'ouvrage ; réalisateur = maître d'œuvre)
2. Conception et spécifications : quels sont les fonctionnalités du logiciel, avec quelle interface ?
3. Programmation : modélisation et codage
4. Tests : obtient-on les résultats attendus, les calculs sont corrects, y a-t-il plantage et dans quelles circonstances ? (tests unitaires, tests d'intégration, etc.)
5. Déploiement : installer-le chez le client (vérification des configurations, installation de l'exécutable et des fichiers annexes, etc.)
6. Maintenance : corrective, traquer les bugs et les corriger (patches) ; évolutive (ajouter des fonctionnalités nouvelles au logiciel : soit sur l'ergonomie, soit en ajoutant de nouvelles procédures)

Méthodologie



1. Notions liées à la programmation:
A- Algorithme

- **Définitions:** Un algorithme est solution « informatique » relative à un Problème;
- Un algorithme est une suite finie et non ambiguë d'opérations ou instructions permettant de résoudre un problème ou d'obtenir un résultat. (Wikipédia).
- Suite d'actions (instructions) appliquées sur des données;
- Un algorithme peut prendre des données en entrée et fournit au moins un résultat en sortie
- Un algorithme est souvent exprimé en notation indépendante de tout langage de programmation.
- Un organigramme est une représentation graphique d'un algorithme
- 3 étapes principales :
  - Saisie (réception) des données en entrée
  - Traitements
  - Restitution (application) des résultats en sortie

1. Notions liées à la programmation:
A- Exemples d'algorithme

Exemples	Données en entrées	Traitement	Résultats
Algorithme permettant de calculer le carré d'un nombre	Nombre	Nombre x Nombre	Carré
Algorithme permettant de calculer un montant TTC	Montant HT	Montant HT + Montant TVA	Montant TTC

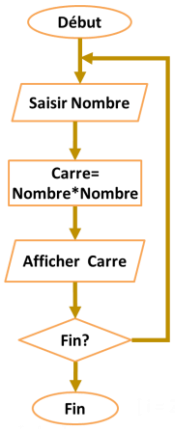
Exemple d'algorithme de calcul d'un carré

Algorithme

Début
- Lire Nombre
- Carré= Nombre * Nombre
- Afficher Carré
- Lire un autre nombre?

Fin

Algorithme sous forme d'organigramme



1. Notions liées à la programmation:
B- Programme

- **Définition** Un programme est une suite d'instructions écrites dans un langage de programmation et exécutées par un ordinateur.
- Cette suite d'instructions manipule des données « **en entrées**» et renvoie des résultats « **en sortie**».
- Une **traduction** d'un algorithme en un langage interprétable par un ordinateur
- **Transcription** d'un algorithme avec une syntaxe prédéfinie;

Exemple d'algorithme et programme de calcul du montant d'une ristourne

Calcul du montant de la ristourne (corps de l'algorithme et du programme)	
Algorithme	**VBA**

LIRE CA
SI CA >10000
ALORS RIST ← CA*0,06
SINON RIST ← CA*0,03
FIN SI
ECRIRE (RIST)
	CA = InputBox("Donner le montant du CA")
	If CA > 10000 Then
	RIST = CA*0,06
	Else RIST = CA*0,03
	End If
	MsgBox ("le montant de la ristourne est de "& RIST &" €")

1. Notions liées à la programmation:
B- Langage de programmation

- Sous le tableur Microsoft Excel, les programmes seront écrits en **langage Visual-Basic for Applications (VBA).**
- Sous Excel, les programmes écrits en VBA sont appelés **macros** ou macro-commandes.
- Les **instructions** peuvent être:
  - **enregistrées automatiquement** par la fonctionnalité: Enregistrer Macro;
  - **ou** être **écrites manuellement** dans un éditeur de code.
- **Chaque feuille** d'un classeur peut contenir **ses propres programmes**.
- Dans le cas d'un programme s'exécutant sur toutes les feuilles d'un classeur, celui-ci sera stocké dans la partie **Module**.

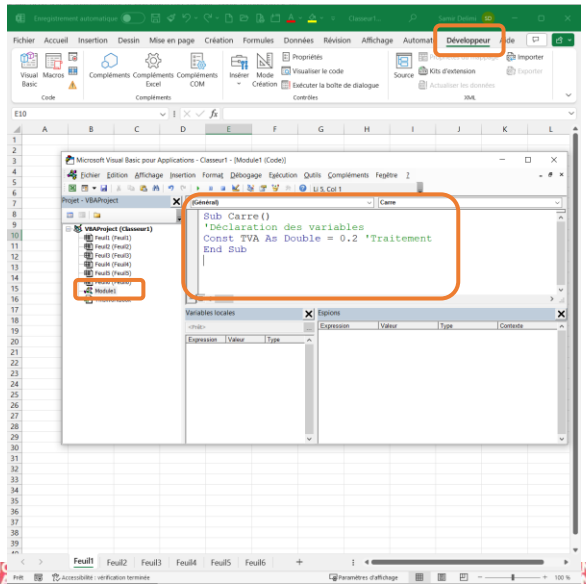
VBA : documents MS Office

- Un document MS Office est composé des plusieurs éléments…

MS Excel — **Fichier Excel** — **Classeur**
- Feuilles
- Données
- Formules
- **Modules (VBA)**

MS Access — **Fichier Access** — **Base de données**
- Tables (données)
- Requêtes
- Formulaires
- États (rapports)
- **Modules (VBA)**

Excel et l'interface de programmation VBA



Programme et langage de programmation

- Visual Basic pour Applications;
- Même principes fondamentaux que les autres langages objets (Java, C#, etc.);
- VBA agit en interaction avec les fonctions prédéfinies disponibles dans la suite Office.

Langage de programmation

- 1993: Date de première version de VBA
- 2013 : Lancement de VBA 7.1 couvrant la suite office 2013, 2016 et 2019
- https://fr.wikipedia.org/wiki/Visual_Basic_for_Applications
- https://fr.wikipedia.org/wiki/Basic_(langage)

Mode compilé vs. Mode interprété

- VBA (Visual Basic pour Applications) **est un langage de programmation** dédié principalement aux applications Microsoft Office. Il est basé sur le langage Visual Basic, mais ne peut s'exécuter que dans une application hôte Microsoft Office, et non de manière autonome.
- Pour info, VB et **VBA**, c'est EXACTEMENT le même **langage**. C'est le même moteur. La différence, c'est que l'on peut **compiler** du VB en EXE ou DLL alors que **VBA** est **interprété**.
- VBA langage compile ou interprète?
  - Langage interprété : + portabilité application ; - lenteur (R, **VBA**, Python…)
  - Langage compilé : + rapidité ; - pas portable
  - (solution possible : write once, compile anywhere ;)
  - Langage pseudo-compilé : + portabilité plate-forme ; - lenteur (?) (principe : write once, run anywhere ; ex. Java et le principe JIT)

Pourqoui VBA avec Excel

- Excel est un logiciel très utilisé, mais souvent très "sous-utilisé". Ses possibilités dépassent largement la création des tableaux et des graphiques. Pour la plupart des utilisateurs, les fonctionnalités plus avancées (mais pas nécessairement plus compliquées) restent cependant méconnues. Tout comme les astuces qui rendent le travail avec Excel plus rapide et plus efficient.
- Le VBA – le langage de programmation de la suite Microsoft Office – est un autre moyen de repousser les limites de l'utilisation d'Excel. De la simple automatisation des tâches répétitives à la création des applications entières en passant par la création de vos propres fonctions, même une connaissance basique de VBA peut grandement améliorer votre manière de travailler.
- Ce Cours a pour but de vous fournir les moyens simples et efficaces pour maîtriser Excel. Que vous soyez débutants ou utilisateurs confirmés, vous trouverez dans ce cours de quoi vous faciliter la vie.

VBA : quoi & pourquoi ?

- VBA : quoi ?
  - **Langage** et **environnement** de programmation Orienté Objets
  - **Attaché aux documents** MS Office
- VBA : pourquoi ?
  - Associer **un comportement actif** à des documents Office
    - Calculs, vérifications, etc.
- Visual Basic pour Applications, VBA, est la solution de programmation proposée avec les applications de la suite Office.
- La connaissance de VBA permet à l'utilisateur d'Excel de tirer pleinement profit du tableur de Microsoft en développant les capacités et les fonctionnalités pour ses besoins spécifiques.
- Maîtriser Visual Basic pour Applications, c'est à coup sûr améliorer grandement sa productivité et celle de son entreprise.

Quand et pourquoi utiliser VBA dans Office

- Il existe trois principales raisons d'utiliser la programmation VBA dans Office:
  - **Automatisation et répétition**
  - **Extensions de l'interaction utilisateur**
  - **Interaction entre les applications Office**

Quand et pourquoi utiliser VBA

- Il existe trois principales raisons d'utiliser la programmation VBA dans Office
  1. **Automatisation et répétition**
     - VBA est efficace en ce qui concerne les solutions répétitives.
       - Si vous avez une modification que vous devez effectuer plus de dix ou vingt fois, cela peut valoir le coup de mettre en place une automatisation avec VBA.
       - S'il s'agit d'une modification que vous avez à faire des centaines de fois, cela vaut définitivement le coup.
       - N'importe quel changement de mise en forme ou modification que vous pouvez effectuer à la main peut être effectué dans VBA.
  2. Extensions de l'interaction utilisateur
  3. Interaction entre les applications Office

Quand et pourquoi utiliser VBA

- Il existe trois principales raisons d'utiliser la programmation VBA dans Office:
  1. Automatisation et répétition
  2. **Extensions de l'interaction utilisateur**

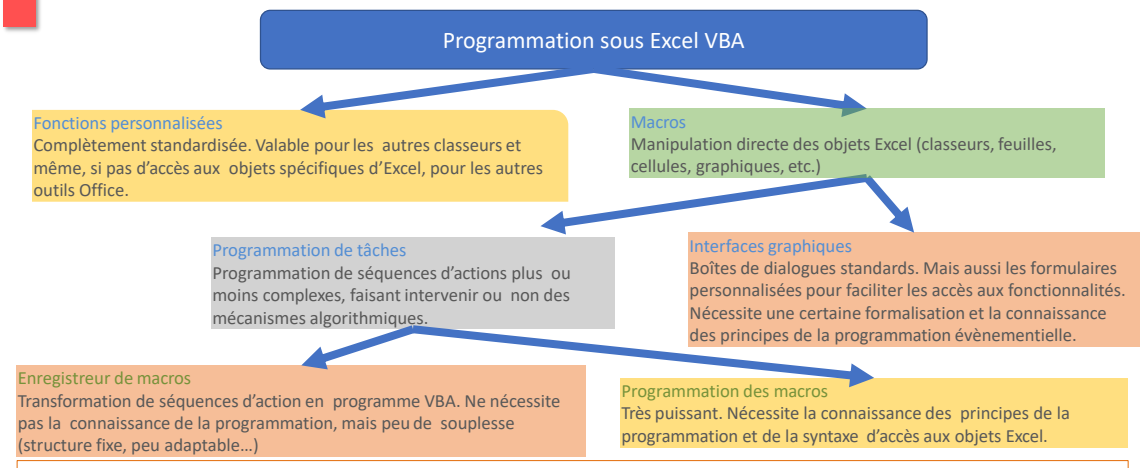
     Il arrive parfois que l'on souhaite encourager ou forcer les utilisateurs à interagir avec l'application ou le document Office d'une manière particulière qui ne fait pas partie de l'application standard.

     Par exemple, vous pourriez vouloir inviter les utilisateurs à effectuer une action spécifique lorsqu'ils ouvrent, enregistrent ou impriment un document.
  3. Interaction entre les applications Office

Quand et pourquoi utiliser VBA

- Il existe trois principales raisons d'utiliser la programmation VBA dans Office
  1. Automatisation et répétition
  2. Extensions de l'interaction utilisateur
  3. **Interaction entre les applications Office**
     - Vous voulez copier tous vos contacts à partir d'Outlook dans Word. Ou vous devez déplacer des données à partir d'Excel vers Access, parfois un simple copier-coller ne fait pas ce que vous souhaitez faire, ou est trop lent.
     - Vous pouvez utiliser la programmation VBA pour interagir avec les détails de deux ou plusieurs applications Office en même temps, puis modifier le contenu dans une application en fonction du contenu dans une autre.

Généralités sur la programmation VBA sous Excel



Points importants. Connaissance de l'algorithmie, langage de programmation Visual Basic. Les instructions sont écrites dans des fonctions (*function*) et procédures (*sub*), qui sont regroupées dans des modules.
Nous travaillons dans VBE (Visual Basic Editor).

Exemple de Macros VBA

```
Sub SupprimerColonne()
'
' SupprimerColonne Macro
' Macro de Suppresssion d'une plage
'
    Range("A1:A14").Select

    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .Color = 45265
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With

End Sub
```

VBA et Excel : Ce que ça peut vous apporter

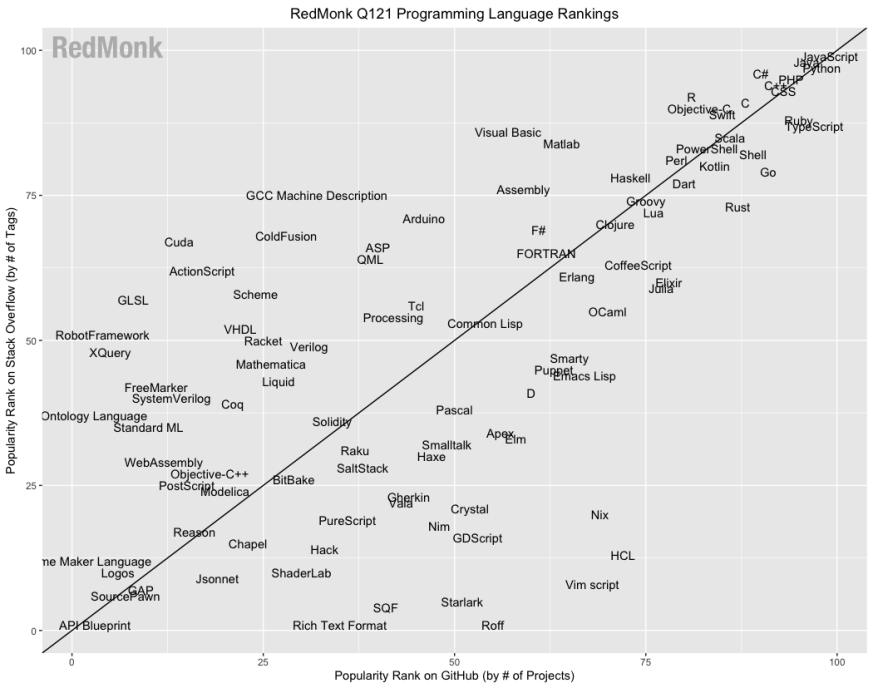
- Créer ses propres fonctions Excel !
- Automatiser des tâches répétitives et gagner énormément de temps
- Ajouter des nouvelles fonctionnalités à Excel
  - Extraire les doublons, etc…
- Modifier l'interface (rajout de menus, boutons, formulaire)
  - Exemple de formulaire des gestions de stocks, factures ou commandes.
- Gérer les données
  - Transfert vers ou depuis Word/PowerPoint, bases de données, web, ERP, CRM …
- Créer des applications métiers
  - Application de Trading en finance
  - Etc…

Pourquoi de nombreuses entreprises continuent-elles à tolérer les macros Excel et le VBA ?

- le but d'une entreprise c'est de gagner de l'argent et que ceux qui font gagner de l'argent sont côté métier et y a beaucoup de métier qui tiennent à VBA
- Tout le monde n'est pas programmeur émérite capable de manipuler un langage du type C++, Java et consorts…
- Le Basic possède une certaine simplicité (d'où son nom) et Excel permet de faire bien des choses. De ce fait, comme dans les entreprises, on est tjrs pressé par le temps qui manque, Excel+VBA est une façon plutôt efficace de résoudre vite et pour pas cher un pb.
- Parce que c'est facile, ça marche, c'est agile, c'est pas cher, c'est transparent, et formateur.
- Et il n'y a pas de mauvais langage mais de mauvais programmeurs.

VBA et votre carrière

- Un énorme plus
- Souvent indispensable en Comptabilité et Finance
- Un précieux allié pour gérer les données Excel
  - Marketing
  - Management
  - Commercial
  - Etc…
- Pas besoin de devenir une As du VBA, même un petit niveau peut apporter beaucoup
- Un investissement que vous ne regretterez pas



LES MACROS (1)

Programmation des macros – Travailler directement sur les feuilles:

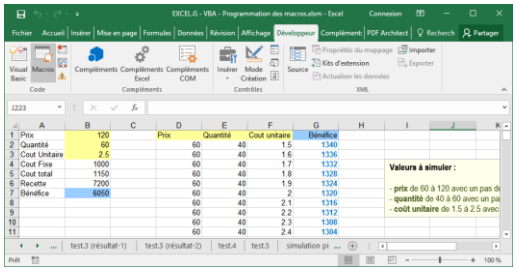
Enregistreur de macros

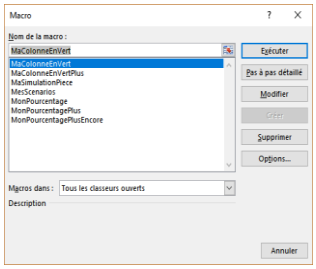
Macros ?

- **Une macro est une séquence d'actions automatisant une tâche dans le langage de programmation Visual Basic pour Applications (VBA).**
- L'enregistreur de macros
  - Pour automatiser une tâche répétitive, mettre en place des mécanismes de contrôle et actualiser des données, il est possible d'enregistrer une macro-commande ou macro.
- Le langage VBA, la programmation orientée Objet
  - **Un programme** est un ensemble d'opérations écrites *par un informaticien* dans un langage sous forme de code
  - **Le VBA** (Visual Basic for Applications) est le langage développé pour les progiciels Microsoft. Il ne peut s'exécuter sans une application hôte (ex.: Excel, Access).
- Une **fonction** est un programme qui exécute une suite d'instructions et renvoie une valeur.
- Une **procédure** (Sub) est également un programme (suite d'instructions), exécuté par un utilisateur ou appelé dans une autre procédure mais ne renvoyant pas de valeur.

Macros ?

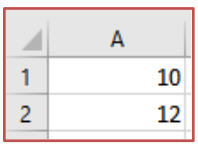
- Les macros sont également des procédures que l'on crée à l'intérieur d'un module.
- A la différence des fonction (Functions), ce sont des procédures (subroutines) Sub() sans paramètres qui peuvent **manipuler** (accéder et modifier) directement **les objets Excel (classeurs, feuilles, cellules, graphiques, scénarios, tableaux croisés dynamiques…**).
- Elles ne s'exécutent pas de la même manière.
- Au lieu de les insérer dans une cellule, elles se lancent globalement via le bouton **MACROS** dans le ruban **DEVELOPPEUR**.



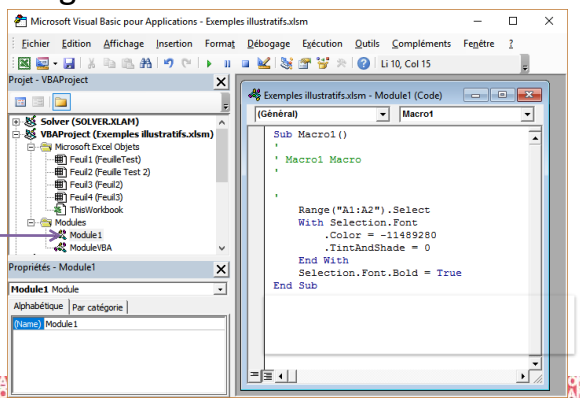


Enregistreur de macros

- Une manière simple de générer une macro est de lancer **l'enregistreur de macros**. Du code VBA est automatiquement généré.
- Exemple : mettre en gras et vert le contenu des cellules A1 et A2



Un nouveau module « Module1 » est automatiquement créé.



Une macro correspond à un Sub() sans paramètres.

Ecrire du code VBA pour les macros se révélera plus simple au final!

Enregistreur de macros - Bilan

- **Avantages :**
  - Il n'y a pas plus simple pour produire du code, on peut créer et exécuter une macro sans aucune notion de programmation
  - Il nous donne des indications précieuses sur les commandes associées aux objets Excel
- **Inconvénients :**
  - On travaille à structure fixée, si la configuration de la feuille change, il n'est pas possible de lancer la macro
  - On ne bénéficie pas de la puissance des structures algorithmiques
  - Impossibilité de créer des fonctions personnalisées.
- **En définitive :**
  - Il peut nous aider à rédiger notre code en nous donnant des pistes sur la syntaxe des commandes et les objets adéquats à manipuler (ex. imprimer automatiquement des feuilles, on lance l'enregistreur une fois, on intègre son code dans le notre à l'intérieur d'une boucle).

1. Notions liées à la programmation: Point sur la Sécurité

- Comme les macros sont des programmes qui peuvent agir dans et en dehors d'Excel (sur les fichiers), les macro-commandes sont par défaut désactivées dans un tableur.
- Pour pouvoir programmer, il faut donc réaliser deux tâches au préalable : activer la barre de développement et activer les macros,

1. Notions liées à la programmation:
C- Bonnes pratiques

- Pour faciliter la maintenance du programme créé. on veille à :
  - **Séparer des instructions** : une instruction par ligne_
  - **Mettre des commentaires** : lignes (non exécutables) d'information repérées par une côte devant le commentaire et apparaissant en vert.

```
Function calculCle(Code As String) As Integer

'calcule la clé correspond an code (n° client+ n° dep + n° remise) fourni en parimètres
```

  - **indenter ses instructions**: décaler ses instructions selon les structures algorithmiques utilisées et imbriquées.

```
Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Instruction 6
```

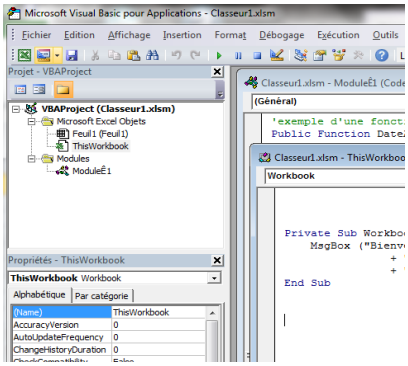
```
Instruction 1
        Instruction 2
        Instruction 3
Instruction 4
        Instruction 5
Instruction 6
```

L'ÉDITEUR VBE et premiers pas en VBA

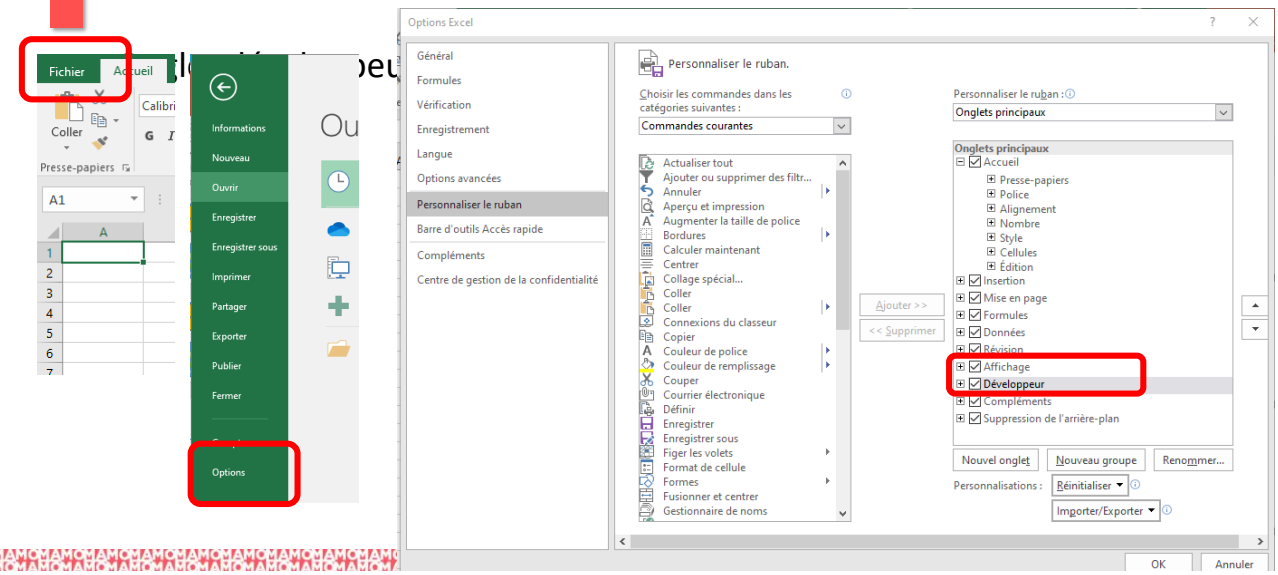
- Pour vous initier à l'utilisation du langage VBA, vous pouvez tout à fait commencer à construire vous-même vos premières macros Excel, en effectuant des ajustements au fur et à mesure.
- Pour ce faire, vous pouvez dès à présent utiliser l'enregistreur de macros : dans le ruban d'Excel, sélectionnez l'onglet « Développeur » et cliquez sur « Enregistrer une macro ».
- Si l'onglet « Développeur » n'est pas visible sur votre ruban Excel, pas de panique ! Il vous suffit de vous rendre dans Fichier > Options > Personnaliser le ruban, puis de cocher l'onglet « Développeur » dans la liste de droite.
- La stratégie d'essais-erreurs est tout à fait adaptée à l'apprentissage du langage VBA, qui est simple et intuitif. De nombreux professionnels de la finance se forment ainsi "sur le tas", par l'expérimentation directe.

VBA : concepts de base

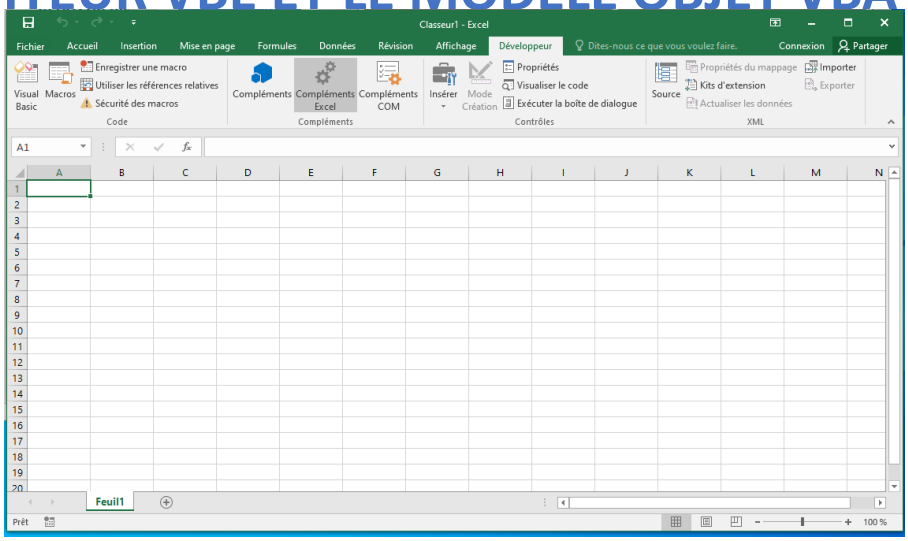
- Environnement de programmation (VBE)
- Excel : Ruban « Développeur » (onglet à activer)
- Access : Ruban « Outils de bases de données »

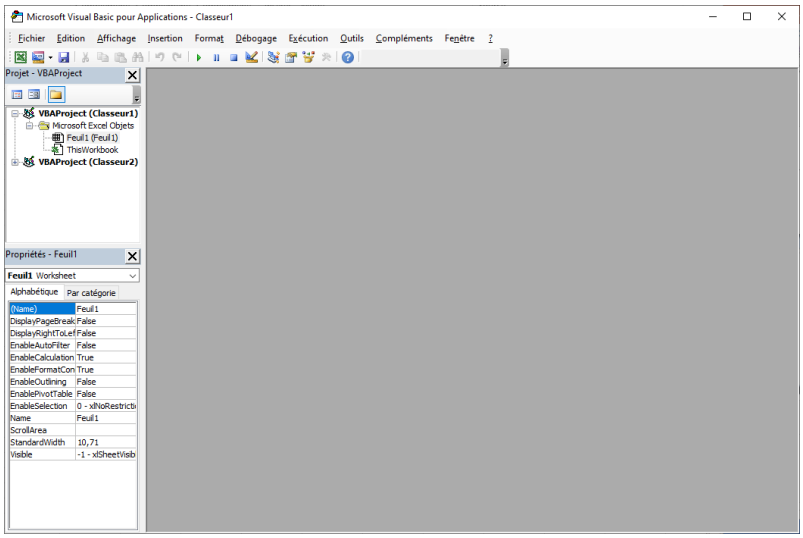


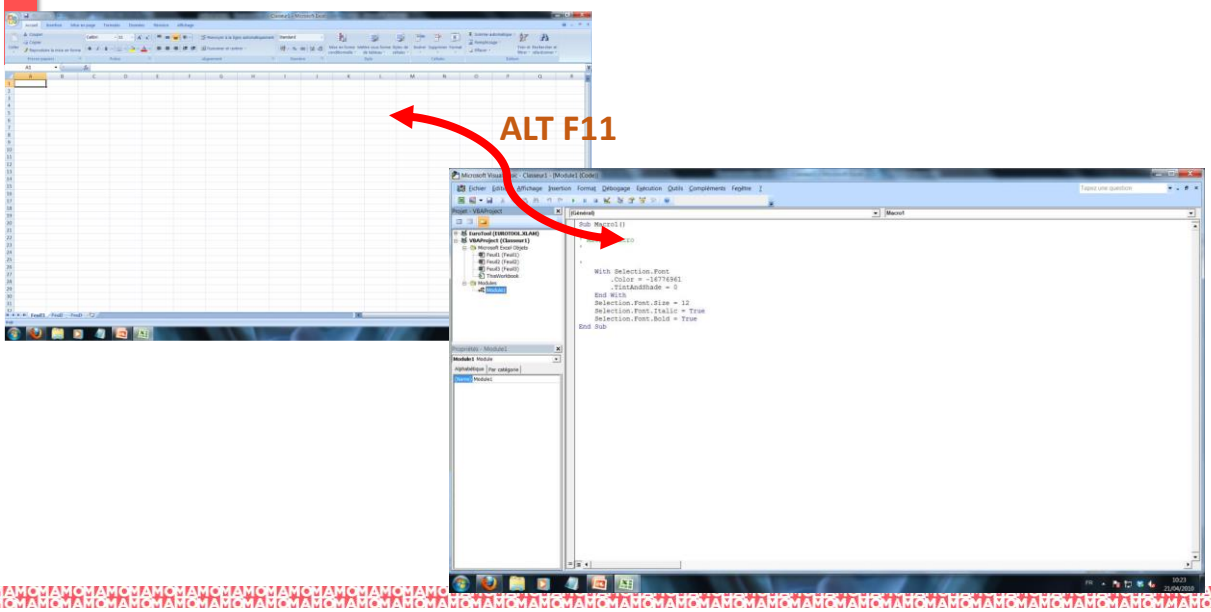
L'ÉDITEUR VBE ET LE MODÈLE OBJET VBA



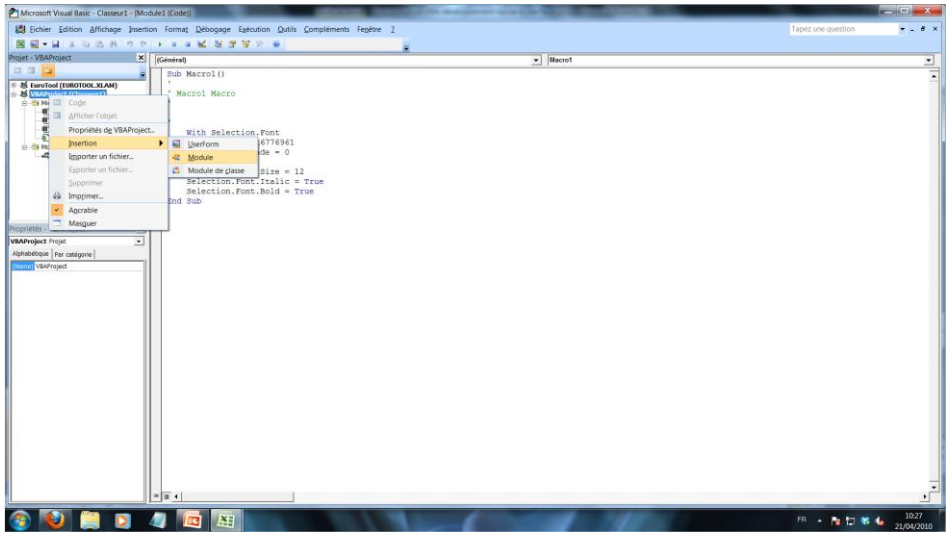
L'ÉDITEUR VBE ET LE MODÈLE OBJET VBA



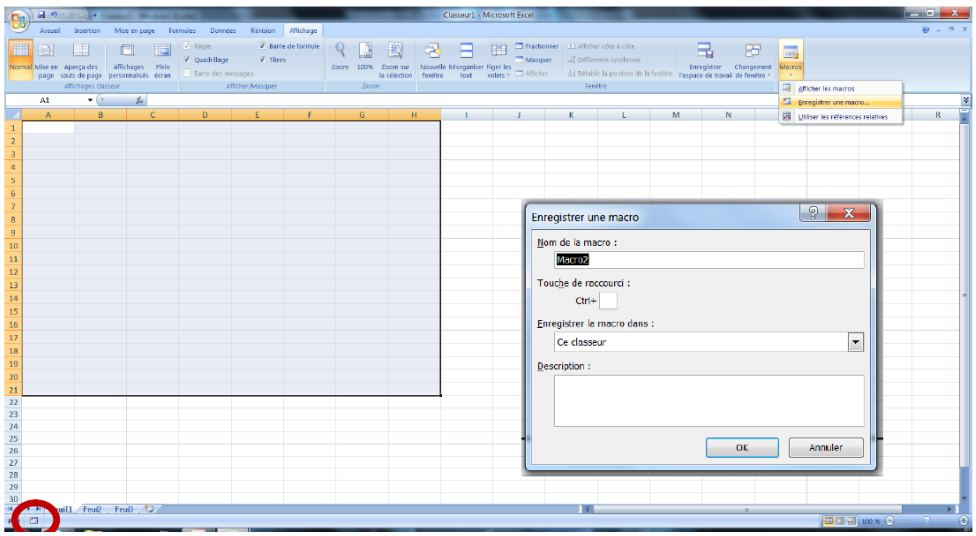

Microsoft Visual Basic pour Applications - Classeur1
Fichier Edition Affichage Insertion Format Débogage Exécution Outils Compléments Fenêtre ?
Projet - VBAProject
VBAProject (Classeur1)
Microsoft Excel Objets
Feuil1 (Feuil1)
ThisWorkbook
VBAProject (Classeur2)
Propriétés - Feuil1
Feuil1 Worksheet
Alphabétique Par catégorie
(Name) Feuil1
DisplayPageBreak False
DisplayRightToLef False
EnableAutoFilter False
EnableCalculation True
EnableFormatCon True
EnableOutlining False
EnablePivotTable False
EnableSelection 0 - xlNoRestricti
Name Feuil1
ScrollArea
StandardWidth 10,71
Visible -1 - xlSheetVisib


ALT F11
With Selection.Font
.Color = -16776961
.TintAndShade = 0
End With
Selection.Font.Size = 12
Selection.Font.Italic = True
Selection.Font.Bold = True
End Sub

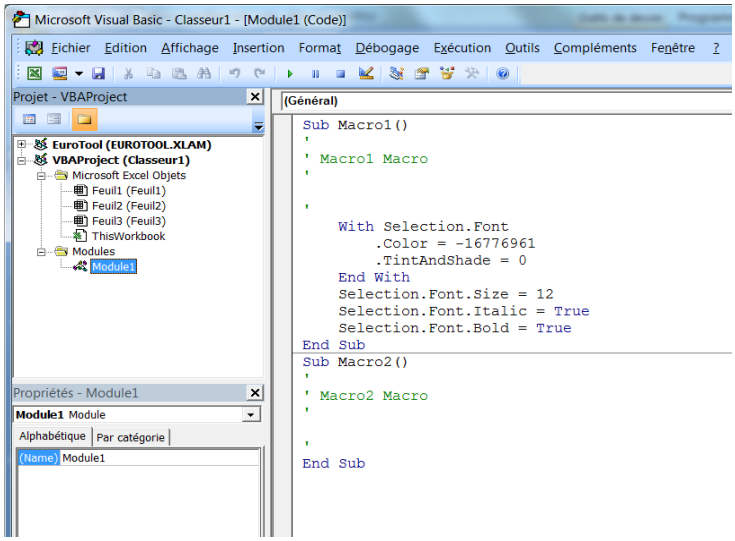
Insertion d'un module dans l'éditeur



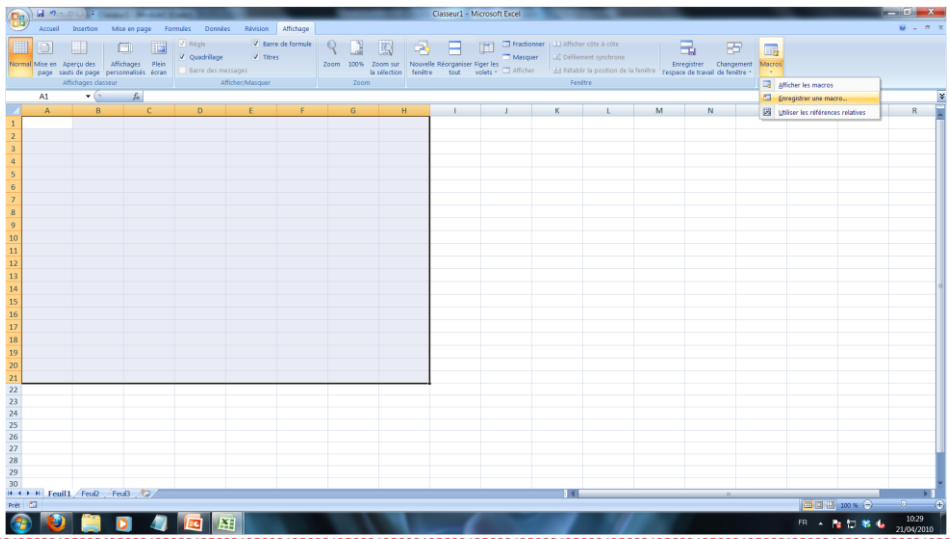
Enregistrer une macro



Impact dans l'éditeur



Enregistrer une macro



Comment accéder au code VBA Excel ?

- Si vous souhaitez aller plus loin que la simple fonction d'enregistrement, Excel vous permet de programmer vos propres macros via le langage VBA.
- Pour ce faire, il vous suffit de cliquer sur l'onglet « Développeur » puis sur « Macros ». Vous pouvez alors donner un nom à votre futur programme, et enfin cliquer sur « Créer ». L'éditeur Visual Basic Editor, qui constitue votre environnement de programmation, s'ouvre alors automatiquement.
- Vous pouvez également accéder directement à l'éditeur de code VBA en utilisant le raccourci clavier Alt + F11.

- L'environnement de programmation se décompose en trois zones principales :
- une zone « Projet » pour naviguer d'un projet à l'autre ;
- une zone « Propriétés » pour modifier certaines caractéristiques des objets VBA ;
- l'éditeur de code VBA pour coder.

Voici quelques exemples de macros VBA simples, utilisées par les financiers :

- l'actualisation des données d'une table de calcul ;
- le pricing de certains produits financiers complexes ;
- la génération de reporting via des macros de "mise en page" ;
- l'exécution automatique d'une macro lors de l'ouverture d'une feuille de calcul ;
- la protection d'une ou plusieurs feuilles de calcul via un identifiant et un mot de passe.

- Les macros VBA permettent également aux professionnels de la finance d'automatiser des tâches bien plus complexes, notamment :
- générer des simulations de modèles économiques ;
- créer des outils de détection et de gestion des risques ;
- mettre en place des techniques d'analyse fondées sur divers paramètres

VBA Concepts de Base: Le modèle Objet

- Avant d ’aller plus loin regardons les concepts de base à connaitre.
- Cela ne vous empêchera pas de passer aux chapitres suivants
- Vous pouvez revenir à ces concepts tôt ou tard.

Programmation modulaire

- De nos jours, l'un des plus grands défis pour un développeur est de ne pas devoir réinventer la roue à chaque nouvelle application.
- C'est pourquoi le développeur moderne est un développeur de **modules.**
- On développe des modules autonomes et plus ou moins génériques que l'on assemble pour former une application.
- La modularité et la réutilisation du code est un des principes de bases de la POO.
- Il est plus facile de débugger des modules de petite tailles qu'une grosse application.
- En dehors de la programmation objet il est aussi très pratique d'utiliser des bibliothèques de code, qui, **une fois que l'on a résolut le problème des liens cassé**, devient indispensable.
- Cela permet d'obtenir du code de meilleurs qualité, plus facile à maintenir, plus robuste.

VBA, un Langage orienté objet ?

- Disons plutôt, par égard pour les puristes, que VBA n'est pas un langage orienté objet mais qu'il permet de faire de la Programmation Orientée Objet (POO), malgré quelques limitations non utilisées pour nous:
  - Certaines des caractéristiques de la POO, comme **l'héritage** ou le **polymorphisme** , ne sont pas présents.
    D'autres part, les classes ne peuvent pas être directement instanciées si elle se trouve dans un complément (*.mda).

VBA, un Langage orienté objet ?

- Disons plutôt, par égard pour les puristes, que VBA n'est pas un langage orienté objet mais qu'il permet de faire de la Programmation Orientée Objet (POO), malgré quelques limitations non utilisées pour nous:
  - Certaines des caractéristiques de la POO, comme **l'héritage** ou le **polymorphisme** , ne sont pas présents. Il est impossible de créer des classes dérivées.
    La surcharge de méthode est impossible, puisqu'il est impossible en VB d'utiliser le même nom pour plusieurs méthodes dans une même classe. Nous verrons le cas particulier des procédures Property
    D'autres part, les classes ne peuvent pas être directement instanciées si elle se trouve dans un complément (*.mda). C'est à mon avis la plus grosse lacune et surtout la plus absurde, étant donné que la réutilisation du code est un des chevaux de bataille de la POO. Il est cependant possible d'utiliser cette classe si le code qui instancie l'objet se trouve dans un modules de code simple dans le même mda que la classe. (**A mettre à jour**)

Le Modèle objet ? Objet, Propriétés et méthodes

- Les développeurs organisent les objets de programmation selon une hiérarchie, qui est appelée **modèle objet de l'application.**
- La définition d'un **objet** est appelée une **classe**, ce qui explique que vous verrez peut-être ces deux expressions utilisées indifféremment.
- D'un point de vue technique, **une classe est la description ou le modèle utilisé pour créer, ou instancier, un objet.**
- Une fois qu'un **objet** existe, vous pouvez le manipuler en définissant ses **propriétés** et en dénommant ses **méthodes**.
  - Si vous pensez à l'objet sous la forme d'un nom, les propriétés sont alors les adjectifs décrivant le nom, et les méthodes sont les verbes qui animent le nom.
  - La modification d'une propriété altère la qualité de l'apparence ou le comportement de l'objet. L'appel à l'une des méthodes d'objet conduit l'objet à effectuer une action

Le Modèle objet ? Objet, Propriétés et méthodes

- Une fois qu'un **objet** existe, vous pouvez le manipuler en définissant ses **propriétés** et en dénommant ses **méthodes**.
  - Si vous pensez à l'objet sous la forme d'un nom, les propriétés sont alors les adjectifs décrivant le nom, et les méthodes sont les verbes qui animent le nom.
  - La modification d'une propriété altère la qualité de l'apparence ou le comportement de l'objet. L'appel à l'une des méthodes d'objet conduit l'objet à effectuer une action
- Word, par exemple, comporte un objet Application de niveau supérieur qui contient un objet Document. L'objet Document contient des objets Paragraphe et ainsi de suite. Les modèles objet reflètent à peu près ce que vous voyez dans l'interface utilisateur. Il s'agit d'une carte conceptuelle de l'application et de ses capacités.

Méthodes 1/2

Dans Word, par exemple, vous pouvez modifier les **propriétés** et les **méthodes** du document en cours d'appel à l'aide de la propriété **ActiveDocument** de l'objet **Application**.

Cette propriété **ActiveDocument** renvoie une référence à l'objet **Document** qui est actuellement actif dans l'application **Word**.

« Renvoie une référence à » signifie « vous donne accès à. »

Le code suivant effectue exactement ce qu'il dit ; c'est-à-dire enregistre le document actif dans application.

Application.ActiveDocument.Save

*Lire le code de gauche à droite, « Dans cette Application, avec le Document référencé par ActiveDocument, appelez la méthode **Enregistrer**. »*

Méthodes 2/2

Application.ActiveDocument.Save

*Lire le code de gauche à droite, « Dans cette Application, avec le Document référencé par ActiveDocument, appelez la méthode **Enregistrer**. »*

Cela ne nécessite aucune instruction détaillée de votre part. Vous donnez l'instruction à un objet **Document** d'**Enregistrer**, aucune autre entrée n'étant requise de votre part.

Si une méthode requiert plus d'informations, celles-ci sont appelées **paramètres**. Le code suivant exécute la méthode **SaveAs**, ce qui nécessite un nouveau nom de fichier.

Application.ActiveDocument.SaveAs ("New Document Name.docx")

Les valeurs figurant entre parenthèses après un nom de méthode sont les paramètres. Dans ce cas, le nouveau nom de fichier est un paramètre pour la méthode **SaveAs**.

Propriétés

- On utilise la même syntaxe pour définir une propriété et pour la lire.
- Le code suivant exécute une méthode pour sélectionner la cellule A1 dans Excel, puis pour définir une propriété afin de placer quelque chose dans cette cellule.

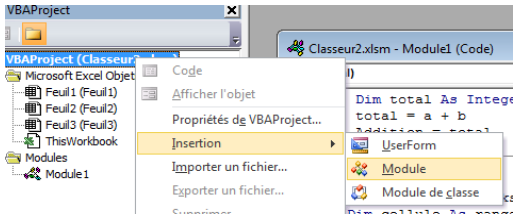
```
Application.ActiveSheet.Range("A1").Select

Application.Selection.Value = "Hello World“
```

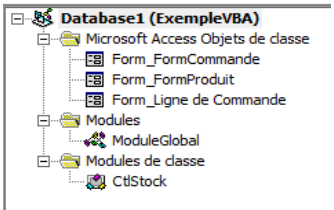
  - À la première ligne de l'extrait de code figure **l'objet Application**, sous Excel cette fois-ci, puis **ActiveSheet**, permettant d'accéder à la feuille de calcul active.
  - Vient ensuite un terme moins familier, **Range (Plage),** qui signifie « définir une plage de cellules de cette façon. »
  - ***Le code demande à l'élément Plage de se créer avec simplement A1 comme étant son ensemble défini de cellules.***
  - **En d'autres termes, la première ligne de code définit un objet, la Plage, et exécute une méthode sur celui-ci pour le sélectionner. Le résultat est automatiquement archivé dans une autre propriété de l' Application appelée Sélection**.
  - La deuxième ligne de code affecte la propriété Valeur de Selection au texte « Hello World » et cette valeur apparaît dans la cellule A1.

VBA Concepts de Base:
Le modèle Objet dans VBA

VBA : Modules

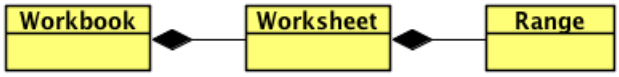
- Modules contiennent le code VBA pour un document
- Plusieurs types de modules
  - MS Objets
    - Evènements
  - Modules standards
    - Procédures utilitaires
  - Modules de classe
    - Classes crées par l'utilisateur



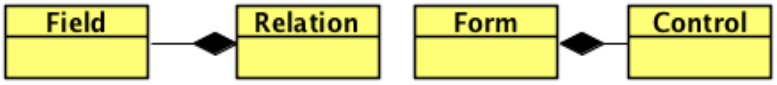


VBA : Modèle OO

- **Modèle Objets de VBA**
- Les éléments manipulés par VBA sont des **objets**
  - Worksheet (feuille de calcul), Range (cellules), Form (formulaire), Report (état), RecordSet (requête)…
- Il est possible de créer de nouvelles classes → **modules de classe**
- Modèle OO Excel

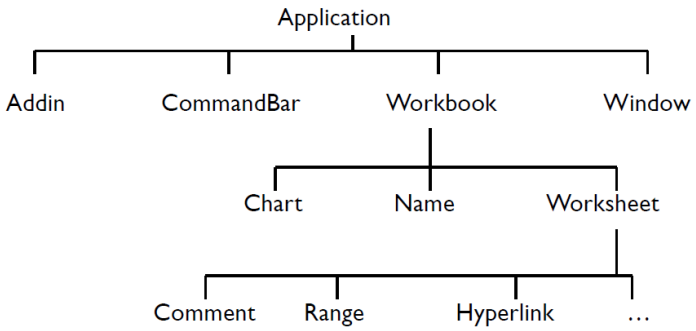


- Modèle OO Access



Le modèle objet dans VBA

- Un objet est constitué d'attributs (ou propriétés) et de méthodes qui lui sont associées
- Les objets existants sont constitués en hiérarchie (relation de composition)



Ecriture des macros – Les principaux objets

- Ecrire directement des macros est simple une fois assimilé la philosophie de l'approche, et identifié les principaux objets et l'accès à leurs propriétés et méthodes (l'enregistreur peut nous y aider).

Classeurs

```
Workbooks("classeur1.xlsm").Activate
```

Activer (sélectionner) le classeur dont le nom de fichier est "classeur1.xlsm"

Feuilles

```
Sheets("Feuil1").Activate
```

Dans le classeur courant, activer la feuille de calcul dont le nom est "Feuil1" (visible dans la languette au bas de la feuille)

```
Workbooks("classeur1.xlsm").Sheets("Feuil1").Activate
```

On peut combiner les écritures.

Cellules

```
Cells(1,1).Value = 15
```

```
Sheets("Feuil1").Cells(1,1).Value = 15
```

Dans la feuille courante du classeur courant, insérer la valeur 15 dans la cellule ligne n°1, colonne n°1
c.-à-d. en A1, les coordonnées sont absolues ici. !

De nouveau, on peut combiner.

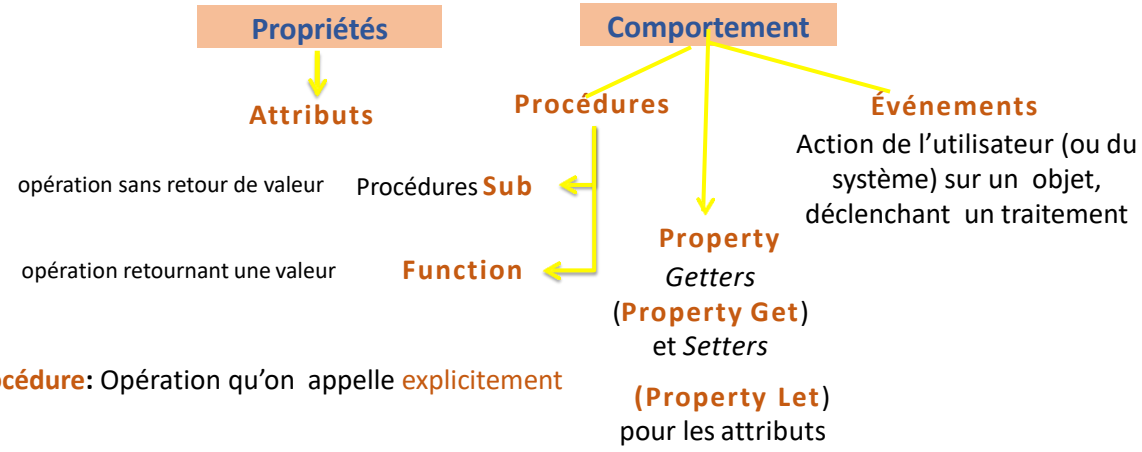
Ecriture des macros – Les principaux objets

- Dans Excel, il est courant de manipuler des cellules, des plages de cellules, des feuilles de calcul et des classeurs. Dans cet article, vous allez découvrir les instructions utilisables pour les manipuler. Il ne s'agit que d'une introduction : vous irez beaucoup plus loin dans les articles à venir.
- Les instructions suivantes permettent de sélectionner des feuilles, cellules, plages nommées, lignes et colonnes :

```
Sheets("Feuil2").Activate 'Activation de la feuille Feuil2
Range("A3").Select 'Sélection de la cellule A3
Range("B2:F6").Select 'Sélection du bloc de cellules B2 à F6
Range("B2,F6").Select 'Sélection des cellules B2 et F6
Range("unePlage").Select 'Sélection de la pahe nommée unePlage
Cells(4,2).select 'Sélection de la cellule à l'intersection de la ligne 4 et de la colonne 2
Range("3:5").Select 'Sélection des lignes 3 à 5
Rows("3:5").Select 'Identique à l'instruction précédente
Range("C:L").Select 'Sélection des colonnes C à L
Columns("C:L").Select 'Identique à l'instruction précédente
```

VBA : Modèle OO

- Un objet VBA possède :



Les collections

- Concept clé
- On rajoute un « s »!
  - Workbooks : collection des objets Workbook
  - Worksheets : collection des objets Worksheet
  - … etc.

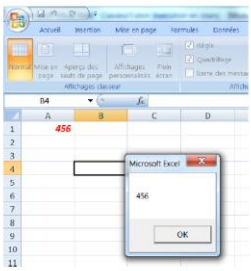
- Faire appel à un élément d'une collection: 2 méthodes:
  - Appel par le nom de l'élément
    - Ex: Worksheets("Feuil1")
  - Appel par l'indice
    - Ex: Worksheets(1)

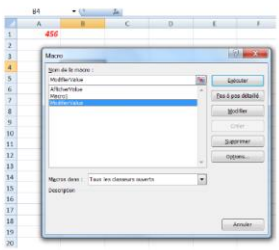
Hiérarchie : Accéder aux objets

- **Opérateur point (.)**
  - Exemple:
    Application.Workbooks("Classeur1.xlsx").Worksheets(1).Range("A1").Value=9
- **Simplification: par exemple si Classeur1.xlsx est le classeur actif:**
  - Worksheets(1).Range("A1").**Value**=9

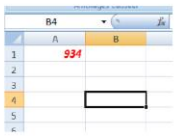
Propriétés d'un objet

```
Sub AfficherValue()
'
Contents = Worksheets("Feuil1").Range("A1").Value
MsgBox Contents


    ActiveWorkbook.Save
End Sub
```





```
Sub ModifierValue()
    Worksheets("Feuil1").Range("A1").Value = 934
End Sub
```



Méthode d'un objet

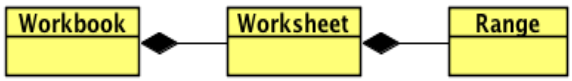
- **Action relative à un objet**

- **Exemples:**
  - Worksheets("Feuil1").Activate
  - Range("A1").Copy Range("B1")

- **Une méthode prend en compte 0, 1 ou plusieurs arguments.**
  - Le premier argument est séparé de la méthode par un espace, les arguments sont séparés entre eux par des virgules
  - OU utilisation des parenthèses

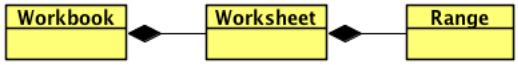
VBA : accès aux données EXCEL

- Les données dans un document Excel (objet ThisWorkbook) s'organisent en feuilles de calcul (objets Worksheet) et en cellules (objets Range)



- Quelques objets importants
  - ThisWorkbook : document courant
  - ActiveWorkbook : document en 1er plan (actif)
  - ActiveSheet : feuille de calcul active
- Les collections vont permière l'accès aux différents objets de la classe correspondante
  - Worksheets("Feuil1") → accès à la feuille « Feuil1 »

VBA : accès aux don



- **Range**
  - La notion de Range est essentielle dans Excel.
  - Un objet Range est une région dans une feuille contenant une ou plusieurs cellules
  - Les opérations de la classe Range vont nous première d'interagir ou de modifier les cellules

```
ActiveSheet.Range("A1").Value
ActiveSheet.Cells(1, 2).Value
```

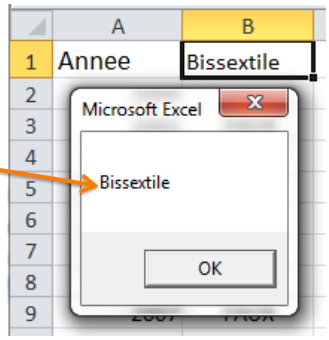
L'opération **Value** récupère la **valeur de la cellule**

La collection **Cells** contient un ensemble d'objets **Range**

Attention :
les collections démarrent en **1 (et pas 0)**

VBA : accès aux données EXCEL

```
Sub UnRange()
    MsgBox ActiveSheet.Range("A1").Value
    MsgBox ActiveSheet.Cells(1, 2).Value
End Sub
```



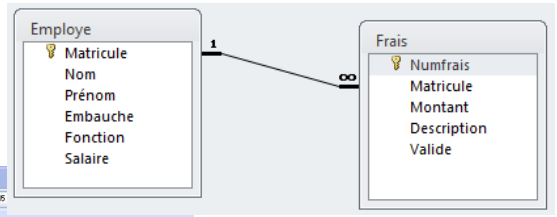

```
Private Sub ChangeA1()
    Worksheets("Feuil1").Cells(1, 1).Font.Size = 12
End Sub
```

changement de la taille de la police

	A	B
1	Annee	Bissextile
2	2000	VRAI
3	2001	FAUX

VBA : accès aux données ACCESS

- Exemple Employés



Formulaire
« Employe »

(mode Création)

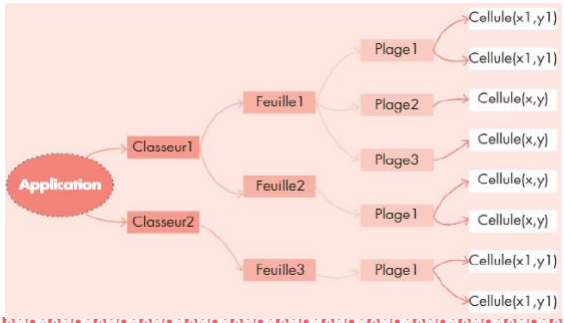
II. Langage VBA

VBA : quoi & pourquoi ?

- VBA : quoi ?
  - **Langage** et **environnement** de programmation Orienté Objets
  - **Attaché aux documents** MS Office
- VBA : pourquoi ?
  - Associer **un comportement actif** à des documents Office
    - Calculs, vérifications, etc.
- Visual Basic pour Applications, VBA, est la solution de programmation proposée avec les applications de la suite Office.
- La connaissance de VBA permet à l'utilisateur d'Excel de tirer pleinement profit du tableur de Microsoft en développant les capacités et les fonctionnalités pour ses besoins spécifiques.
- Maîtriser Visual Basic pour Applications, c'est à coup sûr améliorer grandement sa productivité et celle de son entreprise.

Modèle hiérarchique

- **Le VBA est un langage orienté objets**, ou tout est comme objet!
- Aussi **Excel** est vu comme une **hiérarchie d'objets**.
  - En haut de la hiérarchie du modèle d'objets, on trouve:
    - l'application Excel elle-même, puis les classeurs qui la composent, les feuilles de calcul des classeurs, les plages de cellules et, pour terminer, la plus petite unité: la cellule.



VBA : documents MS Office

- Un document MS Office est composé des plusieurs éléments…

MS Excel **Fichier Excel** **Classeur**

- Feuilles
- Données
- Formules
- **Modules (VBA)**

MS Access **Fichier Access** **Base de données**

- Tables (données)
- Requêtes
- Formulaires
- États (rapports)
- **Modules (VBA)**

Désignation des cellules en VBA

- Les plages et les cellules manipulées par l'utilisateur au sein d'une feuille de calcul sont désignées par les instructions données dans le tableau suivant:

Instructions	Plages ou cellules manipulées
Range("A2")	Cellule A2 de la feuille active
Range("A2:B5")	Plage A2:B5 de la feuille active
Range("A2:B5").Cells(i, j)	Cellule de la $i^{ème}$ ligne et de la $j^{ème}$ colonne relative à la plage A2:B5
Range("A2:B5").Cells(1, 1)	Cellule A2 car Cells(i,j) il commence à 1, 1
Range("A2:B5").Cells(3,2)	Cellule B4

- **Range** peut faire référence à une plage de cellules contenant une ou plusieurs cellules.
- tandis que **Cells** ne fait référence qu'à une seule cellule, toujours en notation L1C1.

  Exemple: les instructions suivantes sont similaires:
  - Application.Workbooks(1).Sheets(3).Range("A4")
  - Application.Workbooks(1).Sheets(3).Cells(4,1)

La notation objet

- Un objet possède toujours des propriétés et des méthodes, les propriétés étant des caractéristiques et les méthodes des actions exécutées sur l'objet. La convention d'écriture est la suivante :
  - **Nom_Objet.Propriété** ou **Nom_Objet.Méthode**
- Exemple 1:
  - On veut définir la taille de la colonne B à 10.
  - On sélectionnera la propriété qui désigne la taille d'une colonne, tel que **Range("B").ColumnWitdh = 10**.
  - On veut copier la colonne B.
    - On va donc exécuter une **action**.
    - On choisira la **méthode *Copy***, tel que **Range("B") .Copy**.

Visual Basic possède tous les attributs d'un langage de programmation

Données typées. Visual Basic propose les types usuels de la programmation : **entier, réels, booléens, chaîne de caractères**. VBA propose des types spécifiques comme Range (plage de cellules) pour Excel par exemple.

Structures avancées de données. Gestion des collections de valeurs (**énumérations**, **tableaux**) et des objets structurés (**enregistrements**, **classes**).

Séquences d'instructions, c'est la base même de la programmation, pouvoir écrire et exécuter une série de commandes sans avoir à intervenir entre les instructions.

Structures algorithmiques : les branchements conditionnels et les boucles.

Les outils de la programmation structurée : pouvoir regrouper du code dans des **procédures** et des **fonctions**. Organisation du code en **modules** et possibilité de distribuer ces dernières.

Visual Basic n'est pas « case sensitive », il ne différencie pas les termes écrits en minuscule et majuscule.

Les variables et les constantes

Les variables: Déclaration et initialisation

- Les variables et constantes en VBA permettent de stocker des données et informations qui seront réutilisées dans votre programme (code). Elles sont donc indispensables en VBA.
  - Si elles sont bien maitrisées, votre code sera performant et fiable. Vos fichiers Excel en seront donc plus efficaces.
- Les données en entrée sont appelées des **variables** si leurs valeurs peuvent varier pendant le traitement algorithmique.
- À l'inverse, les valeurs de certaines données en entrée ne changent jamais. Dans ce cas. on les appelle des **constantes**.

Les variables: Déclaration et initialisation

- **Définition**
- Une **variable** est un espace de mémoire contenant une valeur et identifié par un nom.
  C'est dans les variables que sont stockées les données en entrée.
  Dans un programme, on peut également avoir des variables nécessaires aux traitements.
- Une variable appartient à un **type de données**. c'est-à dire qu'elle peut contenir uniquement des valeurs de ce type.

Variables et premières instructions

- Les variables correspondent à des identifiants auxquels sont associés des valeurs d'un type donné. Elles matérialisent un espace mémoire avec un contenu que l'on peut lire ou écrire.

Déclaration d'une variable

```
Dim v As Double
```

- `Dim` : Dimensionner. Réserver un espace mémoire.
- `v` : Identifiant de la variable, utilisable dans la suite du programme
- `As` : Indication du type
- `Double` : Type associé à la variable

Affectation. Attribuer une valeur à la variable

```
v = 2,5
```

= est le symbole d'affectation.
A gauche de = on modifie le contenu dans une variable, à droite on lit le contenu d'une variable. C'est pour cette raison que l'instruction v = v + 1 est licite.

Opération et **affectation**

```
x = v * 2
```

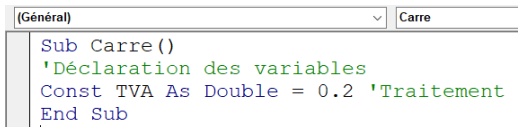
La valeur 5 est écrite dans la variable x qui doit être déclarée au préalable.

Les variables: Exemples

- Exemple: On veut déclarer une constante nommée TVA et initialisée à 20 %.

```
Sub Carre()
    'Déclaration des variables
    Const TVA As Double = 0.2  'Traitement
End Sub
```

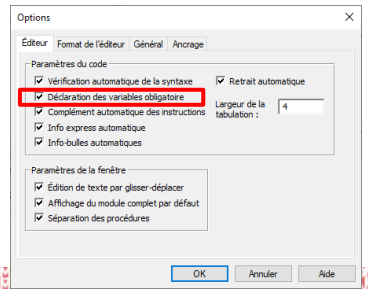
```
Sub Carre()

' Déclaration des variables
Dim TVA As Double

'Traitement
TVA = 0.2

End Sub
```

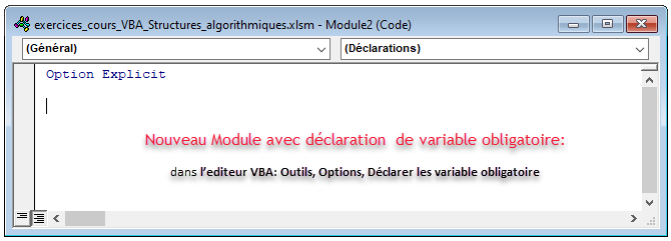


Type de données en VBA

Variables et premières instructions

- Sous VBA, la déclaration des variables n'est pas obligatoire. Cependant, je vous conseille de le faire.
  - Pour ne pas l'oublier, vous pouvez le rendre obligatoire dans votre éditeur VBE en cochant : **Déclarer les variables obligatoires** dans **l'éditeur VBA:**
    - **Outils, Options, Déclarer les variables obligatoires**
  - **Si la case est cochée, l'instruction « Option Explicit » est ajoutée dans les déclarations générales de tout nouveau module.**





Les types de données

Les variables: Type de données en VBA

- Une variable appartient à un type de données.
  - c'est-à dire qu'elle peut contenir uniquement des valeurs de ce type.
- Une variable est toujours déclarée (assignée à un type de données) avant d'être utilisée et parfois initialisée (valorisée avec une valeur de départ).

Nom	Type	Détails
Integer	Numérique	Nombre entier de –32'768 à 32'767
Long	Numérique	Nombre entier de –2'147'483'648 à 2'147'483'647
Double	Numérique	Nombre à virgule flottante de –1,79769313486232D308 à 1,79769313486232D308
String	Texte	Texte
Date	Date	Date et heure
Boolean	Boolean	*True* (vrai) ou *False* (faux).

Type de données en VBA

Les variables: Type de données en VBA

Type de données	Utilisation
String	Pour les chaînes de caractères. Peut contenir jusqu'à 2 milliards de caractères
Integer	Compteur numérique allant de -32 768 à 32 767
Long	Compteur numérique allant de -2 147 483 648 à 2 147 483 647
Variant	Quand le type ne peut pas être définie préalablement
Date	Dates (du 1er janvier 100 au 31 décembre 9999)
Boolean	Retourne une valeur Vrai/True (-1) ou Faux/False (0)
Byte	Pour un compteur allant de 0 à 255
Double	Pour gérer des données à virgule
Currency	Permet de gérer des devises allant jusqu'à 4 chiffres après la virgule
Object	Représente un Objet.

Type de données

- Le type de données définit le type d'opérateurs qu'on peut leur appliquer.
- Numérique qui peut être réel (double) ou entier (long). Les opérateurs applicables sont : +, -, *, / (division réelle), \ (division entière), mod (modulo)
- Exemple : 5 / 2 → 2.5 ; 5 \ 2 → 2 ; 5 mod 2 → 1

- Booléen (boolean) qui ne prend que deux valeurs possibles : True et False. Les opérateurs sont : not, and, or.
- Exemple : True and False → False
- Chaîne de caractères (string) qui correspond à une suite de caractères délimitée par des guillemets '' ''. Les opérateurs possibles sont la concaténation, la suppression d'une sous- partie, la copie d'une sous-partie, etc.
- Exemple : ''toto'' est une chaîne de caractères, toto on ne sait pas ce que c'est (pour l'instant)
    - Habituellement, les opérations font intervenir des données de type identique et renvoie un résultat du même type.

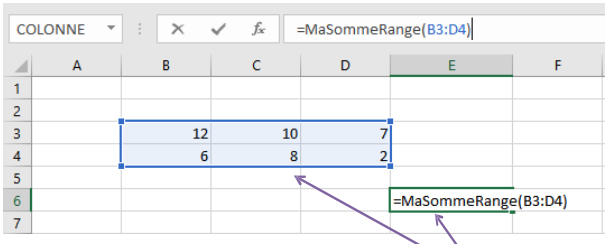
Type de données

Type	Valeurs
Boolean	Vrai, faux
Integer	Entiers
Long	Entiers
Single	Réels
Double	Réels
Currency	4 chiffres après la ,
Date	1/1/100 à 31/12/9999
String	Chaines de caractères
Object	Tout objet
Variant	N'importe quel type

Le type RANGE

Le type « plage de cellules » spécifique à Excel

- Le type RANGE désigne une plage de cellules, c'est un type spécifique à Excel.
- Exemple Coin en haut et à gauche de la plage de cellules passée en paramètre de la fonction = coordonnée (1, 1) c.-à-d. ligne n°1 et colonne n°1, quelle que soit la position absolue de la plage dans la feuille de calcul (ici le coin nord-ouest est en B3)



La fonction MaSommeRange() est censée faire la même chose que la fonction standard SOMME() d'Excel.

Un bloc de cellules (B3:D4) est passé en paramètre de la fonction. Ce bloc est forcément rectangulaire, avec, ici : 2 lignes et 3 colonnes.

Exploiter le type Range en VBA

- Entrée :plage (range)
- Sortie :S (réel)
- Calcul :Somme des valeurs

```
'Travail sur le type Range
Public Function MaSommeRange(plage As Range) As Double
'variables intermédiaires
Dim s As Double, i As Long, j As Long
'initialisation de la somme
s = 0
'parcours de la plage de cellules
For i = 1 To plage.Rows.Count Step 1 'lignes
    For j = 1 To plage.Columns.Count Step 1 'colonnes
        'lecture des valeurs et somme
        s = s + plage.Cells(i, j).Value
    Next j
Next i
'renvoyer le résultat
MaSommeRange = s
End Function
```

Lignes et colonnes commencent à l'indice 1, quelle que soit la position de la plage dans la feuille.

Nombre de lignes de la plage de cellules.

Nombre de colonnes.

Accès à la valeur (Value) de la cellule : ligne n°i, colonne n°j

Une boucle adaptée pour les plages de cellules – For Each

La boucle For Each est adaptée au parcours des collections. Or une plage de cellules est une collection de cellules.

```
'Travail sur le type Range avec un For Each
Public Function MaSommeRangeEach(plage As Range) As Double
'variables intermédiaires
Dim s As Double, cellule As Range
'initialisation de la somme
s = 0
'parcours de la plage de cellules
For Each cellule In plage
    s = s + cellule.Value
Next cellule
'renvoyer le résultat
MaSommeRangeEach = s
End Function
```

Une cellule est une plage de cellules avec une seule cellule.

plage fait figure de collection à traiter. Qu'importe le sens du parcours ici (ligne par ligne, ou colonne par colonne).

C'est bien la valeur contenue dans la cellule qui est exploitée pour la somme.

Le type Variant

Type spécial qui peut contenir toutes sortes de valeurs

- Le type de variant peut gérer tout type de valeurs. Il est très souple, particulièrement commode quand on ne connaît pas à l'avance le type à utiliser. Mais attention, il ne faut pas en abuser, il est très lent parce que multiplie les vérifications à chaque accès à la variable correspondante.

Entrée : a, b (réel)
Sortie : a/b si b ≠ 0, « division par zéro » sinon

Un coup, la fonction renvoie un réel, un autre coup elle doit renvoyer une chaîne de caractères.

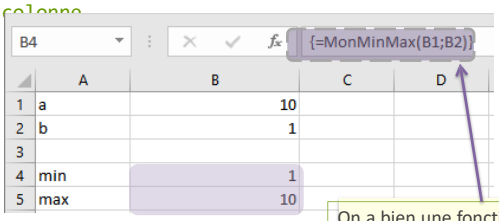
```
'utilisation du type variant
Public Function MaDivision(a As Double, b As Double) As Variant
'var. intermédiaire
Dim resultat As Variant
'calcul
If (b <> 0) Then
    resultat = a / b
Else
    resultat = "division par zéro"
End If
'renvoyer le résultat
MaDivision = resultat
End Function
```

Dans la même variable resultat, de type variant, on peut affecter un réel et une chaîne de caractères.

Le type Variant est vraiment très souple

- On peut s'en servir pour renvoyer un tableau. Une fonction peut donc renvoyer plusieurs valeurs d'un coup, à l'instar des fonctions matricielles d'Excel (il faut valider la saisie de la fonction avec la séquence de touches CTRL + MAJ + ENTREE).

```
'Renvoyer plusieurs valeurs
Public Function MonMinMax(a As Double, b As Double) As Variant
'un tableau interne - matrice 2 lignes et 1 colonne
Dim tableau(1 To 2, 1 To 1) As Double
'identifier le min et le max
If (a < b) Then
  tableau(1, 1) = a
  tableau(2, 1) = b
Else
  tableau(1, 1) = b  tableau(2, 1) = a
End If
'renvoyer le tableau
MonMinMax = tableau
End Function
```



On a bien une fonction matricielle comme peuvent en témoigner les accolades { } qui encadrent l'appel de la fonction.

Les opérateurs mathématiques et logiques

- Les opérateurs de comparaison confrontent des données de même type, mais le résultat est un booléen

= ; <> ; > ; >= ; < ; <=

égalité

différent

Supérieur strict, supérieur ou égal

inférieur strict, inférieur ou égal

- Exemples

5 > 2 → True

5 > "toto" → illicite

5 <> 5 → False

"toto" > "tata" → True

Licite. Comparaison de gauche à droite basée sur le code ASCII. Arrêt des comparaisons dès que l'indécision est levée.

Fonctions mathématiques

- Valeur absolue: Abs(-9) retourne 9
- Signe: Sgn(-18) retourne –1 (ou 0 ou 1)
- Troncation:
  - Int(13.12) retourne 13
  - Int(-14.8) retourne –15
    - Tronque à l'entier inférieur le plus proche
- Partie entière: Fix(-18.3) = -18
- Fix(18.3) = 18

  Enlève la partie décimale

Fonctions mathématiques

- Sqr, Exp, Log
  - Sqr(4) retourne 2, Exp(5) retourne 148.413…,
    Log(9) retourne 2.197224… (en base e)
- Nombres aléatoires
  - Rnd retourne un nombre aléatoire entre 0 (compris) et 1 (non compris)
  - a = Rnd a peut valoir 0.12131441
  - Int((b – a + 1) * Rnd + a) retourne un nombre aléatoire entier entre a et b
- Sin, Cos, Tan, Atn (arc-tangente)

Fonctions de dates

- Date retourne la date actuelle
- Time retourne l'heure courante
  - Date et Time peuvent retourner des chaînes de carctères String
- DateSerial retourne une valeur unique pour une date donnée, sous forme Variant
  - dv1 = DateSerial(2003, 4, 22)
    dv2 = DateSerial(1928, 5, 3)
    dv1 – dv2 représente le nombre de jours entre ces deux dates
- Day, Month et Year retourne respectivement le jour, le mois et l'année d'une date.
  - Year(Date) retourne 2021 cette année (en entier)

FONCTIONS PERSONNALISÉES

Ecriture et utilisation des fonctions personnalisées dans Excel

Programmation des fonctions personnalisées

- Le tableur Microsoft Excel propose à ses utilisateurs un nombre très important de fonctions intégrées telles que **SOMME()**, **MOYENNE()** ou **RECHERCHEV().**
- Mais si vous avez des tâches plus spécifiques à réaliser, il se peut que vous ayez besoin d'une fonction spécifique qui n'existe pas dans la bibliothèque du logiciel. Vous devrez donc la composer vous-même sous la forme d'une fonction définie par l'utilisateur.

Programmation des fonctions personnalisées

- Une fonction personnalisée est une fonction VBA qui peut être appelée dans un classeur Excel.
  Elle prend en entrée des informations en provenance des feuilles du classeur (principalement) et renvoie une valeur insérée dans une cellule (le plus souvent également).
  (C'est comme les fonctions intégrées: somme(), moyenne() etc…)

Formalisme

```
Function NomFonction(paramètres) as type de donnée
```

Est un identifiant qui doit respecter la syntaxe VBA

Les informations que prend en entrée la fonction, elles prennent la forme *nom_parametre as type de paramètre*. Il peut y en avoir plusieurs, ils sont séparés par des « , » dans ce cas.

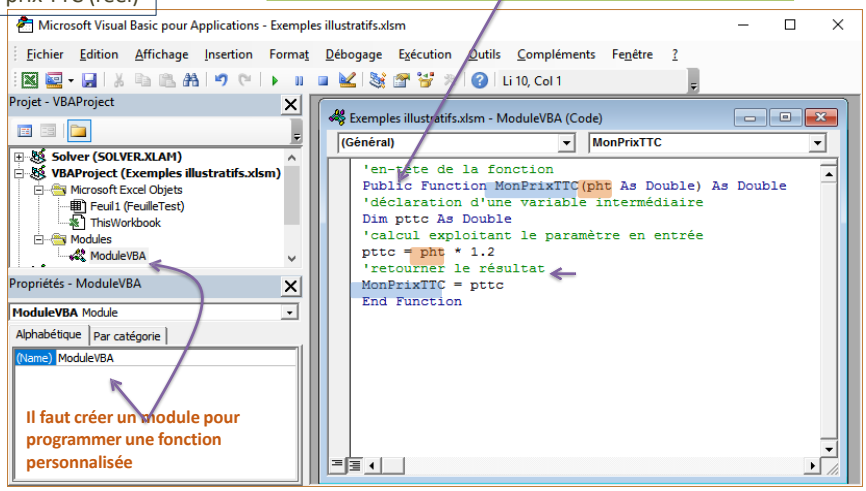
Type de la valeur retournée par la fonction.

Un classeur Excel contenant du code VBA doit être enregistré au format **XLSM**, prenant en charge les macros. Sinon on perd son code.

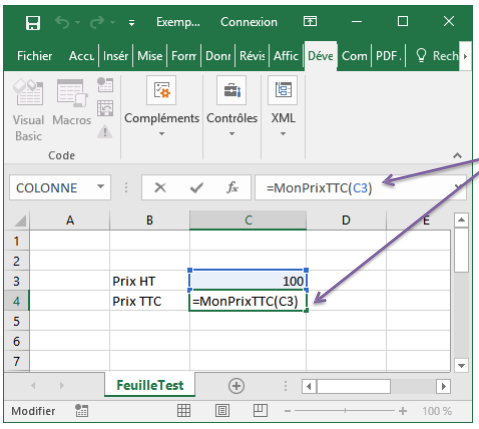
Programmation dans Visual Basic Editor

Entrée : prix HT (réel)
Sortie : prix TTC (réel)

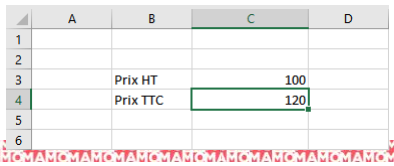
Public pour que la fonction soit visible en dehors du module, notamment dans la feuille de calcul



Utilisation de la fonction dans une feuille Excel

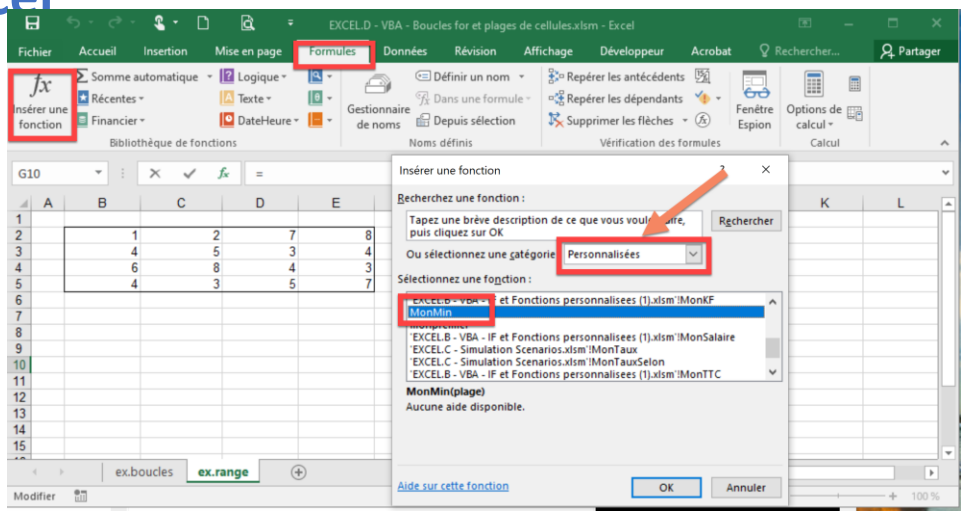


La fonction est insérable dans la feuille de calcul comme n'importe quelle autre fonction Excel. Elle est accessible dans la catégorie « Fonctions personnalisées ».



Le résultat s'affiche une fois la fonction insérée et validée. La fonction est automatiquement appelée à chaque fois que la feuille a besoin d'être recalculée (comme pour les autres fonctions standards d'Excel).

Utilisation de la fonction dans une feuille Excel

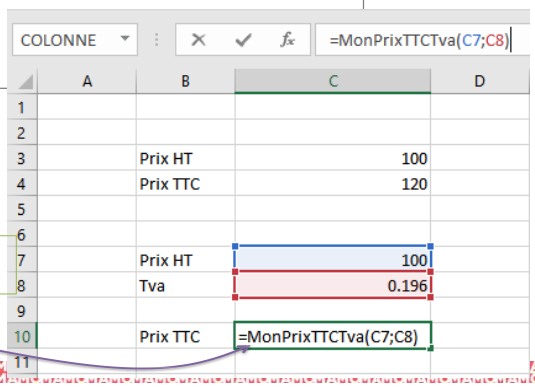


Fonction avec plusieurs paramètres

Entrées : prix HT (réel), tva (réel)
Sortie : prix TTC (réel)

(1) Le séparateur de paramètres est la « , » lors de la définition de la fonction.

```
'fonction avec 2 paramètres
Public Function MonPrixTTCTva(pht As Double, tva As Double) As Double
'déclaration d'une variable intermédiaire
Dim pttc As Double
'calcul exploitant les paramètres en entrée
pttc = pht * (1 + tva)
'retourner le résultat
MonPrixTTCTva = pttc
End Function
```



(2) Très étrangement, il devient « ; » lorsqu'on appelle la fonction dans la feuille de calcul.

EXPLOITER LES FONCTIONS NATIVES D'EXCEL

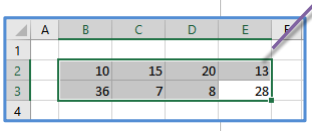
Accéder aux fonctions natives d'Excel dans nos programmes

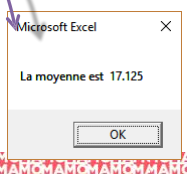
- Excel dispose de fonctions natives puissantes. Nous pouvons y accéder dans nos programmes VBA.

```
Sub MaMoyenneSelection()
'var. intermédiaire
Dim moyenne As Double
'vérifier la sélection
If (Selection.Areas.Count > 1) Then
    MsgBox ("Attention, ce n'est pas une sélection simple")
Else
    'faire calculer la moyenne de la sélection par Excel
    moyenne = Application.WorksheetFunction.Average(Selection)
    MsgBox ("La moyenne est " & Str(moyenne))
End If
End Sub
```

Exemple : Vérifier qu'une sélection est simple (**une seule zone**), puis calculer et afficher la moyenne des valeurs dans une boîte de dialogue.

Noter la syntaxe.





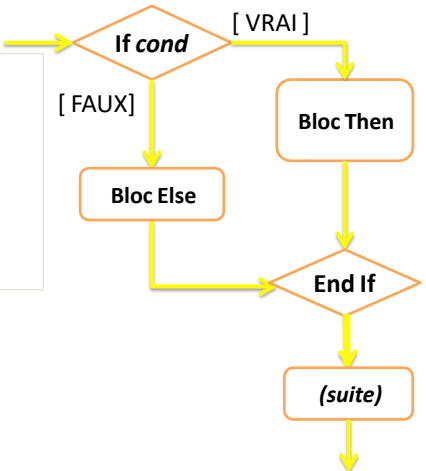
STRUCTURES ALGORITHMIQUES

Plus loin avec la programmation…

Structure alternative simple ou Branchement conditionnel IF

- Permet d'activer une partie du code en fonction de la réalisation d'une condition ou pas.
- SI … ALORS … SINON … FIN SI
- Syntaxe

```
If condition Then
  bloc d'instructions
  si la condition est vraie
Else
  bloc d'instructions
  si la condition est fausse  End If
```



(1) Condition est souvent une opération de comparaison (<,>, <=, >=, = <>)
(2) Plusieurs conditions peuvent se combiner à l'aide des opérateurs logiques « Or » et « And »
(3) La valeur de retour de Condition est de type booléen (True ou False)
(4) Then doit être sur la même ligne que If
(5) La partie Else est facultative (ne rien faire si la condition est fausse)
(6) Il est possible d'imbriquer une autre structure conditionnelle If dans les blocs d'instructions
(7) L'opérateur « Not » renvoie le contraire de l'expression.
(8) S'il y a plusieurs instructions, elles sont séparées par le signe de ponctuation « : ».

La structure alternative simple

Algorithme	VBA

La structure alternative **SI… ALORS… SINON… FINSI** permet des actions sous conditions
SI Condition
ALORS action 1 (ou pas d'action, « rien ») qui s'exécute quand la condition est vérifiée et produit la valeur « vrai »
SINON action 2 (ou pas d'action, « rien ») qui s'exécute quand la condition n'est pas vérifiée et produit la valeur « faux »
FINSI
	If… Then… Else… End If
	If <condition> Then
	<instruction>
	Else <instruction>
	End If
	Le mot-clé « Else » peut être omis quand il y a juste une instruction soumise à condition et pas d'alternative.

Exemple de structure alternative simple

- L'entreprise Alpha accorde une ristourne (RIST) de 6 % aux clients dont le chiffre d'affaires annuel (CA) est supérieur à 10 000 €. Elle se limite à 3 % dans les autres cas.

Calcul du montant de la ristourne (corps de l'algorithme et du programme)	
Algorithme	**VBA**

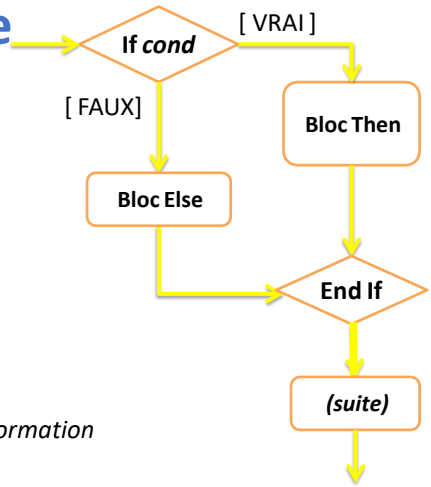
LIRE CA
SI CA >10000
ALORS RIST ← CA*0,06
SINON RIST ← CA*0,03
FIN SI
ECRIRE (RIST)
	CA = InputBox("Donner le montant du CA")
	If CA > 10000 Then
	RIST = CA*0,06
	Else RIST = CA*0,03
	End If
	MsgBox ("le montant de la ristourne est de "& RIST &" €")

Exemple de structure alternative simple

- Entrées : prix HT (réel), catégorie de produit (chaîne) Sortie : prix TTC (réel)

```
Public Function MonTTCBis(pht As Double, cat As String) As Double
'déclarer la variable de calcul
Dim pttc As Double
    'en fonction de la catégorie de produit
    If (cat = "luxe") Then
    pttc = pht * 1.33
    Else
    'la valeur de cat est différente de ''luxe''
    pttc = pht * 1.2
    End If
'renvoyer le résultat
MonTTCBis = pttc
End Function
```

VBA : Instructions de contrôle



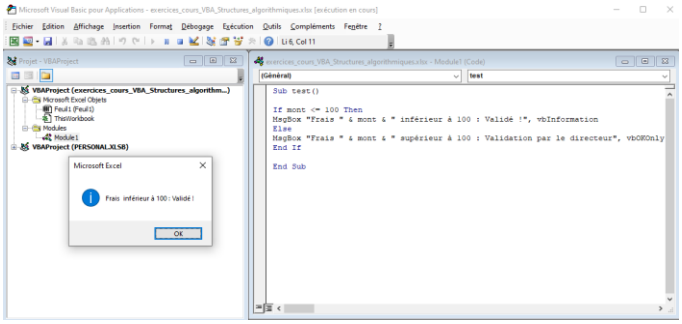
- Instructions conditionnelles : if… else…

```
If condition Then
    'si condition vraie …
Else                          optionnel
    'si condition faux
End If
```

```
If mont <= 100 Then
    MsgBox "Frais " & mont & " inférieur à 100 : Validé !" , vbInformation
Else
    MsgBox "Frais " & mont & " supérieur à 100 : Validation par le directeur", vbOKOnly
End If
```

VBA : Instructions de contrôle

```
Sub test()
   If mont <= 100 Then
      MsgBox "Frais " & mont & " inférieur à 100 : Validé !", vbInformation
   Else
      MsgBox "Frais " & mont & " supérieur à 100 : Validation par le directeur", vbOKOnly
   End If
End Sub
```



VBA : Instructions de contrôle

- Instructions conditionnelles : if…elseIf…else…

```
If condition Then
   'si condition vraie …
ElseIf condition2 Then
   'si condition2 vrai
Else
   'si aucune des
        ' précédentes
End If
```

```
If mont > 0 And mont <= 50 Then
   MsgBox "Frais " & mont & _
      " inférieur à 50 : Validé !", vbInformation
ElseIf mont <= 100 Then
   MsgBox "Frais " & mont & _
   " supérieur à 50 et inférieur à 100 : " _
      & " Validation par le RH",  vbOKOnly
Else
   MsgBox "Frais " & mont & _
   " supérieur à 100 : Validation par le directeur", vbOKOnly
End If
```

VBA : Instructions de contrôle

```
Sub test_InstControle()

   If mont > 0 And mont <= 50 Then

      MsgBox "Frais " & mont & _
      " inférieur à 50 : Validé !", vbInformation
   ElseIf mont <= 100 Then
      MsgBox "Frais " & mont & _
      " supérieur à 50 et inférieur à 100 : " _
      & " Validation par le RH", vbOKOnly
   Else
      MsgBox "Frais " & mont & _
      " supérieur à 100 : Validatioon par le directeur", _
      vbOKOnly
   End If

End Sub
```

La structure alternative imbriquée

- Des structures alternatives peuvent être imbriquées selon le principe suivant

```
If condition Then
  bloc d'instructions
  si la condition est vraie
ElseIf condition then
  bloc d'instructions
  si la condition est fausse
  ….
Else
  Bloc d'instructions
End If
```

La structure alternative imbriquée

- Algorithme et synthèse VBA associée

Algorithme	VBA

SI <condition 1>
ALORS <action 1>
SINON SI <condition 2>
ALORS <action 2>
SINON…
FINSI
FINSI
	If… Then… ElseIf… Else… End If
	If <condition> Then
	<instruction>
	ElseIf <condition> Then
	<instruction>
	…
	Else <instruction>
	End If

La structure alternative imbriquée Exemple

- L'entreprise Alpha accorde dorénavant une ristourne de 8% au-delà de 15 000 €. Elle se limite à 5% entre 10 000 € (compris) et 15 000 €, les autres conditions restent inchangées.

Calcul du montant de la ristourne (corps de l'algorithme et du programme)	
Algorithme	**VBA**

LIRE CA
SI CA >15000
ALORS RIST ⟵ CA*0,08
SINON SI CA ≥ 10000
ALORS RIST ⟵ CA*0,05
SINON RIST ⟵ CA*0,03
FIN SI
FIN SI
ECRIRE RIST
	CA = InputBox("Donner le montant du CA")
	If CA > 15000 Then
	RIST = CA*0,08
	ElseIf CA >= 10000 Then
	RIST = CA*0,05
	Else RIST = CA*0,03
	End If
	MsgBox ("la ristourne est de "& RIST &" €")

La structure de choix Branchement multiple SELECT CASE

- Permet d'activer une partie du code selon les valeurs que peut prendre une variable de contrôle. Peut se substituer au IF (SI) quand le nombre de « SI » imbriqués est excessif. tout dépend de la forme de la condition (*condition composée, on doit passer par un IF*).
- Syntaxe

```
Select Case variable
       Case valeur 1
              bloc d'instructions
       Case valeur 2
              bloc d'instructions
       ...
       Case Else
              bloc d'instructions
End Select
```

(1) Variable est la variable de contrôle, elle peut être de n'importe quel type en VBA, y compris un
- réel ou une chaîne de caractères

(2) Valeur doit être de type compatible avec variable
(3) La partie Case Else est facultative
(4) L'imbrication avec un autre IF ou un autre Select Case (autre variable de contrôle) est possible.

La structure de choix Branchement multiple SELECT CASE

- Algorithme et synthèse VBA associée

Algorithme	VBA
SELON variable CAS Condition1 Action 1 CAS Condition 2 Action 2… CAS Condition n Action n FIN SELON	Select Case Variable_à_tester Case is <condition1> <instruction> 'si la condition 1 est remplie Case is <condition2> <instruction> 'si la condition 2 est remplie Case is <condition3>, <condition4> <instruction> 'si l'une des conditions est remplie … Case Else <instruction> 'si aucune condition n'est remplie End Select

Exemple de structure de choix Branchement multiple SELECT CASE

- Calcul du montant de la ristourne (corps de l'algorithme et du programme)

Calcul du montant de la ristourne (corps de l'algorithme et du programme)

Algorithme	VBA
SELON CA CAS EST > 15000 RIST ⟵ CA*0,08 CAS EST 10000 A 15000 RIST ⟵ CA*0,06 CAS EST < 10000 RIST ⟵ CA*0,03 FIN SELON	Select Case CA Case is > 15000 RIST = CA*0,08 'si le CA est supérieur à 15 000 Case is >= 10000 RIST = CA*0,05 Case Else RIST = CA*0,03 End Select

Exemple de Branchement multiple SELECT CASE

- Entrées : prix HT (réel), catégorie de produit (chaîne)
- Sortie : prix TTC (réel)

```
'fonction select case
Public Function MonTTCSelon(pht As Double, cat As String) As Double
'déclarer la variable de calcul
    Dim pttc As Double
    'en fonction de la catégorie de roduit
    Select Case cat
        Case "luxe"
            pttc = pht * 1.33
        Case Else
            pttc = pht * 1.2 'toute autre valeur que ‘’luxe’’
    End Select
    'renvoyer le résultat
    MonTTCSelon = pttc
End Function
```

```
'fonction select case

Public Function MonTTCSelon(pht As Double, cat As String) As Double
   'déclarer la variable de calcul
   Dim pttc As Double
   'en fonction de la catégorie de roduit
      Select Case cat
    Case "luxe"
    pttc = pht * 1.33
    Case Else
    pttc = pht * 1.2 'toute autre valeur que ‘’luxe’’
    End Select
    'renvoyer le résultat  MonTTCSelon =
    pttc
End Function
```

Branchement multiple SELECT CASE – Plages de valeurs

- Il est possible d'introduire des plages de valeurs dans la partie Case de la structure Select Case. La comparaison devient plus sophistiquée.
- Variable est un numérique dans ce cas, entier ou même réel.

Syntaxe

op.de.comparaison : **<** , **>**, **=** etc…

```
Select Case variable
      Case Is op.de.comparaison valeur
            bloc d'instructions
      Case valeur de départ To valeur de fin
            bloc d'instructions
      Case Else
            bloc d'instructions
End Select
```

Branchement multiple SELECT CASE – Plages de valeurs – Un exemple

Entrée : quantité (entier)
Sortie : prix unitaire (réel)
Calcul :

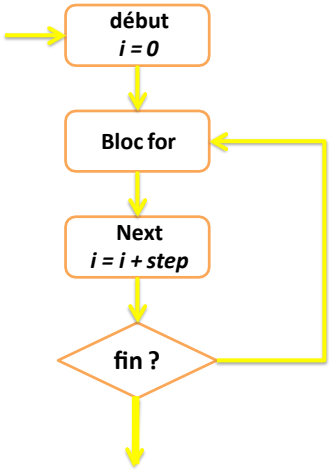
- quantité < 100 p.u. = 0.5
- 100 < quantité < 200 p.u. = 0.3
- quantité > 200 p.u. = 0.2

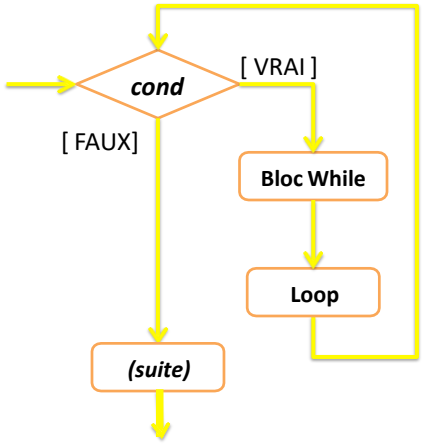
```
'calcul du prix unitaire en fonction de la
quantité
Public Function MonPU(quantite As Long) As Double
'variable intermédiaire
Dim pu As Double
'selon les valeurs de quantité
Select Case quantite
    Case Is < 100
        pu = 0.5
    Case 100 To 200
        pu = 0.3
    Case Is > 200 'Case Else aurait fait l'affaire
        aussi
        pu = 0.2
End Select
MonPU = pu
End Function
```

Les structures itératives

- Les structures itératives ou boucle répète l'exécution d'un traitement. Elle peut revêtir trois formes.
  - Lorsque le nombre de répétitions est connu:
    - **POUR… DE… A… FAIRE… FIN POUR**
      - La boucle s'exécute en fonction de la valeur d'une variable (compteur) qui est incrémentée d'un certain pas à chaque boucle. Elle s'achève quand la variable atteint une valeur déterminée et s'exprime ainsi
  - Lorsque le nombre de répétitions est a priori inconnu Algorithme. Il existe deux formes principales de boucle
    - **TANTQUE… FINTANTQUE**
    - **REPETER… JUSQU'A**

Les structures itératives



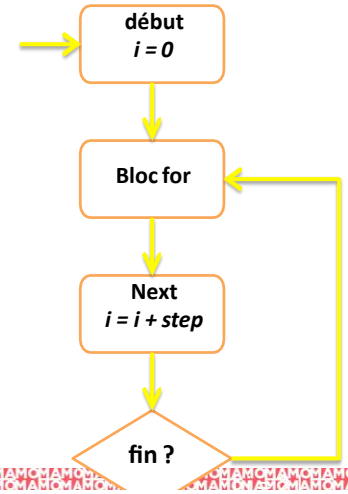


Les structures itératives la boucle POUR (FOR)

- Faire répéter l'exécution d'un bloc d'instructions. **Le nombre d'itérations est connu et contrôlé par un indice.**
- Syntaxe

```
For indice = val.départ to val.fin step pas
bloc d'instructions
...
Next indice
```

(1) Indice est un type ordonné, très souvent un numérique
(2) pas contrôle le passage d'une valeur à l'autre d'indice, si omis, pas = 1 par défaut
(3) Next entérine le passage à la valeur suivante de indice,
si cette prochaine valeur est > à val.fin, on sort de la boucle
(4) Val.fin doit être superieure à val.départ pour que l'on rentre dans la boucle
(5) Si pas est négatif, val.fin doit être inférieure à val.départ cette fois-ci
(6) L'instruction Exit For permet de sortir prématurément de la boucle
(7) On peut imbriquer des boucles (une boucle à l'intérieur d'une autre boucle)



Les structures itératives la boucle POUR (FOR)

- Algorithme et synthèse VBA associée

Algorithme	VBA
boucle POUR… DE… A… FAIRE… FIN POUR POUR variable DE valeurdébut A valeurfin PAS n Action(s) FIN POUR	**For… To… Step… Next** For <Valeur depart compteur> To <Valeur fin> Step <n : valeur incrément> <instructions> Next

Les structures itératives la boucle POUR (FOR) Exemple

- Les salariés de la société Alpha bénéficient d'une prime d'ancienneté complémentaire de 2 % (TXANC) de leur salaire brut au-delà de cinq ans de présence (ANC). Ils sont au nombre de 50 numérotés de 1 à 50 (NUMSAL).

Calcul du montant de l'ancienneté (corps de l'algorithme et du programme)	
Algorithme	**VBA**

POUR NUMSAL de 1 A 50 PAS 1
FAIRE
LIRE ANC
SI ANC > 5
ALORS TXANC ← 0,02
SINON RIEN instruction facultative
FIN SI
ECRIRE TXANC
FIN POUR
	For Numsal = 1 To 50 Step 1
	ANC = InputBox(« Quelle est l'ancienneté en années ?")
	If ANC > 5 Then
	TXANC ← 0,02
	End If
	MsgBox (« le montant du taux d'ancienneté est de " & TXANC & " € ")
	Next

Les structures itératives
Boucle TANT QUE… FAIRE (DO WHILE…LOOP)

- Boucles les plus courantes (algorithme)
- Le nombre d'itérations est à priori inconnu et contrôlé par une condition.

TANTQUE… FINTANTQUE	REPETER… JUSQU'A
Cette boucle permet de ne pas exécuter le traitement si la condition n'est pas initialement remplie. Elle suppose souvent une initialisation de la boucle et se présente ainsi : TANT QUE condition d'exécution des actions Action(s) FIN TANT QUE	Cette boucle impose d'exécuter les actions au moins une fois : REPETER Action(s) JUSQU'A condition d'arrêt

Les structures itératives
Boucle TANT QUE… FAIRE (DO WHILE…LOOP)

- Boucles les plus courantes (VBA)

While Wend (portée générale)	Do Loop (portée plus spécifique)	
Cette boucle exécute une série d'instruction tant qu'une condition est vérifiée	Cette boucle répète une série d'instructions tant qu'une condition est remplie ou jusqu'à ce qu'elle le soit. Elle suppose souvent une initialisation de la boucle et se présente ainsi :	
While <condition> <instructions> Wend	Do While <condition> <instructions> Loop Répète la boucle tant que la condition est remplie (vraie) ou jusqu'à ce que la condition ne soit plus remplie (faux)	Do Until <condition> <instructions> Loop Répète la boucle tant que la condition n'est pas remplie (faux) ou jusqu'à ce que la condition le soit (vrai)

Les structures itératives
Boucle TANT QUE… FAIRE (DO WHILE…LOOP)
Exemple

- Dans l'entreprise Alpha, le nombre de salariés n'est pas fixe, il évoluera dans le temps, Une question sera posée pour savoir si l'on désire un autre calcul. Une

Calcul du montant de l'ancienneté (corps de l'algorithme et du programme)

Algorithme	VBA

Lire NUMSAL
TANT QUE NUMSAL <> zzz
LIRE ANC
SI ANC > 5
ALORS TXANC ⟵ 0,02
FIN SI
ECRIRE TXANC
Lire NUMSAL
FIN TANT QUE
	NUMSAL = InputBox(« donner le numéro du salarié ou non pour quitter :")
	While client <> « non »
	ANC = InputBox(« Quelle est l'ancienneté en années ?")
	If ANC > 5 Then
	TXANC ⟵ 0,02
	End If
	MsgBox (« le montant du taux d'ancienneté est de " & TXANC & " € ")
	'saisir un nouveau client ou quitter
	NUMSAL = InputBox(« donner le numéro du salarié ou zzz pour quitter :")
	'recommence pour un nouveau client d'où Wend
	Wend

Les structures itératives Boucle FOR – Un exemple

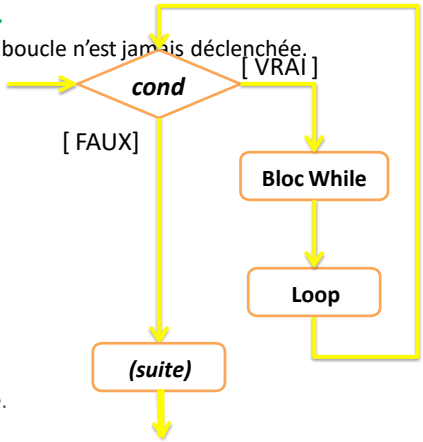
Entrée : n (entier)
Sortie : S (réel)
Calcul : $S = 1^2 + 2^2 + \ldots + n^2$

```
'calcul de la somme des carrés des valeurs
Public Function MaSommeCarre(n As Long) As Double
 'variables de calcul (s pour la somme, i : indice)
Dim s As Double, i As Long
 'initialisation
s = 0
'boucle avec l'indice i
For i = 1 To n Step 1
      s = s + i ^ 2
'Next joue le rôle de l'incrémentation (i suivant)
Next i
'renvoyer le résultat
MaSommeCarre = s
End Function
```

Les structures itératives
Boucle TANT QUE… FAIRE (DO WHILE…LOOP) et REPETER JUSQU'A

- Faire répéter l'exécution d'un bloc d'instructions.
- **Le nombre d'itérations est à priori inconnu et contrôlé par une condition.**
- Attention à la **boucle infinie** c.-à-d. la condition permettant de sortir de la boucle n'est jamais déclenchée.
- Syntaxe 1

```
Do While condition
Bloc d'instructions...
...
Loop
```

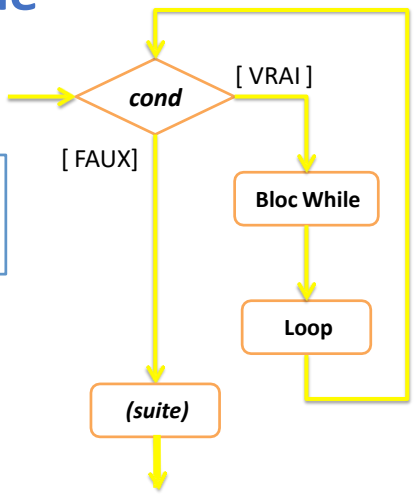


(1) Condition est un booléen, c'est souvent une opération de comparaison
(2) On continue l'exécution TANT QUE la condition est vraie ; si la condition est fausse, on sort de la boucle
(3) Exit Do permet de provoquer la sortie prématurée de la boucle
(4) Si la condition est fausse d'emblée. On peut ne pas rentrer dans la boucle.

VBA : Instructions de contrôle

- Boucles : **Do While… Loop**

```
Do While condition
    ' exécute tant que la condition est vraie
Loop
```




```
'tant qu'il reste des registres dans le  'RecordSet
Do While Not rsl.EOF
    tot = tot + rsl("Montant")  rsl.MoveNext
Loop
```

- **Entrée :** n (entier)
- **Sortie :** S (réel)
- **Calcul :** S = $1^2 + 2^2 + \ldots + n^2$

```
'calcul de la somme des carrés des valeurs
Public Function MaSommeCarreWhile(n As Long) As Double
'variables de calcul
Dim s As Double, i As Long  'initialisation
s = 0
'il nous revient aussi d'initialiser l'indice
i = 1
'boucle TANT QUE
Do While (i <= n)
    'sommer
    s = s + i ^ 2
    'pas de next, nous devons incrémenter l'indice
    i = i + 1
Loop
'renvoyer le résultat
MaSommeCarre
'While = s
End Function
```

Boucle FAIRE…TANT QUE (DO…LOOP WHILE)

- Faire répéter l'exécution d'un bloc d'instructions. Le nombre d'itérations est contrôlé par une condition.
- Syntaxe

```
Do
Bloc d'instructions
...
...
Loop While condition
```

On est sûr de rentrer au moins une fois dans la boucle.

Le choix de la bonne structure (Faire.. Tant Que ou Tant Que.. Faire) dépend du problème à traiter

Les variantes des boucles DO

- Les boucles DO contrôlées par une condition sont très riches en VBA.

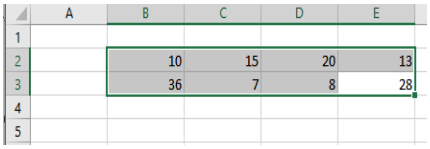
```
Do { While | Until } condition    [ statements ]
     [ Exit Do ]
     [ statements ]
Loop
-or-
Do
     [ statements ]   [ Exit Do ]
     [ statements ]
Loop { While | Until } condition
```

On peut aussi utiliser: Le Répéter… Jusqu'à (Until)

Exemples de Macros VBA

Sélection simple

- Comment programmer une macro qui manipule directement une plage de cellules sélectionnée par l'utilisateur ? Attention, nous ne sommes pas dans la même configuration que les fonctions personnalisées ici, nous n'insérons pas un résultat dans une cellule, nous manipulons et modifions directement la plage sélectionnée.



	A	B	C	D	E
1					
2		10	15	20	13
3		36	7	8	28
4					
5					

Résultat…

10	15	20	13
36	7	8	28

```
Sub MesValeursPaires()
'variable intermédiaire
Dim cellule As Range
'boucler sur la sélection
For Each cellule In Selection
    'tester le contenu
    If (cellule.Value Mod 2 = 0) Then
        'modifier la couleur de la police
        cellule.Font.ColorIndex = 4
    End If
Next cellule
End Sub
```

Selection est un objet Excel (Selection est donc un mot clé). Il est de type Range (que nous connaissons bien).

Exemple : dans cette sélection (les cellules doivent être sélectionnées avant de lancer la macro !), mettre en police verte les cellules contenant une valeur paire.

Sélection simple – On aurait pu écrire…

```
Sub MesValeursPairesBis()
'variables intermédiaires
Dim i As Long, j As Long
'boucler sur les lignes
For i = 1 To Selection.Rows.Count
    'boucler sur les colonnes
    For j = 1 To Selection.Columns.Count
        'tester le contenu
        If (Selection.Cells(i, j).Value Mod 2 = 0) Then
            'modifier la couleur de la police
            Selection.Cells(i, j).Font.ColorIndex = 4
        End If
    Next j
Next i
End Sub
```

Dans une sélection, de nouveau les coordonnées sont relatives c.-à-d. le coin en haut à gauche d'une sélection correspond à la cellule (ligne n°1, colonne n°1) quelle que soit la position de la sélection dans la feuille.

Aucun doute, Selection est bien de type Range

Sélection simple – Un second exemple

- Identifier la première cellule contenant la valeur minimale dans une plage, mettre sa police en bleu.

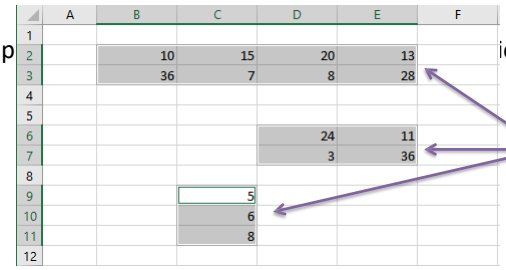
```
Sub MonMinBleu()
'variables intermédiaires
'min va servir de cellule témoin
Dim cellule As Range, min As Range
'initialisation du témoin sur la 1ère cellule
Set min = Selection.Cells(1, 1)
'parcourir
     For Each cellule In Selection
     'comparer avec le contenu de la cellule témoin
        If (cellule.Value < min.Value) Then
            'màj de la cellule témoin
            Set min = cellule
        End If
     Next cellule
'mettre la couleur pour la cellule minimale
min.Font.ColorIndex = 5
End Sub
```

10	15	20	13
36	7	8	28

Range est un **objet**. Une affectation pour une variable objet doit être réalisée à l'aide de l'instruction Set

Sélections multiples

- Une sélection p ieurs "zones"



Un exemple de sélection multiple avec 3 zones.

Très curieusement, le même mot clé Selection peut être exploité.

`Selection.Areas.Count` — Nombre de "zones" dans la sélection.

`Selection.Areas(k)` — Accès à la zone n°k (qui est de type Range). Areas est une collection de zones.

Sélection multiple – Un exemple

Pour chaque zone, mettre en police bleue la cellule contenant la valeur minimale.

```
Sub MonMinZoneBleu()
'var. intermédiaires
Dim zone As Range, min As Range
'pour chaque zone
For Each zone In Selection.Areas
    'à l'intérieur de chaque zone
    'initialisation
    Set min = zone.Cells(1, 1)
    'parcours des cellules
    For Each cellule In zone
        'comparer
        If (cellule.Value < min.Value) Then
            'màj de la variable témoin
            Set min = cellule
        End If
    Next cellule
    'mettre la couleur pour la cellule minimale
    min.Font.ColorIndex = 5
'passage à la zone suivante
Next zone
End Sub
```

Selection.Areas est une collection. On peut utiliser un For Each. On aurait pu aussi passer par un accès indicé. Par ex.

```
For k = 1 to Selection.Areas.Count
   Set zone = Selection.Areas(k)
   Etc…
```

Résultat…

	A	B	C	D	E	F
1						
2		10	15	20	13	
3		36	7	8	28	
4						
5						
6				24	11	
7				3	36	
8						
9			5			
10			6			
11			8			
12						

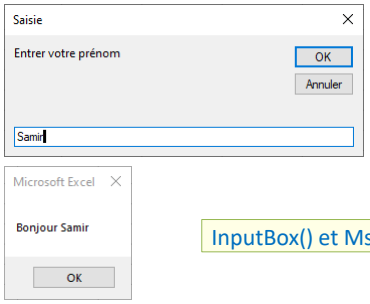
BOÎTES DE DIALOGUE

MsgBox et InputBox

Boîtes de dialogue standards

- Les boîtes de dialogue permettent d'interagir avec l'utilisateur. Nous nous en tenons aux plus simples ici. InputBox() pour la saisie, MsgBox() pour l'affichage.

```
Sub MesBoitesDeDialogue()
'var. intermédiaire
Dim prenom As String
'saisie
prenom = InputBox("Entrer votre prénom", "Saisie", "")
'affichage
MsgBox ("Bonjour " & prenom)
End Sub
```



InputBox() et Msgbox sont des fonctions

Noter la concaténation de chaînes de caractères pour faire apparaître le prénom dans la boîte de dialogue.

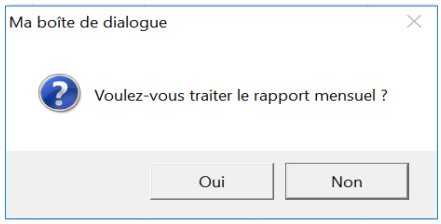
Ma MsgBox

- Boite de dialogue de base, "personnalisable"

vbOKOnly	0	N'affiche que le bouton ok
vbOKCancel	1	Ok et Annuler
vbAbortRetryIgnore	2	Abandonner, Recommencer, Ignorer
vbYesNoCancel	3	Oui, Non, Annuler
vbYEsNo	4	Oui, Non
vbRetryCancel	5	Recommencer, Annuler
vbCritical	16	Icône message critique
vbQuestion	32	Icône Question
vbExclamation	48	Icône exclamation
vbInformation	64	Icône Information
vbDefaultButton1	0	Le premier bouton est par défaut
vbDefaultButton2	256	Le 2ième bouton est par défaut
vbDefaultButton3	512	Le 3ième bouton est par défaut
vbDefaultButton4	768	Le 4ième bouton est par défaut
vbSystemModal	4096	Suspend tout jusqu'à une réponse de l'utilisateur

Ma MsgBox

```
Sub MaMsgBox()
Dim Config As Integer
Dim Reponse As Integer
Config = vbYesNo + vbQuestion + vbDefaultButton2
Reponse = MsgBox("Voulez-vous traiter le rapport mensuel ?", Config, "Ma boîte de dialogue")
If Reponse = vbYes Then
    MsgBox ("Youpi")
Else
    MsgBox ("Ohhhh")
End If
End Sub
```



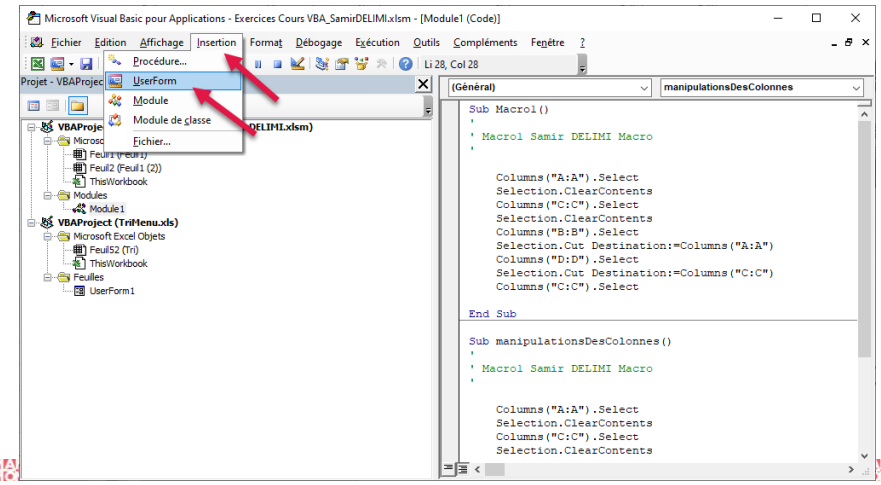
BOÎTES DE DIALOGUE (formulaires) Personalisées (Les UserForms)

BOÎTES DE DIALOGUE (formulaires) Personalisées (Les UserForms)

- Les UserForm (USF) servent à créer des boîtes de dialogue personnalisées.
- Vous pouvez y ajouter des contrôles afin de mettre en place une Interface utilisateur adaptée à votre projet.
- Il sera ainsi possible d'effectuer des saisies depuis ce support préformaté ou y visualiser des informations très diversifiées (Textes, données numériques, images, vidéos...)
- l'objet utilisé: **UserForm**.

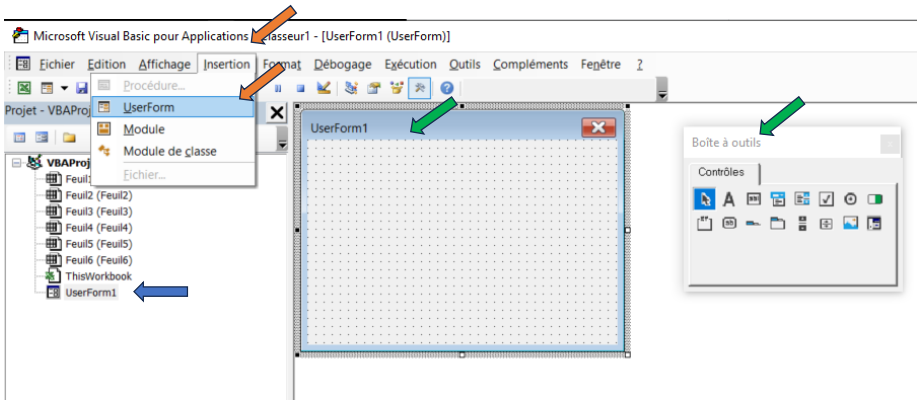
BOÎTES DE DIALOGUE (formulaires) Personalisées (Les UserForms)

- La création et la mise en forme de la boite de dialogue dans votre projet.



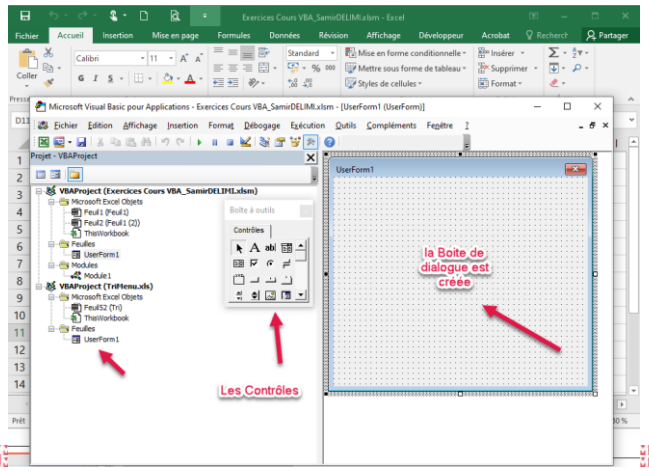
BOÎTES DE DIALOGUE (formulaires) Personalisées (Les UserForms)

- La création et la mise en forme de la boite de dialogue dans votre projet. *(ancienne interface de l'éditeur vba, rien ne change pour l'essentiel)*



BOÎTES DE DIALOGUE (formulaires) Personalisées (Les UserForms)

- La création et la mise en forme de la boite de dialogue dans votre projet.



BOÎTES DE DIALOGUE (formulaires) Personalisées (Les UserForms)

- La création et la mise en forme de la boite de dialogue dans votre projet.

La suite dans le TP

- Exemple de déclaration:
- https://indexmatch.fr/constantes-et-variables-vba

Sous-programme

Sous-programme: Présentation

- **Un sous-programme est un ensemble d'instructions appelées par un programme principal.**
- L'intérêt d'utiliser un sous-programme est double:
  - Le code devient plus lisible (et donc plus facilement maintenable), puisqu'il est divisé en plusieurs parties.
  - Les sous-programmes peuvent être réutilisés autant de fois que nécessaire pour un programme appelant.
- On distingue deux types de sous programmes :
  - **les procédures**
  - et **les fonctions**.

Sous-programmes: Paramètres

- Certains sous-programmes nécessitent des valeurs en entrée appelées paramètres.
- **Un paramètre est une valeur transmise par le programme principal à un sous-programme.**

- *Tout paramètre transmis à un sous-programme doit être déclaré dans l'en-tête de ce dernier.*
- *En cas de passage de paramètres à un sous-programme, l'ordre de transmission et le type de chaque paramètre doivent être rigoureusement identiques à ce qui a été défini lors de l'appel.*

Procédures

- **Une procédure est une suite d'instructions, dotée ou non de paramètres et appelée par un programme principal.**
-
-

VERSION Procédure sans paramètre avec variables *globales*

Programme en VBA

```
'Déclaration des variables globales
Dim Nombre as integer
Dim Carré as integer

Sub Principal()
'Déclaration des variables locales
'Traitement
        Nombre = Cells(4, 1).Value
        CalculCarre()
        AfficheCarre()
End Sub


Sub CalculCarre()
        Carré = Nombre * Nombre
End Sub

Sub AfficheCarre()
        Range("C4").value = Carré
End Sub
```

VERSION Procédure avec paramètres

Programme en VBA

```
Sub Principal()
'Déclaration des variables

Dim Nombre as integer
'Traitement

        Nombre = Cells(4, 1).Value
        Call DonneCarre (Nombre)
End Sub

Sub DonneCarre (Nb as integer)

        Range("C4").value = Nb * Nb
End Sub
```

Fonctions

- **Une fonction est une suite d'instructions, dotée ou non de paramètres, appelée par un programme principal et qui renvoie une valeur ou programme appelant.**
- On distingue les fonctions personnalisées (écrites par le développeur) et les fonctions prédéfinies (fournies par le tableur).

Les fonctions personnalisées

VERSION Fonction avec paramètres

Programme en VBA

```
Sub Principal()
'Déclaration des variables
Dim Nombre as integer
Dim Resultat as integer

'Traitement

     Nombre = Cells(4, 1).Value
     Resultat = CalculCarre(Nombre)
     Range("C4").value = Résultat
End Sub

FUNCTION CalculCarre(Nb as integer) as integer
     CalculCarre = Nb * Nb
End Sub
```

VERSION avec Procédures, fonctions et paramètres

Programme en VBA

```
Sub Principal()
'Déclaration des variables
Dim Nombre as integer
Dim Résultat as integer

'Traitement
     Nombre = Cells(4, 1).Value
     Resultat = CalculCarre(Nombre)
     AfficheCarre(Resultat)
End Sub


FUNCTION CalculCarre(Nb as integer) as integer
     CalculCarre = Nb * Nb
End Sub



Sub AfficheCarre (Res as integer)
     Range("C4").value = Res
End Sub
```

les fonctions prédéfinies par le tableur

- Le langage VBA propose un important nombre de fonctions prédéfinies (déjà écrites) pouvant être appelées dans un programme. Le tableau suivant mentionne quelques fonctions et n'est pas exhaustif.

Fonctions	Actions	Exemple
Fonctions	Actions	Exemples
=LEN(texte)	Renvoie le nombre de caractères du texte passé en paramètre	=LEN("DCG) → 3
=LEFT(texte; nombre)	Extrait du texte à partir de la gauche, le nombre de caractères étant précisé	=LEFT ("DCG",1) → D
=RIGHT(texte; nombre)	Extrait du texte à partir de la droite, le nombre de caractères étant précisé	=RIGHT("DCG", 1) → G
=UCASE(texte)	Renvoie le texte en majuscule	=UCASE("dcg") → DCG
=LCASE(texte)	Renvoie le texte en minuscule	=LCASE("DCG") → dcg
=MID(texte; position; nombre)	Extrait du texte, le nombre de caractères étant précisé à partir de la position précisée	=MID("DCG", 1;2) → DC
=REPLACE(texte1; position ; nombre; texte2)	Remplace dans texte1 à la position précisée, le nombre de caractères étant précisé par le texte2	=REPLACE("DCG";3;1;"A") → DCA

Utilisation des couleurs en VBA

Utilisation des couleurs en VBA

- En VBA (Visual Basic for Applications), les codes de couleurs sont généralement représentés par des nombres correspondant à une couleur spécifique.
- Ces codes de couleurs sont utilisés pour définir la couleur de remplissage, la couleur d'arrière-plan, la couleur de texte, la couleur des bordures, etc., pour les cellules, les formes ou d'autres éléments dans Excel ou d'autres applications Office via VBA.

  - Range("A1").Interior.Color = RGB(0, 0, 255)

  ou
  - Range("A1").Interior.Color = vbBlue

  ou
  - Range(“A1").Interior.Color = 65535

  ou
  - Range(“A1").Select
  - Selection.Interior.Color = 65535

- Voici quelques informations utiles sur les codes de couleurs en VBA :

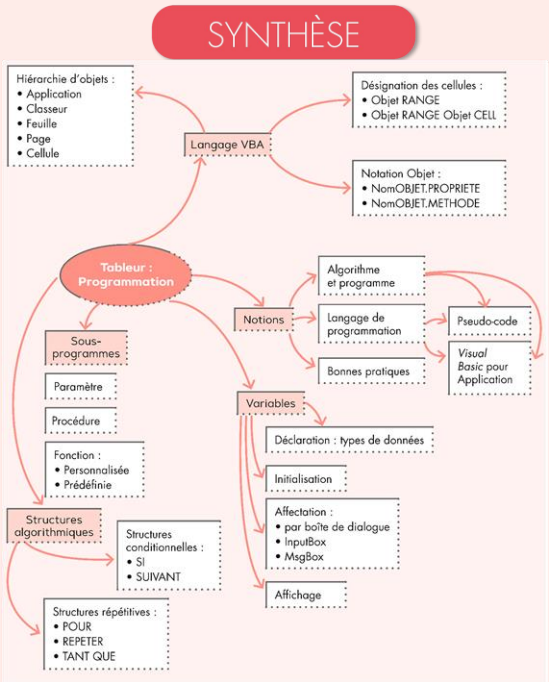
```
Sub test_expliquee()
' Début de la macro test
    Range("B3:H23").Select
    ' Sélectionne la plage de cellules de B3 à H23

    With Selection.Interior
    ' Commence un bloc de code avec la mise en forme intérieure de la sélection
        .Pattern = xlSolid
        ' Définit le motif de remplissage des cellules comme solide
        .PatternColorIndex = xlAutomatic
        ' Définit l'index de couleur du motif comme automatique
        .Color = 65535
        ' la propriété .color définit la couleur de remplissage des cellules comme jaune (65535 est le code pour le jaune)
        .TintAndShade = 0
        ' Définit la teinte et la nuance de couleur à 0 (pleine intensité)
        .PatternTintAndShade = 0
        ' Définit la teinte et la nuance du motif à 0 (pleine intensité)
    End With
    ' Termine le bloc de code pour la mise en forme intérieure

    Range("E16").Select
    ' Sélectionne la cellule E16
End Sub
' Fin de la macro
```

Utilisation des couleurs en VBA

- **Palette de couleurs standard :**
  - **RGB (Red, Green, Blue)** : Les couleurs peuvent être définies à l'aide des valeurs RGB, où chaque couleur primaire (rouge, vert, bleu) a une valeur comprise entre 0 et 255.
    - **Exemple** : RGB(255, 0, 0) représente le rouge pur, RGB(0, 255, 0) le vert pur, RGB(0, 0, 255) le bleu pur, et ainsi de suite.
      - Range("A1").Interior.Color = RGB(0, 0, 255)
  - **Index de couleurs prédéfinies** : Les couleurs peuvent également être définies par un index de couleur prédéfini dans VBA.
    - **Exemple**, vbRed, vbGreen, vbBlue, etc. Ces constantes ont des valeurs numériques spécifiques correspondant à différentes couleurs.
      - Range("A1").Interior.Color = vbRed
- **Codes de couleurs spécifiques :**
  - Certaines couleurs sont définies par des codes spécifiques en VBA.
    - Exemple : vbBlack: 0 , vbWhite: 16777215, vbRed: 255, vbGreen: 65280 , vbBlue: 16711680
      - Range("B3:H23").Interior.Color = 65535

Utilisation des couleurs en VBA

- la représentation des couleurs peut varier en fonction de l'application ou du contexte dans lequel elles sont utilisées.
- Les valeurs RGB sont largement utilisées pour spécifier des couleurs personnalisées, tandis que les constantes prédéfinies telles que vbRed, vbGreen, etc., offrent une manière pratique d'accéder à des couleurs spécifiques prédéfinies dans VBA.
- Choisissez la méthode qui convient le mieux à votre besoin spécifique et à la façon dont vous souhaitez définir les couleurs dans votre code VBA.



QCM en ligne:
https://app.wooclap.com/ATNDJU?from=event-page

ATNDJU

- https://app.wooclap.com/ATNDJU?from=instruction-slide

- Allez sur wooclap.com
- Entrez le code d'événement dans le bandeau supérieur
- Code d'événement
- ATNDJU

Cas de synthèse

Saisie Journalières TBC + VBA

Sur l'ENT

Sécurité des Macros

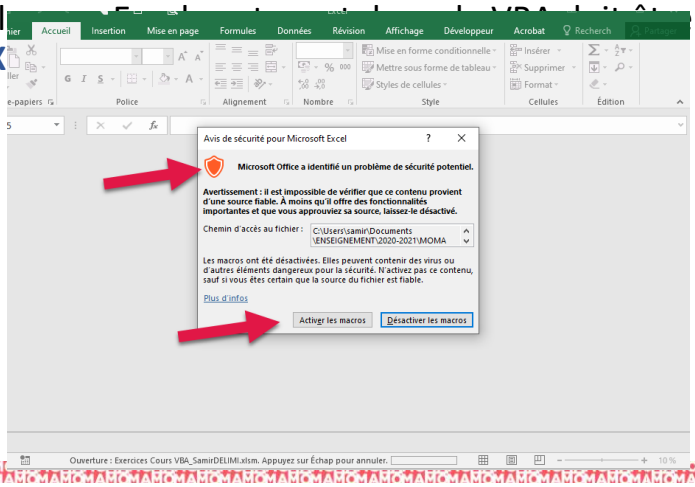
Sécurité des macros

- Excel offre une protection contre les virus qui peuvent être transmispar les macros.
- Si des macros sont partagées avec c'autres utilisateurs, on peut les certifier à l'aide d'une signature numérique. (voir cours sécurité des SI et/ou sécurité des échanges de données).
- Les autres utilisateurs sauront ainsi que ces macros proviennent d'une source sûre.
- A l'ouverture du classeur contenant des macros, leur source pourra être vérifiée avant de les activer.

Ouverture d'un Classeur contenant des Macros + Open office et VBA , autres

Ouverture d'un Classeur contenant des Macros

- La création et la mise en forme de la boite de dialogue dans votre projet. Un cl… enregistré au format X… perd son code.



VBA Excel – xlsx, xlsm, xlsb ?

- Depuis la sortie d'Excel 2007, les classeurs peuvent être stockés avec une extension xlsx, xlsm ou xlsb.
  - Les fichiers xlsx ne peuvent pas contenir du code VBA.
  - Les fichiers xlsm peuvent contenir du code VBA.
  - Les fichiers xlsb (classeurs Excel binaire) peuvent contenir ou ne pas contenir du code VBA. Contrairement aux deux autres formats de fichiers, ils sont compressés et prennent donc moins de place sur le disque.
- En ce qui concerne cette série d'articles, **vous sauvegarderez vos classeurs au format xlsm, sans quoi, le code VBA ne sera pas mémorisé.**

Compléments

Les couleurs en VBA

IA et EXCEL

Intelligence Artificielle et Excel

- L'intelligence artificielle (IA) et Excel peuvent être combinés pour améliorer les fonctionnalités d'Excel, automatiser des tâches complexes, effectuer des analyses avancées, et bien plus encore. Voici comment l'IA peut être intégrée à Excel :
- Automatisation des tâches : L'IA peut être utilisée pour automatiser des tâches répétitives dans Excel. Par exemple, des macros VBA peuvent être développées pour exécuter des opérations complexes basées sur des modèles d'apprentissage automatique (ML) ou des algorithmes d'IA.
- Analyse des données : Les capacités d'analyse de l'IA peuvent être utilisées pour explorer, nettoyer et analyser de grandes quantités de données dans Excel. Des algorithmes d'apprentissage automatique peuvent être appliqués pour prédire des tendances, effectuer des classifications, ou même identifier des schémas complexes dans les données.
- Visualisation des données : L'intégration de l'IA peut permettre la création de graphiques et de visualisations plus avancés et personnalisés dans Excel. Par exemple, des outils utilisant des réseaux neuronaux pourraient générer des représentations visuelles avancées des données.
- Prise de décision assistée : L'IA peut être utilisée pour créer des modèles de prédiction ou de recommandation dans Excel, aidant ainsi à la prise de décisions plus éclairées. Ces modèles pourraient être utilisés pour proposer des suggestions automatiques basées sur les données.
- Assistant de traitement du langage naturel (NLP) : Les fonctionnalités de traitement du langage naturel peuvent être intégrées à Excel à l'aide d'API ou d'outils dédiés. Cela peut permettre d'analyser des textes, de réaliser des résumés automatiques ou même d'extraire des informations pertinentes à partir de documents textuels.
- Extensions et compléments d'IA : Certains compléments tiers ou extensions pour Excel peuvent ajouter des fonctionnalités d'IA spécifiques, comme la reconnaissance d'image, la traduction automatique, ou d'autres capacités spécialisées.
- Il est important de noter que bien que l'IA puisse offrir de nombreuses opportunités pour enrichir les fonctionnalités d'Excel, cela nécessite souvent une expertise technique en matière de programmation et d'analyse de données. De plus, il est crucial de comprendre les données utilisées, de vérifier la qualité des résultats et de prendre en compte les implications éthiques lors de l'utilisation de l'IA dans des contextes professionnels.
- En résumé, l'intégration de l'IA dans Excel offre un potentiel considérable pour améliorer les capacités d'analyse, d'automatisation et de prise de décision au sein de l'outil de feuille de calcul le plus populaire.

Exemple d'utilisation de Python avec Excel

VBA avec Python pour IA EXCEL

- Excel ne supporte pas directement Python pour ses macros, il existe des méthodes pour utiliser Python en conjonction avec VBA dans Excel en utilisant le **module**
  - **Xlwings**
- xlwings permet d'exécuter du code Python à partir d'Excel en utilisant VBA.

Exemple: Comment utiliser VBA pour déclencher du code Python via xlwings dans Excel

- Voici un exemple simple qui montre comment utiliser VBA pour déclencher du code Python via xlwings dans Excel :

1. Tout d'abord, assurez-vous d'avoir installé xlwings via pip dans votre environnement Python :
   - *pip install xlwings*
2. Ensuite, ouvrez Excel et accédez à l'éditeur VBA (Alt + F11). Insérez un nouveau module et ajoutez le code VBA suivant :

```
Sub AppelerPython()
    Dim python As Object
    Set python = CreateObject("Python.Runtime.PythonWrapper")
     'Exécuter du code Python
    python.ExecScript "import xlwings as xw xw.Range('A1').value = 'Hello from Python!' »
    Set python = Nothing
End Sub
```

- Ce code VBA crée une instance de Python via CreateObject et exécute ensuite du code Python à l'aide de python.ExecScript. Cela permettra d'écrire "Hello from Python!" dans la cellule A1 de la feuille active dans Excel.

- Assurez-vous d'avoir un environnement Python configuré avec xlwings. Vous pouvez ensuite exécuter le code Python enregistré dans Excel. Cela écrira le texte spécifié dans la cellule A1.
- N'oubliez pas que cette méthode nécessite que Python soit installé sur votre système et que **xlwings** soit configuré correctement. Assurez-vous également de comprendre les implications de sécurité liées à l'exécution de scripts Python depuis Excel, en particulier s'il y a des macros dans des fichiers potentiellement non sécurisés.

- Cet exemple illustre comment utiliser VBA pour appeler du code Python via **xlwings** dans Excel, mais notez que cela reste un exemple simple. Des cas d'utilisation plus complexes peuvent être envisagés, mais nécessiteront une compréhension plus approfondie de **xlwings**, VBA et Python pour être réalisés avec succès.

Open Office Calc (OOCalc)

- Le tableur libre Open Office Calc (idem pour LibreOffice) propose les mêmes outils qu'Excel, notamment les outils d'analyse (Valeur cible, solveur, simulation, tableaux croisés dynamiques, etc.).
- A la différence qu'OOCalc est totalement gratuit.
- Il est possible de programmer en VBA sous OOCalc. Il suffit d'introduire préalablement l'instruction magique "Option VBASupport 1".
- Ces deux classeurs en témoignent (attention, réduire le niveau de sécurité pour pouvoir exécuter les macros).

Futur outils VBA

- ***Laissons la question à nos successeurs ou attendions d'être décideur.***
- tant d'applications ont été développées sur VBA par les sociétés qu'il n'est pas pensable que Microsoft prenne sa gomme pour effacer VBA.
- il faut prévoir le futur : VBA or not ? Est-il incorrect de penser que Microsoft poussera toujours plus la version online de sa suite bureautique et délaissera un peu plus la version offline (installation de MS Office sur les PCs des utilisateurs). Dans ce schéma, les jours de VBA sont compté puisque non supporté par le navigateur web.
- Je comprends bien qu'on parle ici en années et qu'il n'y a aucune urgence (contrairement au message alarmiste donné par le technicien du premier sujet de ce post).
- Ca fait plus de 40 ans que j'entends parler de la mort du Basic et que je le vois renaitre et s'adapter sous de nouvelles formes : TI-Basic, GWBasic, Quick Basic, Visual Basic, Phoenix Basic, VB.Net et bien d'autres.

Futur outils VBA

- Avant VBA, Excel utilisait des "macro" sous forme de suite de fonctions/commande Excel plus une poignée d'instructions pour les enchaîner.
  Je n'ai pas vérifié mais je pense qu'elles sont encore compatibles (puisque dans le VBA, des méthodes permettent de les utiliser). Je suis par contre certain qu'à la sortie de VBA, ces macros sont restées fonctionnelles plusieurs années (j'en avais quelques unes).

  Si VBA est abandonné pour être remplacé par un autre langage (en entend parler de JavaScript, Java, VBNet, Python, et autres), je doute que ça se fasse brutalement.
  Il est probable qu'il y aura une longue période de recouvrement pendant laquelle il restera utilisable (toujours dans le pack mais sans évolutions), comme ce fut le cas pour les "macro" précédentes.

Les compléments Visual Studio Tools pour Office (VSTO)

- *Microsoft a pris sa décision il y a plusieurs années sur le langage pour succéder à VBA, et ce sera JavaScript. Ils développent actuellement l'API qui sous-tendra le nouveau langage macro, et espèrent le rendre disponible sur tous les points de terminaison Office (bureau Windows et Mac, Office Online, iOS, Android, Windows phone).*

  Le code JavaScript vivra sur un serveur, il sera donc facile à mettre à jour. Il fonctionnera également de manière asynchrone (contrairement à VBA). Vous devez donc programmer en attendant que les valeurs se mettent à jour sur la feuille de calcul ou le document avant de continuer.

  Microsoft a choisi JavaScript parce que le langage est largement connu dans la communauté des programmeurs et continue d'être enseigné dans les universités. Les moteurs d'exécution JavaScript sont également disponibles universellement dans les points de terminaison tels que iOS et Android. Moteurs d'exécution pour d'autres langues - pas tellement.

  Le modèle objet pour Excel (l'application qu'on connais le mieux) est actuellement utilisable, mais incomplet. Microsoft vient de publier la version 1.4 et semble publier de nouvelles versions plusieurs fois par an.

  Tous les tableaux JavaScript sont basés sur zéro et une dimension. Si vous avez besoin d'un tableau à deux dimensions, vous devez utiliser l'équivalent le plus proche, qui est un tableau de tableaux à une dimension. JavaScript est très particulier sur la capitalisation, mais ne se soucie pas du tout du type de variable. Pour un programmeur VBA expérimenté, ce seront certainement des pierres d'achoppement lorsque vous apprendrez la langue.

  Il n'existe actuellement aucun enregistreur de macros JavaScript. Il n'y a donc pas de chemin facile vers l'automatisation pour les employés de bureau qui ont besoin d'un peu de productivité mais dont le travail quotidien n'a rien à voir avec le service informatique de l'entreprise. Étant donné que l'écrasante majorité des personnes qui écrivent du code VBA ont peu ou pas de formation en programmation formelle, je m'attends à ce que VBA continue de vivre pendant de nombreuses années à venir.

  Ne vous attendez pas à ce que les macros JavaScript soient des démons de la vitesse. Les benchmarks actuels sont beaucoup plus lents que VBA. Au lieu de cela, sachez que le véritable avantage des macros JavaScript est qu'elles fonctionneront sur des points de terminaison qui ne prendront jamais en charge VBA, comme iOS, Android ou Office Online.

  Si vous souhaitez en savoir plus sur la mise en œuvre de JavaScript, l'un des développeurs Mikhail Zlatkovsky a un ebook (Building Office Add-ins using Office.js) qu'il tient à jour à chaque nouvelle version.

Nouveau modèle des compléments Office.

- Vous souhaitez développer des solutions qui étendent l'expérience Office sur plusieurs plateformes ? Consultez le nouveau modèle des compléments **Office**. Les compléments Office ont un faible encombrement par rapport aux compléments et solutions VSTO, et vous pouvez les créer en utilisant presque toutes les technologies de programmation Web, telles que HTML5, JavaScript, CSS3 et XML.
- **Documentation pour compléments Office**
- Utilisez la plateforme des applications Office pour créer des solutions qui étendent les applications Office et interagissent avec du contenu dans des documents Office et dans les messages électroniques et les éléments de calendrier Outlook. Avec les add-ins Office, vous pouvez utiliser des technologies web familières telles que HTML, CSS et JavaScript pour créer des solutions qui peuvent s'exécuter dans Office sur le web, Windows, Mac et mobile.
- https://docs.microsoft.com/fr-fr/office/dev/add-ins/
- https://developer.microsoft.com/en-us/office/gallery/?filterBy=Samples,Excel

Concepts fondamentaux de programmation avec l'API JavaScript pour Excel

- https://docs.microsoft.com/fr-fr/office/dev/add-ins/excel/excel-add-ins-core-concepts
- **API Office.js pour Excel**

Microsoft lance Power Fx, un nouveau langage de programmation low-code open source Power FX l'avenir mars 2021

- https://programmation.developpez.com/actu/313065/Microsoft-lance-Power-Fx-un-nouveau-langage-de-programmation-low-code-open-source-base-sur-Excel/
- L'événement Ignite 2021 2 mars 2021
- Lindhorst a déclaré que Power Fx est basé sur Excel pour faciliter son utilisation et attirer davantage de "citoyens" développeurs. Microsoft estime en effet que des centaines de millions de personnes créent des feuilles de calcul avec Excel tous les jours. Ainsi, Power Fx va leur permettre d'exploiter les concepts d'Excel qu'elles connaissent déjà pour créer une application aussi facilement qu'elles créent une feuille de calcul. Power Fx est un langage déclaratif, tout comme Excel. L'utilisateur définit le comportement qu'il veut, mais c'est au système de déterminer et d'optimiser comment et quand l'accomplir.

.NET : il est l'heure pour les entreprises de passer à l'open source

- ***Le framework .NET reste l'une des principales plateformes de développement en entreprise. La sortie de .NET Core 3.0 invite les entreprises à adopter la version open source de la plateforme.***
- L'officialisation de la sortie de .NET Core 3.0 marque une étape essentielle dans la modernisation des développements d'entreprise et la production d'outils métiers notamment destinés aux postes Windows.
Lancé en 2000, soit il y a bientôt 20 ans, le .NET Framework s'est rapidement imposé comme l'une des plateformes de prédilection des entreprises aussi bien côté serveur (notamment avec ASP.NET) que côté client (applications Windows Forms et WPF).Beaucoup d'eau a depuis coulé sous les ponts et le .NET Framework a connu de nombreuses évolutions au fil des versions de Windows ainsi que plusieurs déclinaisons (pour Windows Mobile mais aussi cross-plateforme via Mono puis Xamarin).
- **L'ère open source**
- Dès 2014, Microsoft entame une véritable révolution de sa plateforme. L'idée consiste à totalement réécrire et repenser le framework avec, cette fois, un objectif affiché d'être à la fois cross-système et open source ! Il est en effet temps de réinventer la plateforme pour un monde où Windows n'est plus dominant, où le cloud est omniprésent, où la mobilité réinvente la notion d'applications et où les containers révolutionnent toute la chaîne de développement du développement à l'opérationnel. Il est surtout urgent de redynamiser la communauté, de la souder, de lui redonner le pouvoir.
- Dans un premier temps, Microsoft opte pour une focalisation des efforts autour du « backend ». Ainsi naissent ASP.NET Core (le framework Web) et .NET Core (le framework côté serveur) distribués en open source sous la très souple et permissive licence MIT et simultanément lancés à la fois sous Linux, macOS et Windows.
Depuis, Microsoft n'a eu de cesse de toujours enrichir cette version « Core » et surtout d'ouvrir encore davantage à l'open source les plus essentielles composantes de son univers .NET.
- En mai dernier, Microsoft a complété sa stratégie et averti les entreprises : **le .NET Framework n'a pas d'avenir. Le futur de .NET repose sur .NET Core !** Le message a le mérite d'être clair. La roadmap à venir l'est tout autant.
- **.NET Core 3.0 est disponible**
- L'introduction de « .NET Core 3.0 » cette semaine marque le « GO » de Microsoft aux entreprises : il est temps de migrer progressivement votre existant vers la version open source.
« .NET Core 3.0 » apporte de très nombreuses améliorations (dont une optimisation pour les containers, le support du langage F#, le support d'ARM) dont la plus essentielle demeure le support de Windows Forms et WPF (tous deux versés en open source) afin de permettre le développement d'applications « Desktop » (limitées à Windows) et non plus uniquement serveur.
Le principal frein à l'adoption de « .NET Core » par les entreprises historiquement très liées au « .NET Framework » saute donc avec cette édition « 3.0 » qui prépare le futur de la plateforme.
- **.NET 5, le futur de la plateforme**
- Car après « .NET Core 3.0 » (une évolution 3.1 est attendue en novembre), le futur du framework portera le nom de « .NET 5 » dès novembre 2020. Un renommage et une renumérotation plus logique qu'il n'y paraît.
En effet, le mot « Core » cherchait à mettre en avant l'idée que la version open source du « .NET Framework » n'en conserverait que l'essentiel. Mais depuis, de nombreuses librairies ont été ajoutées et tout un écosystème s'est mis en place. « Core » permettait aussi de bien le différencier de la version « Framework » intégrée à Windows. Cette dernière disparaissant du paysage, la première mention n'a plus de sens. Quant au numéro « 5.0 » qui oublie un « 4.0 », il a lui aussi du sens afin d'éviter toute confusion avec les versions « .NET Framework 4.x » présentes dans Windows 10 et Windows Server 2016/2019.
- **Il est l'heure de moderniser ses devs**
- Bref l'avenir de « .NET » est donc « .NET 5 », un framework complet de développement aussi bien côté « Serveur » que côté « Desktop », mobiles et IoT. Car avec .NET 5, l'idée est bien d'obtenir un framework 100% open source et 100% cross-plateforme compatible Windows, Linux, macOS, Android, iOS, ou encore Raspberry Pi… Quelles que soient les applications envisagées.
Les entreprises ont donc tout intérêt à mettre d'ores et déjà leurs développeurs à « .NET Core 3.0 » pour être fin prêtes à « .NET 5 ». D'autant que la conversion de l'existant ne sera pas toujours triviale même si l'entreprise continue de se focaliser sur la seule plateforme Windows. Notamment parce que certaines technologies « .NET » désuètes ne feront jamais le voyage vers l'open source : à commencer par WCF (Windows Communication Framework) et les Web Forms.
- De son côté, Microsoft encourage vivement toutes les entreprises lançant de nouveaux projets à ne les réaliser que sous « .NET Core 3.0 ». Pour les applications plus anciennes, l'opportunité d'un minimum de refactoring doit être sérieusement envisagée. Il est en effet bien loin le temps des applications client-serveur… .NET Core 3.0 et plus encore .NET 5 sont conçus pour un monde multiplateforme, containerisé et construit sur des microservices et du cloud.
- *Source:*
Announcing .NET Core 3.0

- https://www.tiobe.com/tiobe-index/
- https://superbasic-v2.azurewebsites.net/

Du Low-Code pour les mordus d'Excel Microsoft Ignite 2 mars 2021

- L'une des raisons d'être des solutions No-Code/Low-Code n'est pas de retirer le développement des mains des développeurs mais de permettre aux métiers de personnaliser ou ajuster eux-mêmes la logique métier de l'application crée par les développeurs. Et c'est justement toute la raison d'être du nouveau langage Power FX introduit par Microsoft à l'occasion de MS Ignite et surtout libéré en open source. Car le langage n'est pas si nouveau. Il est une formalisation de celui déjà utilisé au sein du canvas Power Apps. Sa particularité est d'être essentiellement basé sur des formules Excel. L'idée est de permettre aux développeurs de créer les applications avec leurs outils professionnels habituels tout en codant la logique métier en Power FX au sein d'un fichier texte attaché. Parce que la plupart des utilisateurs métiers avancés maîtrisent Excel et ses formules, ils n'auront aucune difficulté à la personnaliser ou la modifier.
- Microsoft espère non seulement insérer Power FX dans tous les modules de sa Power Platform (Power Bi, Power Apps, Power Automate, Power Virtual Agents, …) mais également voir les éditeurs et fournisseurs de services l'adopter largement maintenant que le langage est versé en open source.
- Source : https://www.itforbusiness.fr/ce-que-les-dsi-doivent-retenir-de-microsoft-ignite-2021-42670

Visual Basic : Chronique d'une mort annoncée…

- https://www.itforbusiness.fr/visual-basic-chronique-dune-mort-annoncee-34719
- ***Très répandu en entreprise, essentiel au succès de Windows dans les années 90 et 2000, le langage Visual Basic va être officiellement abandonné par Microsoft dans les mois à venir…***
- Cette annonce ne laissera probablement pas insensible la génération des cinquantenaires et quarantenaires qui ont, pour beaucoup, découvert la programmation à travers Visual Basic.
- Certes, VB date d'une autre époque, est excessivement verbeux et s'aligne très peu avec les pratiques courantes de développement et les attentes des jeunes générations de développeurs. Mais il demeurera à jamais l'un des langages de programmation les plus simples d'apprentissage. Ses défauts sont aussi ses qualités : des années après leur création les lignes de code écrites en VB restent d'une grande lisibilité même par ceux qui n'ont jamais appris le langage.

Quelles alternatives à Visual Basic ?

- N'enterrons pas trop vite le langage pour autant. Microsoft et les entreprises n'arriveront pas à s'en défaire d'un simple claquement de doigts ne serait-ce que parce qu'il reste le langage dans lequel sont écrites tant de macros qui accompagnent les tableaux Excel et les documents Word. D'ailleurs, Visual Basic demeure toujours dans le TOP 10 des langages les plus populaires de l'indice TIOBE.
  Par ailleurs les développeurs qui continuent de vouloir se créer des outils en Basic peuvent s'orienter vers d'autres solutions à vocation professionnelle et bénéficiant de communautés très actives.
  C'est notamment le cas de **Xojo** (autrefois connu sous le nom de RealBasic) qui permet de développer rapidement avec un IDE très convivial et un Basic très proche de VB des applications aussi bien pour Windows, macOS ou Linux. Xojo permet même de développer en basic des Web Apps, des apps mobiles ou des apps IoT (pour Raspberry Pi).
  PureBasic est un autre basic doté d'un IDE convivial et doté d'une approche cross-plateforme. Le langage est en effet disponible pour Windows, macOS et Linux. Il produit un code 32 bits ou 64 bits très rapide à l'exécution. Il offre même un support étendu de la 3D via le moteur ludique et open source OGRE.
  Anywhere Software a récemment rendu son « **B4X** » à la fois gratuit et open source. Né à l'ère de la mobilité, l'outil de développement permet de créer des apps en Basic pour Android et iOS mais aussi pour Windows, Mac, Linux, Arduino, Raspberry Pi, ESP8266 et autres…
  Enfin, et de façon assez ironique, rappelons que Microsoft propose également un autre Basic destiné à l'initiation des plus jeunes. **Small Basic** existe à la fois sous forme d'IDE Windows (distribué via Windows Store) et sous forme d'IDE en ligne accessible depuis n'importe quel navigateur Web.
- https://www.itforbusiness.fr/visual-basic-chronique-dune-mort-annoncee-34719

- https://superbasic-v2.azurewebsites.net/

Références

De la documentation à profusion **(inutile d'acheter des livres sur VBA)**

Site de cours de Microsoft

VBA sous Excel : https://msdn.microsoft.com/fr-fr/library/office/ee814737(v=office.14).aspx Structures de décision : https://msdn.microsoft.com/fr-fr/library/hh892482(v=vs.90).aspx Structures de boucle : https://msdn.microsoft.com/fr-fr/library/ezk76t25(v=vs.90).aspx

https://docs.microsoft.com/fr-fr/office/client-developer/excel/excel-home?redirectedfrom=MSDN

Autres cours et supports

Cours VBA Ricco Rakotomalala lyon2

Le Compagnon Info : http://www.lecompagnon.info/excel/ Excel Easy : http://www.excel-easy.com/

Cours VBA Gratuit : http://www.excel-pratique.com/fr/vba.php

Excel VBA for Complete Beginners : http://www.homeandlearn.org/index.html

Et d'autres très nombreux encore… faites chauffer les moteurs de recherche.

58

Aller plus loin

- https://laurent-ott.developpez.com/tutoriels/programmation-excel-vba-tome-1/#LXIV
- https://laurent-ott.developpez.com/tutoriels/programmation-excel-vba-tome-2/

- La syntaaxe de base:
- **Les syntaxes de base**
- https://mhubiche.developpez.com/vba/fiches/syntaxes/bases/

- Modules de Classes: POO VBA
- https://sinarf.developpez.com/access/vbaclass/

- http://boisgontierjacques.free.fr/pages_site/procedures.htm

- https://silkyroad.developpez.com/VBA/DebuterMacros/

- https://silkyroad.developpez.com/VBA/GestionErreurs/
- https://silkyroad.developpez.com/VBA/UserForm/

- VBA: Rédiger des macros plus rapides
  - https://www.lecfomasque.com/vba-rediger-des-macros-plus-rapides/
- http://cours.pise.info/algo/introduction.htm#Haut
- OLE: Macros pour mind manager: http://aries.serge.free.fr/content/MAPPING/MacroMM/index.html

The program which Integrates with SAP ERP and SAP BW

- Merci

Samir DELIMI

Samir.delimi@umontpellier.fr

CAS DE SYNTHÈSE

GÉNÉRATION ET VÉRIFICATION DE CODES CLIENT CHEZ FLEURCOM

30 min

COMPÉTENCES ATTENDUES :
- **Interpréter un programme répondant à un problème de gestion.**
- **Rédiger ou compléter le code d'une fonction ou d'une procédure.**
- **Corriger ou modifier un programme, afin de l'adapter à un nouveau problème de gestion.**
- **Utiliser une macro-commande.**
- **Enregistrer une macro-commande.**

Mise en situation

L'entreprise FleurCom, située près de Toulouse, est un fournisseur de fleurs comestibles qui vend en boutique et sur Internet. La dégustation de fleurs a été introduite assez récemment par les grands chefs, pour agrémenter et embellir leurs mets gastronomiques. Ainsi, fleurs de courgette, de ciboulette, de concombre, de capucine, de fenouil et bien d'autres sont des variétés qu'il est possible de rajouter dans des plats. Décidée à commercialiser ces compléments culinaires naturels en France, la société a développé un site Web marchand grâce auquel elle vend différents produits et gagne peu à peu des parts de marché.

Afin de développer une stratégie marketing par département, elle référence ses clients avec un code composé de trois parties : numéro du client, numéro de département du client et numéro de la remise accordée. Pour vérifier la validité de ces codes client, elle a ajouté à ce code une quatrième partie : un nombre *clé*, dont la valeur ne sert qu'à vérifier les trois autres parties de la référence client.

Une feuille de calcul, nommée *Codification* et contenant le détail d'une référence client, ainsi que l'ébauche d'un programme écrit en VBA, ont été créés, pour permettre de générer ce nombre *clé* et de vérifier une référence client. En tant que comptable, nouvellement embauché, chez FleurCom, votre mission consiste à compléter l'ébauche de ce programme. Vous avez à votre disposition une base documentaire.

Mission n° 1 : programme de calcul du nombre clé

La feuille de calcul (voir Document 1) fait apparaître trois éléments : le numéro de client en plage B5:E5, le numéro de département en plage F5:G5 et le numéro de remise en cellule H5. On souhaite que le programme associé à la feuille calcule la valeur du nombre *clé* lorsque l'utilisateur clique sur le bouton de commande *DonnerCle* (voir Document 3).

L'ébauche de programme a prévu trois fonctions (Creacode(), VerifValidite() et CalculCle()) (voir Document 2) et deux procédures (DonnerCle() et VerifRef()), décrits ci-après :

- `Function CreaCode() As String`
'Crée et renvoie une chaîne de caractères à partir du n° client, du n° dep, du n° remise
- `Function VerifValidite(Code As String) As Boolean`
'Vérifie la validité du n° client, du n° dep, du n° remise et renvoie VRAI si tout est valide, FAUX dans le cas contraire
- `Function CalculCle(Code As String) As Integer`
' Calcule et renvoie la clé correspondant au code (n° client + n° dep + n° remise) fourni en paramètre
- `Sub DonnerCle()`
'Affiche la clé correspondant à la référence client après avoir vérifié sa validité
- `Sub VerifRef()`
'Vérifie la validité du n° client, du n° dep, du n° remise, de la clé, puis affiche un message de conformité ou de non-conformité

1.1 À l'aide du document 4, compléter en VBA la procédure **DonnerCle()**, qui calcule et affiche en cellule I5 le nombre *clé*, sachant que cette procédure réutilise les trois fonctions précitées.

1.2 Compléter en VBA la fonction **CreaCode()**.

1.3 Compléter en VBA la fonction **VerifValidite()**.

1.4 Compléter en VBA la fonction **CalculCle()**.

Mission n° 2 : programme de vérification d'une référence client

On souhaite maintenant que le programme associé à la feuille vérifie une référence client complète à partir des quatre éléments lorsque l'utilisateur clique sur le bouton de commande *VerifRef*.

2.1 Compléter en VBA la procédure **VerifRef()**, qui vérifie la référence client en plage B5:I5 et affiche un message de conformité ou de non-conformité selon le résultat de la procédure (voir Document 5). Cette procédure réutilisera également les trois fonctions du programme.

BASE DOCUMENTAIRE

Document 1 – Feuille de calcul *Codification*

Référence CLIENT						
N° Client			N° Dep		N° Remise	Cle
9	7	4	6	0	2	17

DonnerCle VerifRef

Document 2 – Ébauche du programme VBA

```
Option Explicit
Function CreaCode() As String
'Crée et renvoie une chaine de caractères à partir du n° client, du n° dep, du n° remise

End Function
Sub DonnerCle()
'Affiche la clé correspondant à la référence client

End Sub
Sub VerifRef()
'Vérifie la validité du n° client, du n° dep, du n° remise, de la clé

End Sub
Function CalculCle(Code As String) As Integer
' Calcule la clé correspond au code (n° client + n° dep + n° remise) fourni en parametre

End Function
Function VerifValidite(Code As String) As Boolean
'Vérifie la validité du n° client, du n° dep, du n° remise

End Function
```

Document 3 – Composition d'une référence client

a) numéro du client (séquentiel de 0001 à 9999) ;
b) numéro du département de livraison sur 2 positions (de 01 à 99) ;
c) numéro de remise accordée (0 à 2) ;
d) nombre *clé* sur 2 positions est obtenu de la manière suivante :
- multiplication de chaque chiffre par son rang (le premier chiffre de gauche est de rang 1 et le dernier est de rang 7) ;
- addition des résultats obtenus ;
- le nombre *clé* est égal au modulo 23 du résultat précédent.

Document 4 – Aide sur certaines fonctions Excel à utiliser dans le programme

- La longueur d'une chaîne en VBA s'obtient par l'instruction :

```
Longueur = Len(chaine)
```

Exemple : Longueur = Len("DCG"). Ici, Longueur contiendra 3.

Le modulo d'un nombre est égal au quotient d'une division (par exemple, 10/2, le reste est 0, le modulo est 5). Le module s'obtient par l'instruction :

```
VariableModulo = Nombre_à_diviser Mod Nombre_Diviseur
```

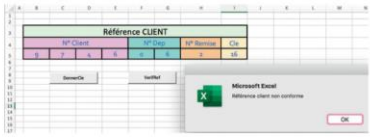
Exemple : VariableModulo = 10 Mod 2. Ici, VariableModulo contiendra 5.

- L'extraction d'un caractère d'une chaîne en VBA s'obtient par l'instruction :

```
Caractère = Mid(chaine, Position_Départ, Nb_Caractères_à_
            extraire)
```

Exemple : lettre = Mid("DCG",3 ; 1). Ici Lettre contiendra "G".

Document 5 – Test d'une référence client avec affichage d'un message de conformité ou de non-conformité



SUJET TYPE D'EXAMEN 3

Ce sujet vous met dans les conditions de l'examen.
Il porte sur les notions que vous avez acquises dans cette partie, essentielles à la compréhension et à la résolution des prochains sujets.

Le sujet se présente sous la forme de trois dossiers.
Dossier 1 – Étude des données (5 points)
Dossier 2 – Automatisation des calculs (10 points)
Dossier 3 – Analyse des données (5 points)

BASE DOCUMENTAIRE

Document 1 – Extrait du fichier Excel Ciqual
Document 2 – Exemple de menu fourni par les deux cuisiniers
Document 3 – Ébauche du menu automatisé sous le tableur
Document 4 – Fonction TROUVE()
Document 5 – Graphique du menu
Document 6 – Tableau affichant le nombre d'aliments par catégorie

Avertissement

Aucun document ni aucun matériel ne sont autorisés. En conséquence, tout usage d'une calculatrice est interdit et constituerait une fraude.

SUJET

Reprenons le cas du restaurant Healthy Food vu dans le sujet type d'examen 2.

DOSSIER 1 Étude des données

La première étape de votre mission est de récupérer les données Ciqual. Pour cela, vous vous connectez sur le site de l'Anses à l'adresse https://ciqual.anses.fr, sur lequel les données sont proposées en téléchargement au format Excel.