Véronique Plihon-Heiwy

VBA : l'essentiel

Cours et exemples de programmes pour *Visual Basic For Applications*



ellipses

Références sciences

VBA : l'essentiel

Cours et exemples de programmes pour *Visual Basic For Applications*

Véronique Plihon-Heiwy

ellipses

Collection Références sciences

dirigée par Paul de Laboulaye
paul.delaboulaye@editions-ellipses.fr

Retrouvez tous les livres de la collection et des extraits sur www.editions-ellipses.fr

ISBN 9782340-085602
©Ellipses Édition Marketing S.A., 2024
8/10 rue la Quintinie 75015 Paris

DANGER LE PHOTOCOPILLAGE TUE LE LIVRE

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5.2° et 3°a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit constituerait une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

www.editions-ellipses.fr

Avant-propos

Cet ouvrage est dédié au langage de programmation VBA (Visual Basic for Applications). Le langage VBA est un prolongement d'Excel, un tableur très utilisé par les gestionnaires mais aussi par les ingénieurs.

L'apprentissage et la compréhension du langage VBA nécessitent la connaissance préalable du tableur Excel. La programmation en VBA permet d'automatiser certaines commandes réalisées avec Excel (tri personnalisé, filtrage avancé, mise en forme de tableau, génération de graphique ou de PDF, etc.), de créer des procédures, des macros et des fonctions. VBA est un langage orienté-objet manipulant des objets appartenant à des classes. Comme langage de programmation, VBA offre notamment des instructions conditionnelles, répétitives, d'entrées-sorties, et d'affectation mais aussi de nombreuses procédures et fonctions prédéfinies pour manipuler des données de différents types (numérique, texte, etc.). Le langage VBA permet également de construire des boîtes de dialogue avec programmation événementielle associée. L'éditeur Visual Basic et le menu « Développeur » forment l'environnement de programmation VBA. Le travail du programmeur est facilité dans la création de ses programmes, par l'enregistrement de macro par exemple, mais aussi grâce à l'explorateur d'objets qui fournit un catalogue complet des classes prédéfinies de VBA ainsi que de leurs membres. La possibilité d'exécuter pas à pas ses programmes et de suivre l'évolution des valeurs des variables grâce à des « espions » fournit également une aide précieuse pour des programmeurs débutants, dans la mise au point de leurs programmes.

Cet ouvrage commence par une introduction qui présente les principales caractéristiques du langage de programmation VBA, son environnement de programmation et comment y avoir accès depuis le tableur Excel. Un exemple d'utilisation de l'enregistreur de macro est également présent dans ce premier chapitre. Le chapitre deux est consacré aux déclarations. Il présente les déclarations de variables, leur visibilité et les types de données prédéfinis de VBA qui précisent les valeurs possibles pour ces dernières. Ce chapitre aborde également les concepts qui font de VBA un langage orienté objet. Il se termine par la présentation des déclarations de procédures et de fonctions. Le chapitre trois est consacré aux instructions et blocs d'instructions contenus dans les programmes VBA, sans toutefois aborder les instructions d'entrées-sorties, assurant le dialogue entre l'utilisateur et le programme. Le chapitre quatre est dédié à la programmation événementielle, aux instructions d'entrées-sorties et à la création de boîtes de dialogue avec ou sans formulaire. Le cinquième chapitre propose une synthèse des

éléments de cours présentés dans les chapitres précédents et liste un ensemble de bonnes pratiques pour programmer en VBA. Cet ouvrage se termine par une annexe dédiée à des sujets d'entraînement corrigés qui montrent un florilège de cas d'utilisation du langage de programmation VBA. Les notions VBA abordées dans les sujets sont présentées en début d'annexe, sous forme de tableau croisé.

Cet ouvrage s'adresse aux étudiants qui ont VBA au programme de leur études, en Licence d'économie et de gestion, en Bachelor, en BTS, en école de commerce ou bien en cursus ingénieur pour accroître leur productivité individuelle. Il peut également être utile aux étudiants de Master, de mastère spécialisé ou de MBA. Il s'adresse aussi aux professionnels ayant besoin de se former en VBA. Il propose de présenter l'essentiel pour apprendre et comprendre VBA, sous forme de cours agrémenté de nombreux exemples et de nombreuses illustrations.

Pas besoin d'être expert en programmation pour prendre en main VBA, tout est fait pour le rendre le plus simple possible pour les futurs programmeurs. Cet ouvrage propose **un cours et des exemples de programmes pour Visual Basic for Applications**. Il décrit les éléments utiles à un apprentissage et une compréhension du langage, il n'a pas pour objectif de faire une présentation exhaustive de toutes les notions proposées par le langage VBA pour programmer mais plutôt de se focaliser sur **l'essentiel** pour débuter la programmation en VBA et devenir ensuite un utilisateur chevronné. Tous les exemples de programmes fournis sont abondamment commentés pour les rendre compréhensibles et reproductibles par le lecteur. Il ne me reste plus qu'à vous souhaiter, cher lecteur, d'apprécier la lecture de cet ouvrage.

Sommaire

Chapitre 1 - Introduction..... 9

Prérequis à l'apprentissage de VBA : connaître Excel..... 9

I. Importance de Visual Basic for Applications (VBA) 9

II. Modules..... 11

III. Langue utilisée 11

IV. Présentation de l'environnement VBA 12

IV.1 Prérequis : Disposer du menu « Développeur »..... 12

IV.2 Présentation du menu Développeur 16

IV.3 Présentation de l'interface de l'éditeur Visual Basic..... 16

IV.3.1 Les trois principales fenêtres de l'environnement VBA 18

IV.3.2 Les menus de l'éditeur Visual Basic..... 19

V. Enregistrement direct d'une macro..... 24

Chapitre 2 - Déclarations 31

I. Types prédéfinis..... 31

II. Les modules VBA..... 39

III. Déclarations de variables..... 39

III.1 Variable simple et visibilité 39

III.2 Déclaration de Tableaux 44

III.2.a Instruction *Option Base* 44

III.2.b Tableaux à plusieurs indices 44

III.2.c Accès à un élément d'un tableau..... 45

IV. VBA Langage Orienté Objet 46

IV.1 Accès aux propriétés et méthodes d'un objet 47
IV.2 Généralités sur l'orienté objet 47
IV.3 Accès aux propriétés d'un objet 48
IV.4 Principales classes d'objets prédéfinies dans Excel 49
IV.5 Variables prédéfinies simplifiant la navigation 55
IV.6 Exemples d'utilisation du conteneur Range et du conteneur Cells 69
V. Déclaration des procédures et fonctions 75
V.1 Procédures 75
V.2 Fonctions 77

Chapitre 3 - Instructions 89

I. Instructions d'affectation 89
Affectation d'une valeur 89
II. Contrôle de l'exécution 92
II.1 Instructions conditionnelles 92
II.1.a L'instruction If 92
II.1.b L'instruction Select Case 98
II.2 Instructions répétitives 105
II.2.a L'instruction While … Wend 105
II.2.b L'instruction Do … Loop 107
Première variante, sans condition 107
Seconde variante de l'instruction Do…Loop : condition While avant la « Séquence d'instructions répétée » 110
Troisième variante : Condition Until avant la « Séquence d'instructions répétée » 112
Quatrième variante : condition While après la « Séquence d'instructions répétée » 115
Cinquième variante : Condition Until après la « Séquence d'instructions répétée » 116
Résumé des variantes de l'instruction Do : 120

II.2.c L'instruction For ... Next 121
II.2.d L'instruction For Each ... Next 129
II.2.e Synthèse des instructions répétitives 135
II.3 Branchements 136
L'instruction GoTo 136
II.4 Instructions de sortie anticipée de programme 140
II.4.a L'instruction Exit Sub 140
II.4.b L'instruction Exit Function 140
II.5 Les déroutements 143
II.5.a L'instruction On Error GoTo 143
II.5.b L'instruction On Error Resume Next 143
II.5.c L'objet Err 144
II.5.d L'instruction Resume 145
III. Appels de procédure 147
IV. Appels de fonction 150
IV.1 Complément sur les appels de fonction 150
IV.2 Fonctions et opérateur manipulant les chaînes de caractères 152
IV.3 Fonctions d'affichage de nombres formatés 158
IV.4 Fonctions VBA utiles pour écrire des expressions booléennes : 161

Chapitre 4 - Procédures de gestion d'événements 169

I. Evénements associés au classeur 169
I.1 Workbook_Open 170
I.2 Workbook_BeforeClose 170
I.3 Workbook_SheetActivate 170
I.4 Workbook_SheetBeforeDoubleClick 170

II. Evénements associés à une boîte de dialogue ou à un contrôle..... 171

II.1 Les boîtes de dialogue..... 171

II.1.a Les boîtes de dialogues élémentaires..... 171

La procédure MsgBox..... 171

La fonction InputBox..... 176

La méthode de la classe Application InputBox..... 176

II.1.b Création d'une boîte de dialogue dans une feuille de calcul Excel 179

II.1.c Les formulaires et leurs contrôles..... 184

Chapitre 5 - Synthèse et bonnes pratiques..... 193

I. Synthèse – Unum Necessarium..... 193

I.1 Environnement de programmation VBA et processus de développement..... 194

I.2 Synthèse des déclarations..... 197

I.3 Synthèse des instructions..... 201

I.4 Synthèse de programmation événementielle et des boîtes de dialogue..... 205

II. Bonnes pratiques..... 208

Annexe..... 211

I. Sujets d'entraînement..... 212

I.1 Générer une courbe de Gauss..... 213

I.2 Génération d'un PDF..... 223

I.3 Loto..... 234

I.4 Les soldes du père Noël..... 242

I.5 Conversion de notes en compétences..... 253

I.6 Test psycho..... 262

I.7 PGCD..... 269

II. ABCD en 26 questions..... 276

Table des illustrations..... 281

Index..... 287

Chapitre 1 - Introduction

Prérequis à l'apprentissage de VBA : connaître Excel

Excel est un tableur permettant de créer des classeurs. Un classeur est composé de feuilles de calcul et de graphiques. Dans une feuille de calcul se trouvent des cellules. Chaque cellule est identifiée par le numéro de ligne et par le nom de colonne dans laquelle elle se trouve. A2 par exemple, désigne la cellule positionnée dans la colonne A et à la ligne 2. Les colonnes peuvent également être référencées par un numéro. A2 devient alors L2C1 (la cellule positionnée en deuxième ligne (L2) et en première colonne (C1)). Dans une cellule il est possible de saisir un texte, un nombre ou une formule.

Excel permet de créer des **tableaux de données**, de créer des **formules** dans des cellules et propose une grande variété de **fonctions** organisées en catégories (mathématiques, statistiques, logiques, de recherche, de base de données, de date, etc.). Excel propose également une grande variété de **commandes** pour trier des tableaux de données, filtrer des données, créer des mises en forme conditionnelles, créer des listes de choix, créer des tableaux croisés dynamiques, insérer des graphiques, mettre en forme des tableaux de données ou créer des formats personnalisés d'affichage pour les nombres notamment.

Malgré tout, Excel présente certaines **limites** : il peut arriver que les utilisateurs ne trouvent pas de fonctions d'Excel correspondant à leurs besoins ou qu'ils souhaitent automatiser l'exécution de commandes Excel. La solution pour dépasser ces limites réside dans le langage de programmation VBA.

I. Importance de Visual Basic for Applications (VBA)

Visual Basic for Applications (VBA) est le langage de programmation que Microsoft recommande d'utiliser pour les nouvelles applications depuis la sortie d'Excel 5, en 1993.

La stratégie de Microsoft est de se servir de ce langage VBA pour uniformiser l'automatisation de tous ses produits bureautiques, y compris le traitement de textes Word et le gestionnaire de bases de données Access.

Visual Basic pour Applications, en anglais Visual Basic for Applications, est donc un langage d'une grande importance dans la stratégie Microsoft. C'est lui que nous étudions ici. Nous abrégerons son nom en **VBA**.

VBA (Visual Basic for Applications) est un langage complet, moderne, qui n'a pratiquement plus rien à voir avec les premières versions du langage BASIC. Les actions réalisées par le code VBA s'effectuent à la demande de l'utilisateur, ce code est toujours écrit sous forme de sous-programmes.

- Les sous-programmes de type **procédure** (Subroutine ou **Sub**) servent à réaliser des actions. Le cas le plus simple est celui d'une procédure ***sans paramètre***, qui peut être **appelée par un clic sur un bouton** (c'est ainsi que se programme en VBA l'équivalent des anciennes macros). Les procédures ***avec paramètres*** ne peuvent être appelées que depuis une autre procédure ou une fonction.
- Les sous-programmes de type **fonction** (**Function**) ont pour but de calculer une valeur et de la renvoyer à l'appelant. Cet appelant peut être une autre procédure ou fonction, ou une formule d'un tableau Excel. C'est ainsi qu'il est possible de créer de nouvelles fonctions de feuille de calcul. L'appel d'une fonction peut également être utilisé comme terme dans une expression.

Exemple :

z = (max(x, y)+3)/6

L'appel de fonction *max(x, y)* est inclus dans une expression où sa valeur est augmentée de *3*, puis divisée par *6* avant d'être affectée à *z*.

VBA (Visual Basic for Applications) possède une structure de blocs pour les instructions conditionnelles :

If... End If

Select Case... End Select

et pour les instructions répétitives (boucles) :

Do... Loop

For... Next

For Each... Next

While... Wend

VBA ne crée des blocs que dans les cas d'instructions conditionnelles et des instructions répétitives (boucles) vues plus haut.

VBA (Visual Basic for Applications) est un langage de programmation orienté objet. Les objets y ont trois utilités :

- fournir une **interface** entre le programme VBA et le contenu des feuilles de calcul Excel : par exemple les objets de classe Range, qui représentent des ensembles de cellules (il est possible de traduire Range par étendue, zone, ou domaine) ;
- réaliser des **interfaces graphiques** permettant au programme VBA de communiquer avec l'utilisateur (objets **boutons**, objets **boîtes de dialogue**, objets **zones de texte**) ;
- permettre au programme VBA de **communiquer avec d'autres applications** en appelant des objets d'autres applications (Word ou Access) à partir du VBA d'Excel. L'utilisation d'Automation (anciennement OLE automation) ou d'échange dynamique de données (DDE) est nécessaire.

VBA (Visual Basic for Applications) permet au programmeur de définir et d'utiliser des **procédures** et des **fonctions**, avec passage **d'arguments** soit **positionnel** (les arguments sont identifiés par leur **position** dans une liste), soit **par nom**. Par ailleurs, ces paramètres peuvent être passés **par valeur** (une copie est passée) ou **par référence** (leur adresse est passée). Les procédures et fonctions sont le moyen naturel d'étendre le langage pour l'adapter aux besoins.

II. Modules

Microsoft a doté les tableurs d'une véritable structure de langage informatique, où les programmes sont séparés des données (ou presque, car les cellules peuvent toujours contenir des appels de fonctions).
Pour séparer les programmes des données, les **procédures et fonctions VBA** (**Visual Basic for Applications**) depuis Excel 97 se situent dans des structures particulières nommées **modules**.

Il est également possible d'associer des procédures à des **événements** (par exemple « *clic* ») liés à des **objets** (par exemple « *bouton de commande* ») présents sur des feuilles de calcul ou dans des boîtes de dialogue.

III. Langue utilisée

L'aide en ligne, quand elle est disponible, n'est plus valable que pour des instructions et des noms d'objets exprimés en **anglais**.

Cette restriction ne vaut que pour les mots-clés du langage, les **noms** définis par le programmeur (**types**, **variables**, **procédures** et **fonctions**) peuvent être des chaînes de caractères quelconques ou presque.

Depuis Excel 97, **l'anglais** s'est imposé comme langue de programmation pour VBA.

IV. Présentation de l'environnement VBA

Cette section est consacrée à la présentation de l'environnement VBA, tant pour les versions d'Excel pour PC que pour Mac.

Prendre connaissance de cet environnement est nécessaire pour avoir les clés pour créer, sauvegarder, mettre au point et exécuter ses propres programmes VBA.

IV.1 Prérequis : Disposer du menu « Développeur »

Pour programmer avec le langage VBA, il faut disposer d'un onglet spécifique dénommé « Développeur » dans le ruban.

Pour faire apparaître le menu « Développeur » :

Sur PC : Fichier/Options/Personnaliser le ruban.

Dans les onglets principaux, Cocher « Développeur ».

Puis valider par OK.

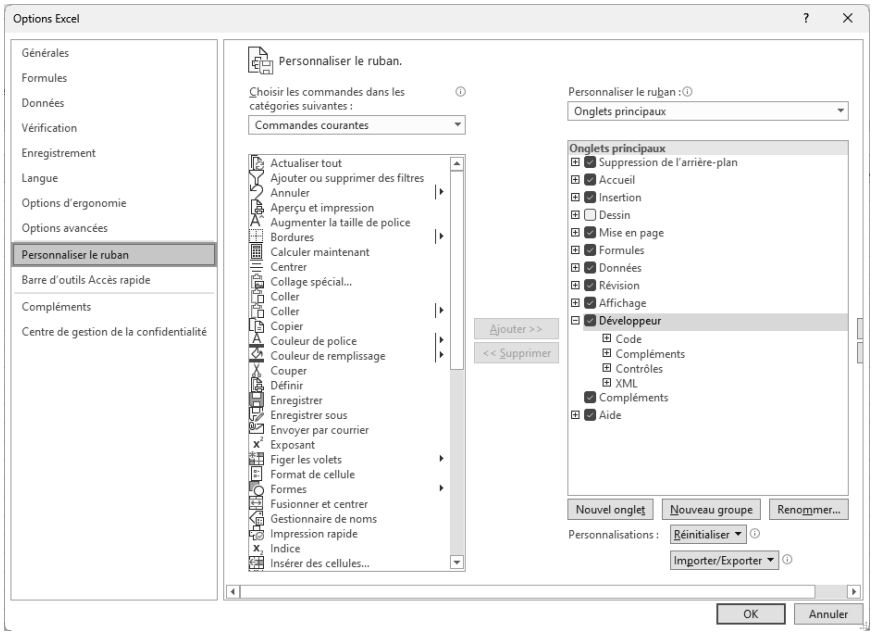


Figure 1 : Personnaliser le ruban (sur PC) pour faire apparaître le menu Développeur

L'onglet « Développeur » apparaît maintenant dans votre barre de menus ou ruban.

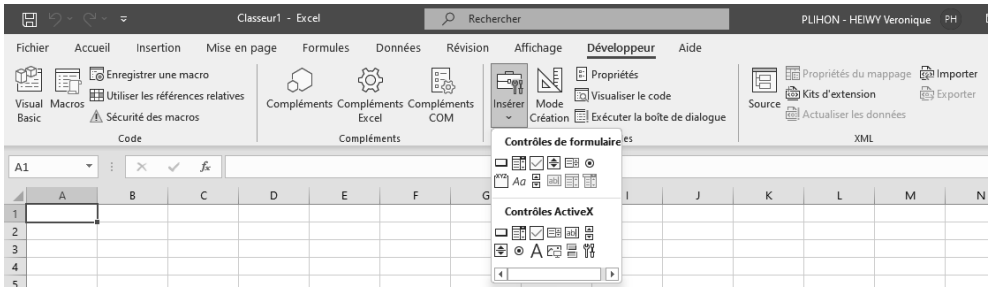


Figure 2 : Le menu Développeur d'Excel pour PC

Sur Mac : Excel/ Préférences/ Ruban et barre d'outils

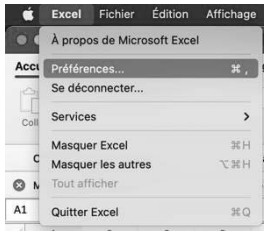


Figure 3 : Excel /Préférences...

Dans les « Préférences d'Excel », choisir, dans le groupe « Création » la troisième icône en partant de la gauche dénommée « Ruban et barre d'outils » (voir Figure 4). Un clic sur cette icône fait apparaître la boîte de dialogue de la Figure 5.

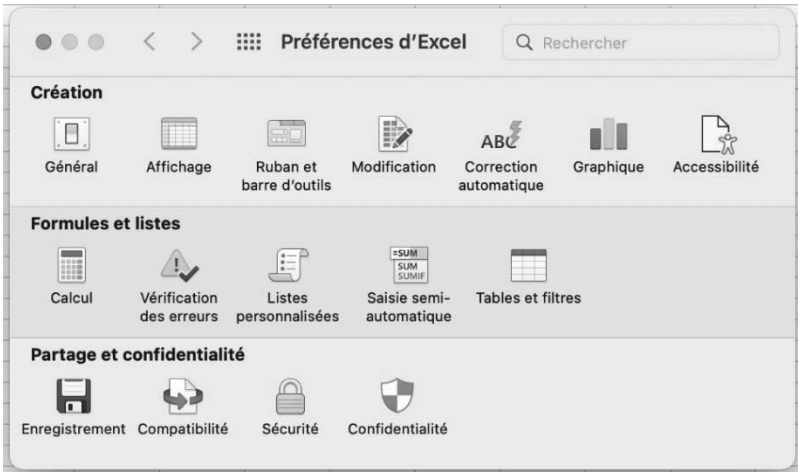


Figure 4 : Préférences d'Excel pour « Ruban et barre d'outils »

Dans l'onglet « Ruban », pour « Personnaliser le ruban » (à droite), choisir « Onglets principaux » et cocher la case « Développeur ».

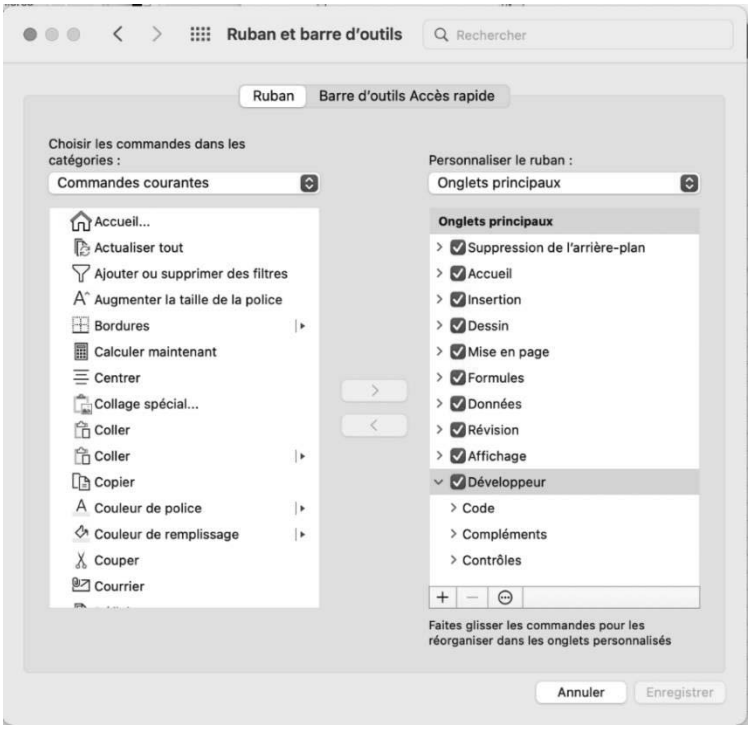


Figure 5 : Personnaliser le ruban (sur Mac) pour faire apparaître le menu Développeur

L'onglet « Développeur » apparaît maintenant dans votre barre de menus ou ruban.

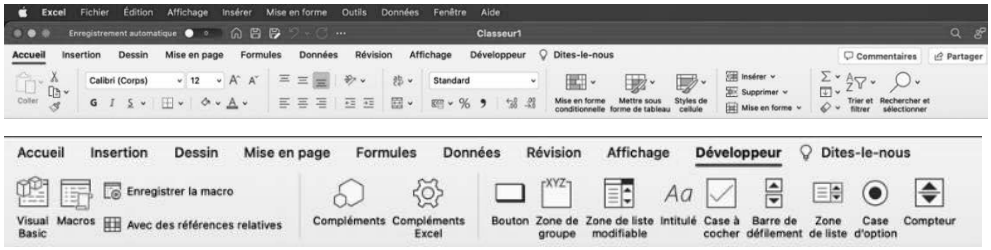


Figure 6 : Le menu Développeur d'Excel pour Mac

IV.2 Présentation du menu Développeur

Dans Excel pour PC comme dans Excel pour Mac, le menu « Développeur » comporte les icônes :

1. **Visual Basic** : pour accéder à l'éditeur Visual Basic

2. **Macros** : pour afficher la liste des macros qu'il est possible d'exécuter en choisissant celle qui convient dans la liste

3. **Enregistrer une macro** : pour démarrer l'enregistrement de macro. Une fois l'enregistrement démarré, le bouton permet d'arrêter l'enregistrement.

Dans la version Excel pour PC,

4. **Insérer** : pour afficher la **boîte à outils** contenant les contrôles graphiques de formulaires qu'il est possible d'insérer dans une feuille de calcul pour créer rapidement une interface graphique.

Dans la version Excel pour Mac,

la **liste des contrôles graphiques** (équivalente aux contrôles de formulaire de la boîte à outils de la version Excel pour PC) qu'il est possible d'insérer dans une feuille de calcul apparaît directement.

IV.3 Présentation de l'interface de l'éditeur Visual Basic

L'éditeur Visual Basic est accessible par un clic sur le bouton « Visual Basic » du menu « Développeur ». Il permet de créer du code VBA, de sauvegarder le projet Excel/VBA au format **.xlsm** (classeur Excel avec macros), d'afficher du code, de compiler le code, de mettre au point le code en l'exécutant pas à pas et d'exécuter le code sans mode pas à pas, directement, autant de fois que nécessaire.

La figure ci-dessous présente les différentes étapes nécessaires à la réalisation d'un programme en langage VBA.

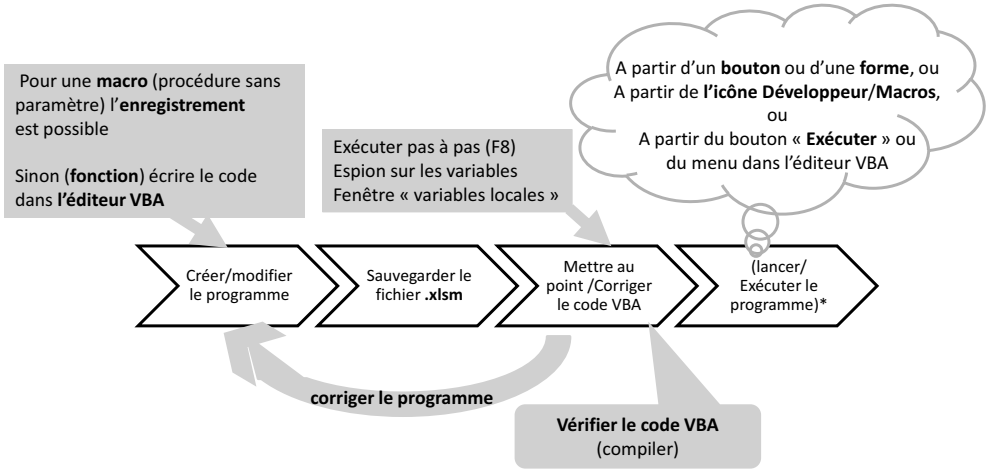


Figure 7 : Processus de développement

Le processus propose de commencer par (1) la **création du programme** qui peut être réalisée par enregistrement de macro ou par saisie du code VBA, ou par combinaison des deux approches.

Ensuite il faut (2) **sauvegarder le fichier Excel** en choisissant le type « Classeur Excel (prenant en charge les macros) ». Le fichier ainsi sauvegardé apparaîtra avec l'extension **.xlsm**.

Puis il faut (3) **mettre au point le programme** en le compilant et en l'exécutant pas à pas si besoin. Si des erreurs apparaissent soit dans la syntaxe soit dans la logique d'exécution du programme, il faudra **corriger le programme** et le sauvegarder à nouveau.

Une fois le programme au point, il sera possible (4) **d'exécuter** le programme autant de fois que souhaité, comme le symbolise l'étoile (*).

Nous remarquons une boucle d'actions qui permet d'aboutir à un programme conforme au besoin exprimé et respectant la syntaxe du langage VBA. Cette boucle permet d'insister sur la nécessité de sauvegarder le programme à nouveau lorsqu'il a été modifié.

IV.3.1 Les trois principales fenêtres de l'environnement VBA

L'éditeur Visual Basic est très proche dans la version Excel pour PC et dans la version d'Excel pour Mac.

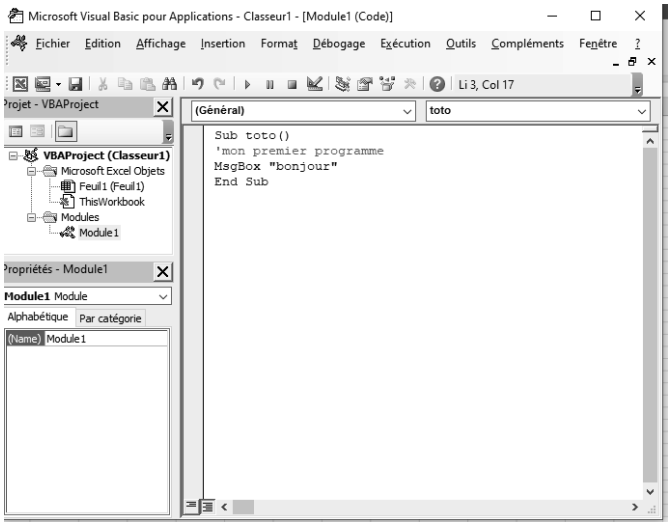


Figure 8 : L'éditeur Visual Basic d'Excel pour PC

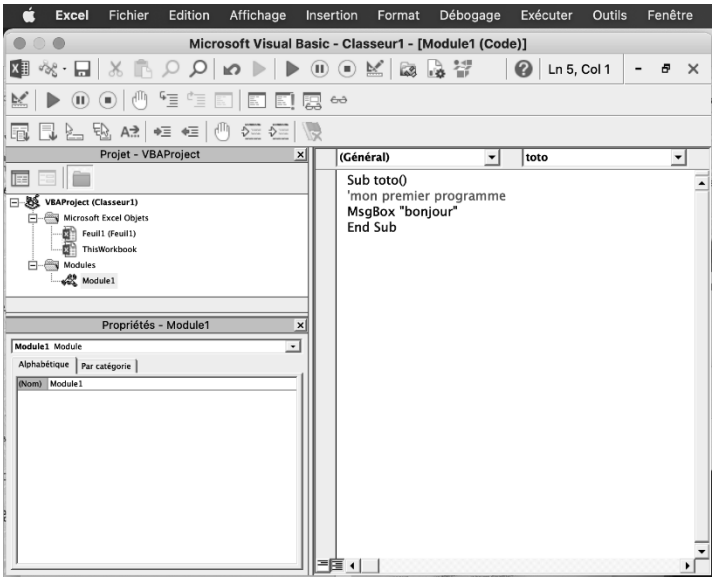


Figure 9 : L'éditeur Visual Basic d'Excel pour Mac

Voici les trois principales fenêtres de l'éditeur Visual Basic :

La **fenêtre « Explorateur de projets »**, affichée traditionnellement en haut à gauche de l'écran affiche la liste des feuilles de calcul contenues dans le classeur, une référence au classeur lui-même (ThisWorkbook) ainsi que les modules qui seront créés pour contenir les macros, procédures et fonctions VBA du classeur avec macros (correspondant au type de fichier avec l'extension .xlsm) et les formulaires utilisés pour créer des boîtes de dialogue (dans les versions d'Excel pour PC uniquement).

La **fenêtre des « Propriétés »** affichée traditionnellement en bas à gauche de l'écran. Elle affiche les propriétés d'un élément sélectionné. Si un module du projet VBA est sélectionné, son nom apparaît dans la fenêtre des propriétés et peut y être modifié. Attention, les espaces ne sont pas autorisés dans les noms des modules.

La **fenêtre du module ou « d'édition du code »** est affichée traditionnellement à droite. Pour le faire apparaître, il faut demander l'insertion d'un module (**Insertion/Module**). Le module (par défaut nommé « Module1 ») contiendra le code VBA des macros, procédures et fonctions créées.

IV.3.2 Les menus de l'éditeur Visual Basic

Voyons les principaux menus de l'éditeur Visual Basic d'Excel pour PC :

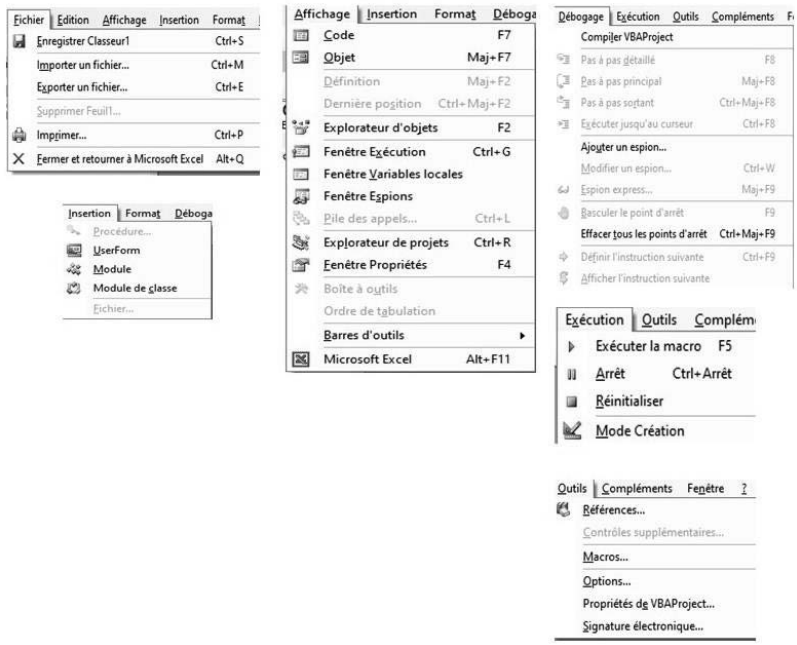


Figure 10 : Les principaux menus de l'éditeur VBA d'Excel pour PC

Voyons maintenant les menus de l'éditeur Visual Basic dans la version d'Excel pour Mac. Ils sont très similaires à ceux de la version Excel pour PC.

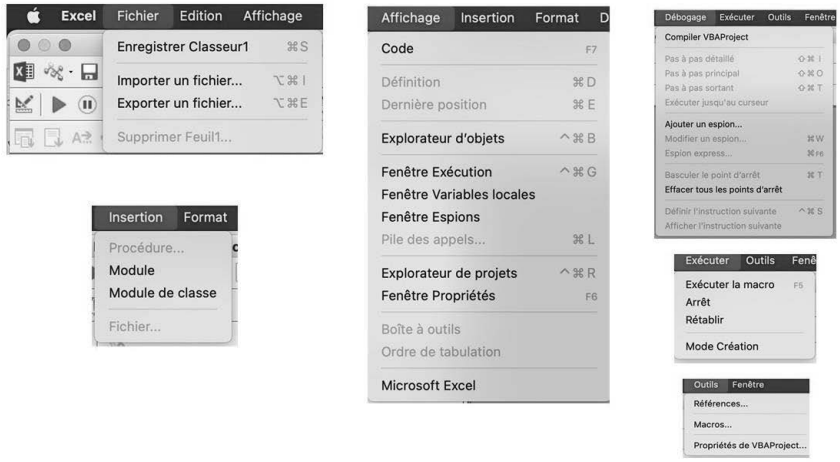


Figure 11 : Les principaux menus de l'éditeur VBA d'Excel pour Mac

Parmi les principaux menus, nous pouvons citer :

- Le menu **Fichier** qui permet notamment de sauvegarder le fichier Excel au format .xlsm, (classeur Excel prenant en charge les macros).
- Le menu **Edition** qui permet de copier/ coller et d'indenter le code VBA.
- Le menu **Affichage** qui permet d'afficher ou de désafficher les fenêtres présentant «l'explorateur de projets », « les propriétés », « les modules », « les variables locales », « les espions », ou encore « l'explorateur d'objets».
- Le menu **Insertion** qui permet d'insérer un « module » ou un « module de classe » ou (uniquement pour Excel dans sa version pour PC) un « formulaire ».
- Le menu **Format** qui permet notamment de gérer l'alignement.
- Le menu **Débogage** qui permet d'aider à corriger le programme et à le mettre au point en le compilant, ou en l'exécutant pas à pas, et/ou en ajoutant des espions ou des points d'arrêt.

- Le menu **Exécuter** qui permet d'exécuter les macros autant de fois que souhaité.
- Le menu **Outils** qui permet notamment d'accéder à la liste des macros créées ou de définir des propriétés pour le projet VBA ou (uniquement pour les PC), de fixer la taille d'affichage de la police pour le code VBA.

Voyons maintenant comment l'éditeur Visual Basic ou le menu Développeur permettent de mener à bien les différentes activités du processus de développement décrit à la Figure 7.

Comment créer du code VBA ?

Créer du code VBA peut se faire soit par enregistrement de macro (**Développeur / Enregistrer une/la macro**), soit en saisissant le code directement dans l'éditeur Visual Basic (**Développeur/Visual Basic**), dans un module (**Insertion/Module**), soit en mixant les deux approches (commencer par enregistrer une macro puis la compléter par des instructions saisies dans le module).

L'enregistrement de macro (**Développeur / Enregistrer une/la macro**) ne permet de créer qu'une macro, c'est-à-dire une procédure sans paramètre, (Sub ... End Sub) dont les instructions s'exécuteront séquentiellement (les unes à la suite des autres). Dès que nous voulons insérer des conditions ou des répétitions dans le déroulement des instructions, l'enregistreur de macro n'est plus opérant. Il en est de même pour la création d'une fonction (Function ... End Function), elle ne pourra pas être réalisée par enregistrement de macro.

Dans tous les cas, pensez à ajouter des **commentaires** dans vos programmes. Un commentaire commence par **'** (une « simple quote » ou apostrophe droite) ou par **REM**. Le commentaire n'est pas exécuté par le programme. Le commentaire peut être mis en début de ligne ou en fin de ligne à la suite d'une instruction. Dans l'éditeur Visual Basic, les commentaires apparaissent en vert, les instructions en noir, les mots-clés en bleu et les erreurs de syntaxe en rouge.

Dans cet ouvrage, les commentaires seront très largement utilisés afin de rendre les programmes les plus compréhensibles possibles pour le lecteur.

Comment mettre au point un programme VBA ?

Différentes possibilités sont offertes aux programmeurs dans l'éditeur Visual Basic pour aider à mettre au point son code. **L'exécution pas à pas** du programme (**F8** sur PC ou **MAJUSCULE + COMMANDE + I** sur Mac) ainsi que la possibilité de placer des « espions » sur les variables du programme afin de suivre leur évolution

dans la fenêtre des « variables locales » lors de l'exécution pas à pas du programme fait partie des aides fournies dans le menu « Débogage ».

Comment exécuter une macro ?

Plusieurs solutions sont possibles :

- Soit à partir de l'éditeur Visual Basic (bouton « Exécuter » ou menu « Exécution » sur PC ou « Exécuter » sur Mac voir Figure 8 et Figure 9). Cette solution n'est pas très intéressante si l'utilisateur du programme VBA n'est pas un spécialiste de VBA.
- Soit en recherchant la macro dans la liste des macros créées (**Développeur / Macros**).
- Soit en insérant un bouton de commande dans une feuille de calcul Excel (**Développeur/ Insérer/ Contrôle de formulaire/ Bouton (Contrôle de formulaire)**[1]), puis en rattachant la macro créée au bouton de commande (**clic droit/ Affecter une macro**). Le déclenchement de la macro se fait en cliquant sur le bouton de commande ainsi créé.

[1] Pour les Mac, l'insertion d'un bouton de commande se fait par **Développeur/Bouton de commande.**

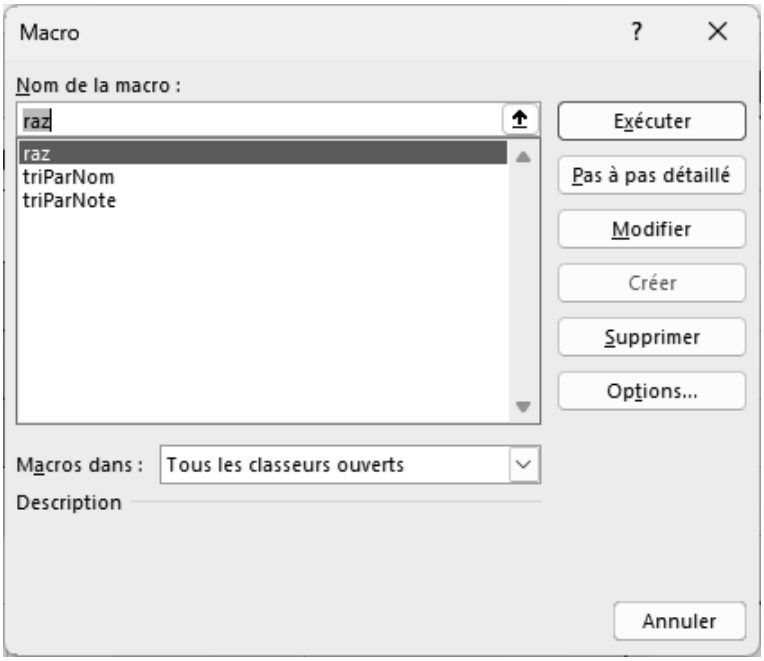


Figure 12 : La fenêtre Macros (Développeur/Macros)

Comment permettre au programme d'interagir avec l'utilisateur ?

Plusieurs solutions sont possibles pour faire interagir l'utilisateur avec le programme. Elles seront détaillées dans le chapitre 4.

Nous pouvons citer principalement trois méthodes :

- l'utilisation des instructions d'entrées/sorties **InputBox** et **MsgBox**, dans le code VBA,
- l'utilisation **des contrôles de formulaire** du menu « Développeur » pour insérer directement dans la feuille Excel les contrôles graphiques comme des « boutons (de commande) », des « cases à cocher », des « zones de liste modifiable », des « intitulés », etc. Sur PC, les contrôles sont regroupés dans une boîte à outils (**Insérer/Contrôles de formulaire**), voir Figure 2 alors que sur Mac, ils sont directement accessibles dans le menu « Développeur » voir Figure 6.
- la création de **formulaire** (UserForm) via l'éditeur Visual Basic (**Insertion/ UserForm**). Une boîte à outils apparaît alors et permet d'ajouter les contrôles graphiques nécessaires à la boîte de dialogue. Le fonctionnement de cette boîte de dialogue nécessitera la compréhension des principes de la programmation événementielle.

Attention, dans les versions d'Excel pour Mac, la création de formulaires via l'éditeur Visual Basic n'est pas possible.

Il est possible de citer une dernière manière de faire interagir l'utilisateur et le programme, présentée au chapitre 3 : l'appel d'une fonction programmée en VBA dans une cellule d'une feuille de calcul Excel en précisant ses arguments. VBA, lui, affiche le résultat de la fonction appelée.

V. Enregistrement direct d'une macro

Excel permet de créer des macros (procédures sans paramètre, par exemple : sub raz()) par enregistrement des actions de l'utilisateur, sans avoir besoin de connaître la syntaxe précise des commandes qui seront exécutées. Cette façon de faire permet de « rejouer » un nombre indéterminé de fois une même séquence d'instructions réalisées une première fois manuellement. L'utilisateur peut ainsi créer une macro avec l'aide de VBA (**Développeur/ Enregistrer une/la macro**) pour automatiser l'exécution d'une commande Excel sans connaître le langage, par exemple, pour relancer un tri, un filtrage avancé ou pour effacer le contenu de cellules.

Vous pourrez aussi, par ce biais, avoir une idée d'une des façons possibles d'écrire du code en **VBA (Visual Basic for Applications**) pour ce que vous désirez faire, avant d'incorporer ce fragment de code dans une procédure que vous êtes en train d'écrire, quitte à modifier le code généré pour qu'il s'adapte à vos besoins.

Une fois l'enregistrement de macro démarré, il faut choisir un nom pour la macro (par défaut « Macro1 », voir Figure 13). Attention, n'utilisez pas des mots-clés du langage VBA ni un nom déjà donné à une macro du classeur créée précédemment. Le caractère « espace » est interdit dans le nom d'une macro, vous pouvez utiliser le caractère « souligné » (_) ou mettre des majuscules en début de mot si votre nom en comporte plusieurs.

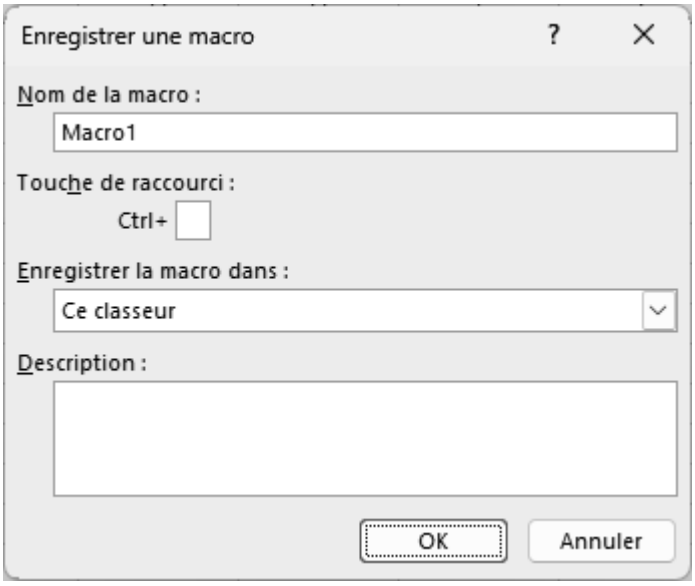


Figure 13 : La fenêtre « Enregistrer une macro »

Il convient ensuite d'enchaîner les commandes Excel que vous souhaitez automatiser grâce à cette macro. N'oubliez pas d' « Arrêter l'enregistrement » (avec le même bouton que pour le démarrage) une fois votre macro terminée.

Exemple 1 : enregistrement de trois macros simples

L'exemple suivant présente un classeur Excel avec macros comprenant deux feuilles Excel respectivement nommées « Exemple_petit_dej » et « Exemple1-notes etudiants » et un module comprenant trois macros nommées « raz() », « triParNom() », « triParNote() », créées par enregistrement de macro. Pour améliorer la compréhension, des commentaires ont été ajoutés dans le code VBA, une fois les macros enregistrées, en utilisant l'éditeur Visual Basic.

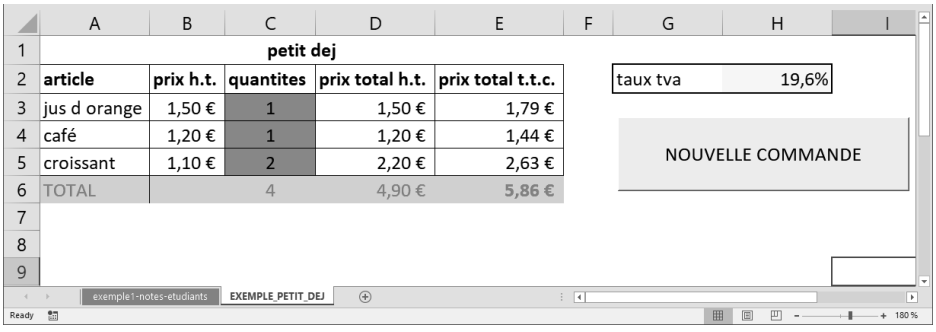


Figure 14 : La feuille de calcul « EXEMPLE_PETIT_DEJ »

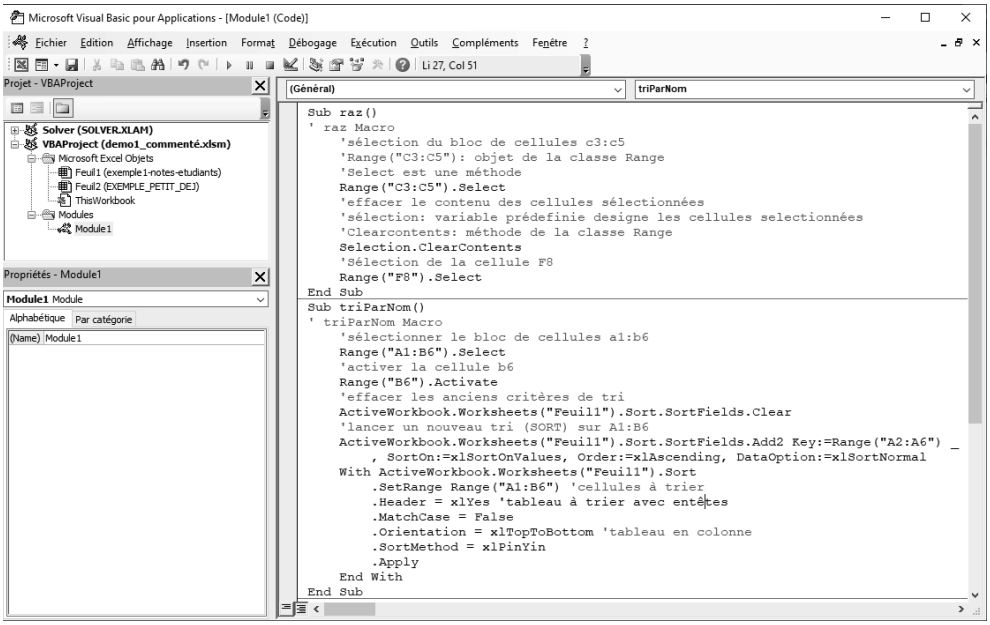


Figure 15 : Code VBA généré par enregistrement de macro pour raz()

```
Sub raz()
' raz Macro
    'sélection du bloc de cellules c3:c5
    'Range("C3:C5"): objet de la classe Range
    'Select est une méthode
    Range("C3:C5").Select
    'effacer le contenu des cellules sélectionnées
    'Selection: variable prédéfinie qui désigne les cellules sélectionnées
    'Clearcontents: méthode de la classe Range
    Selection.ClearContents
    'sélection la cellule F8
    Range("F8").Select
End Sub
```

La macro ci-dessus (créée par enregistrement de macro (**Développeur/ Enregistrer une/la macro**)), nommée « raz » (pour « remise à zéro »), efface le contenu du bloc de cellules C3:C5 et se positionne dans la cellule F8.

L'ajout d'un bouton « Nouvelle commande » (**Développeur/Contrôles/Insérer/ Contrôles de formulaire/Bouton (Contrôle de formulaire)** sur PC ou **Développeur/Bouton** sur Mac) auquel la macro « raz() » est rattachée, dans la feuille Excel (voir Figure 14), permettra d'exécuter la macro autant de fois que souhaité.

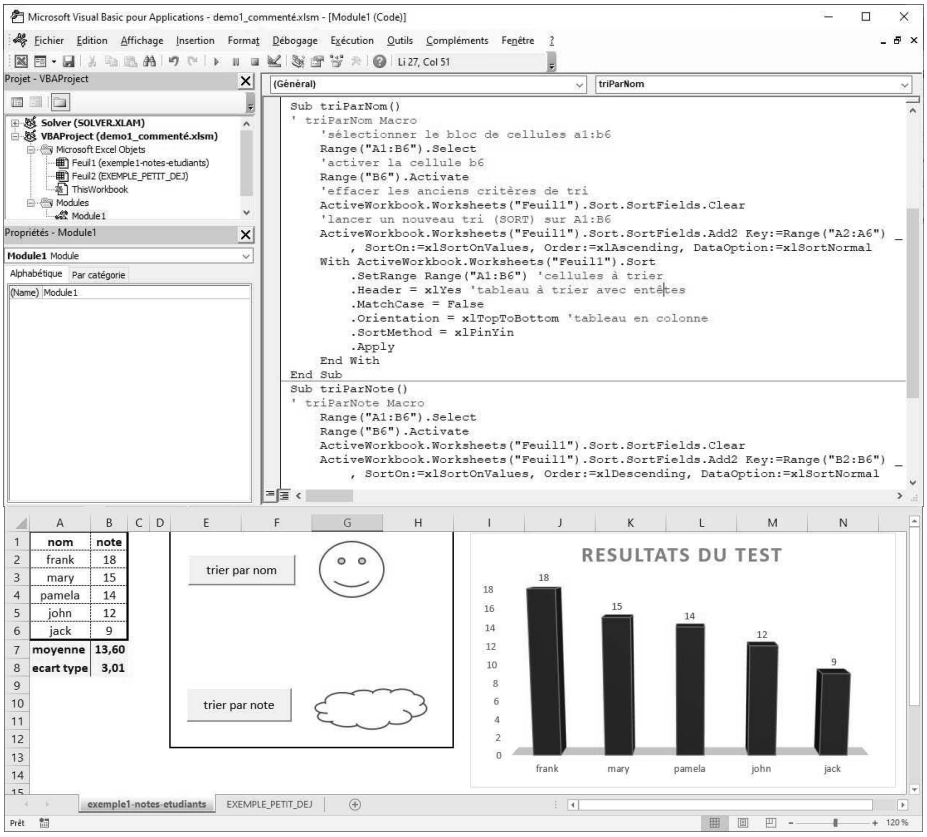


Figure 16 : Code VBA associé à la feuille exemple1-notes-etudiants

Dans la feuille « exemple1-notes-etudiants » se trouve, dans les cellules A2 à B6 une liste de noms et de notes d'étudiants.

Nous souhaitons créer une seconde macro nommée « triParNom » qui trie les cellules A1 à B6 par ordre alphabétique des noms. Cette création de macro est réalisée par enregistrement de macro (**Développeur/ Enregistrer une/la macro**).

```
Sub triParNom()
' triParNom Macro
    'sélection du bloc de cellules a1:b6
    Range("A1:B6").Select
    'activer la cellule b6
    Range("B6").Activate
    'effacer les anciens critères de tri
    ActiveWorkbook.Worksheets("Feuil1").Sort.SortFields.Clear
    'lancer un nouveau tri (SORT) sur A1:B6
    ActiveWorkbook.Worksheets("Feuil1").Sort.SortFields.Add2
Key:=Range("A2:A6") _
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Feuil1").Sort
        .SetRange Range("A1:B6") 'cellules à trier
        .Header = xlYes 'tableau à trier avec entetes
        .MatchCase = False
        .Orientation = xlTopToBottom 'tableau en colonne
        .SortMethod = xlPinYin
        .Apply
    End With
End Sub
```

Dans la feuille de calcul « exemple1-notes-etudiants », un bouton de commande « trier par nom » auquel la macro « triParNom() » est rattachée, a été ajouté. La macro peut également être rattachée à une **forme** (**Insertion/Illustrations/Formes**), par exemple « *un smiley* », où il faudra cliquer pour exécuter la macro.

Nous souhaitons maintenant créer une troisième macro nommée « triParNote » qui trie les cellules A1 à B6 par ordre décroissant des notes. Cette création de macro est réalisée par enregistrement de macro (**Développeur/ Enregistrer une/la macro**).