Guide pratique de programmation VBA pour Excel

Maîtriser les fondamentaux du langage VBA

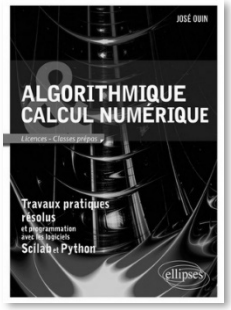
José OUIN

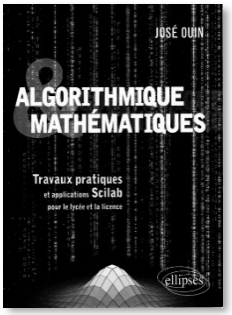
Ingénieur I.N.S.A Toulouse
Professeur Agrégé de Génie Civil
Professeur Agrégé de Mathématiques

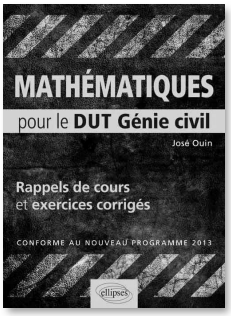
Site Internet de ressources pédagogiques : https://www.joseouin.fr

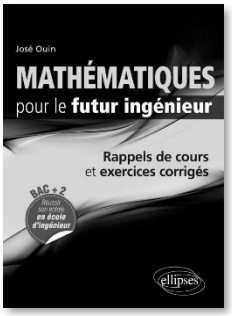
Tutoriels vidéo : https://www.youtube.com/c/mathematiquesmagiques

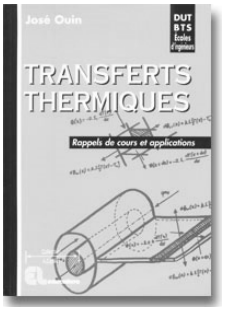
Du même auteur aux Editions Ellipses et Educalivre

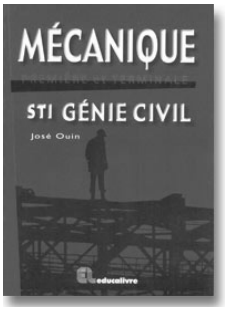


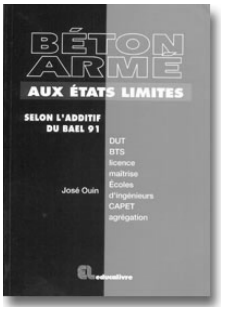












ISBN : 978-2-9592760-0-2

www.joseouin.fr

2- Est-ce que les macros en langage VBA peuvent être remplacées par des calculs dans les cellules du tableur ?

Voici quelques arguments et exemples concrets qui démontrent pourquoi la programmation de macros VBA pour Excel ne peut pas toujours être remplacée par des calculs simples avec des formules dans les cellules :

- **Automatisation de tâches complexes :**

Exemple : Si vous devez effectuer des tâches complexes impliquant des itérations, des conditions, des boucles, etc., la programmation VBA offre une approche beaucoup plus puissante. Par exemple, la création d'une macro pour traiter une grande quantité de données en fonction de critères spécifiques.

- **Manipulation avancée des données :**

Exemple : La programmation VBA permet la manipulation avancée des données, y compris la fusion de données provenant de différentes feuilles de calcul, la transformation de données selon des règles spécifiques, et la création de rapports complexes qui ne peuvent pas être réalisés avec des formules standard.

- **Interaction avec des applications externes :**

Exemple : Si vous avez besoin d'importer/exporter des données vers/depuis d'autres applications (par exemple, des bases de données externes, des fichiers texte, des API), la programmation VBA offre la flexibilité nécessaire pour automatiser ces processus.

- **Personnalisation d'interfaces utilisateur :**

Exemple : La programmation VBA permet de créer des interfaces utilisateur personnalisées avec des boîtes de dialogue, des formulaires et des contrôles interactifs, offrant une expérience utilisateur bien plus riche que ce qui peut être réalisé avec des formules de cellules Excel.

- **Gestion d'événements complexes :**

Exemple : La programmation VBA permet de gérer des événements complexes tels que des changements dans une feuille de calcul, des clics de boutons, etc. Cela peut être utilisé pour déclencher des actions spécifiques automatiquement, ce qui est difficile à réaliser avec des formules standard.

- **Optimisation des performances :**

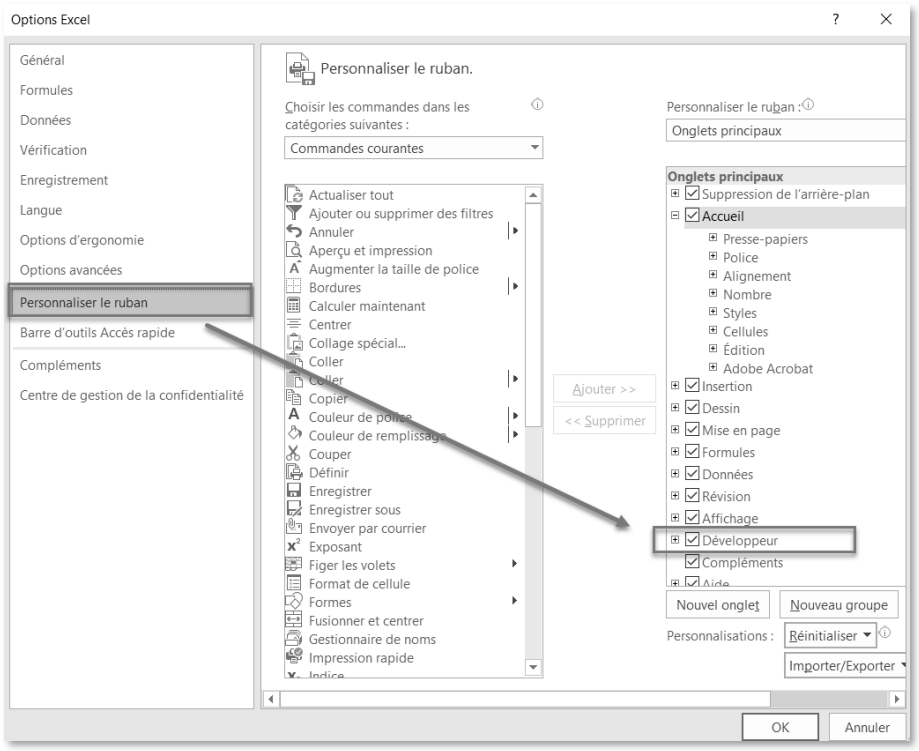
Exemple : Pour des ensembles de données massifs, la programmation VBA peut être plus efficace en termes de performances par rapport à des formules complexes dans les cellules, car elle permet un contrôle plus fin sur le flux d'exécution du code.

En résumé, bien que les formules dans les cellules Excel soient puissantes, la programmation de macros VBA offre une flexibilité et une puissance beaucoup plus grandes pour automatiser des tâches complexes, manipuler des données de manière avancée et créer des solutions sur mesure en fonction des besoins spécifiques.

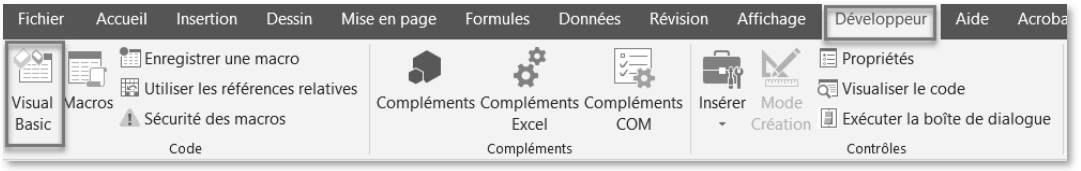
Excel et l'éditeur Visual Basic

1- Accès à l'éditeur Visual Basic

Affichage du ruban « Développeur » : Cliquez sur « **Fichier/Options/Personnaliser le ruban** » puis cocher « **Développeur** ».



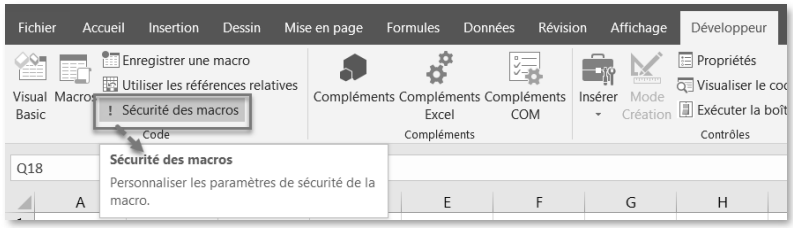
- **Utilisation du ruban Excel :**
  - **Onglet "Développeur" :** L'onglet "Développeur" dans le ruban Excel donne accès à divers outils de développement, y compris l'éditeur VBA.
  - **Bouton "Visual Basic" :** En cliquant sur le bouton "Visual Basic", l'éditeur VBA s'ouvre, offrant un environnement dédié à la programmation.

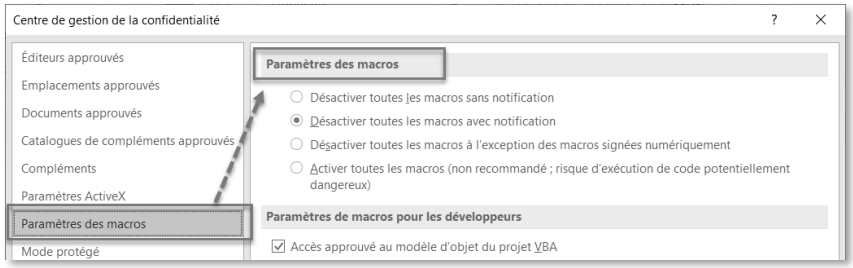


- **Raccourcis clavier :**
  - **Alt + F11 :** Une autre méthode d'accès rapide à l'éditeur VBA consiste à utiliser le raccourci clavier "Alt + F11". Ce raccourci bascule rapidement entre Excel et l'éditeur VBA.

2- Sécurité des macros

Le bouton de menu « Sécurité des macros » du ruban « Développeur » permet de régler les paramètres de sécurité des macros :





Désactiver toutes les macros sans notification :

- Description : Avec cette option activée, Excel désactive automatiquement toutes les macros sans vous avertir.
- Impact sur la programmation VBA : Les macros, même celles provenant de sources de confiance, ne seront pas exécutées sans notification explicite.

Désactiver toutes les macros avec notification :

- Description : Excel demande à l'utilisateur si les macros doivent être exécutées ou non dans le classeur qui a été ouvert.
- Impact sur la programmation VBA : Dès l'ouverture du classeur Excel, les utilisateurs seront avertis que des macros existent. Ils peuvent choisir de les exécuter ou non.

Désactiver toutes les macros à l'exception des macros signées numériquement :

- Description : Cette option empêche l'exécution de toutes les macros, sauf celles qui ont été signées numériquement par un éditeur de confiance.
- Impact sur la programmation VBA : Si cette option est activée, seules les macros signées numériquement pourront être exécutées, ce qui peut ajouter une couche de sécurité supplémentaire.

Activer toutes les macros (non recommandé, risque de sécurité potentiel) :

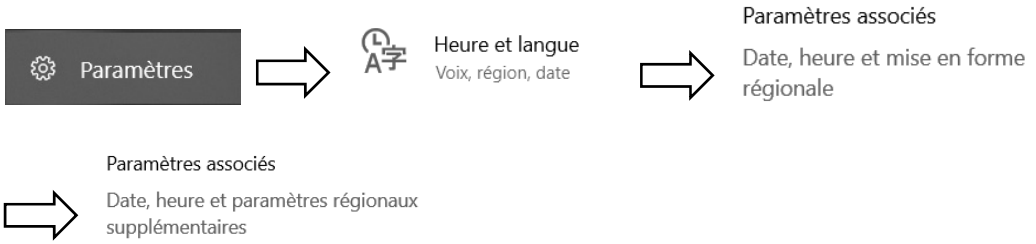
- Description : Cette option permet l'exécution de toutes les macros sans aucune restriction ni notification.
- Impact sur la programmation VBA : Toutes les macros, y compris celles potentiellement dangereuses, seront exécutées **sans avertissement**.

Il est important de noter que la dernière option ("Activer toutes les macros") est considérée comme non recommandée en raison des risques potentiels pour la sécurité. Il est généralement recommandé de maintenir des paramètres de sécurité plus stricts pour éviter l'exécution non autorisée de macros potentiellement malveillantes.

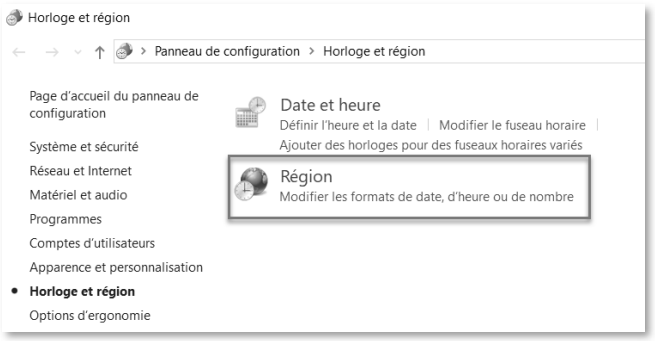
3- Symbole décimal : réglage des paramètres régionaux

Le symbole décimal est fonction du réglage des paramètres régionaux qui ont été définis dans l'environnement Windows : le point « **.** » ou la virgule « **,** ».

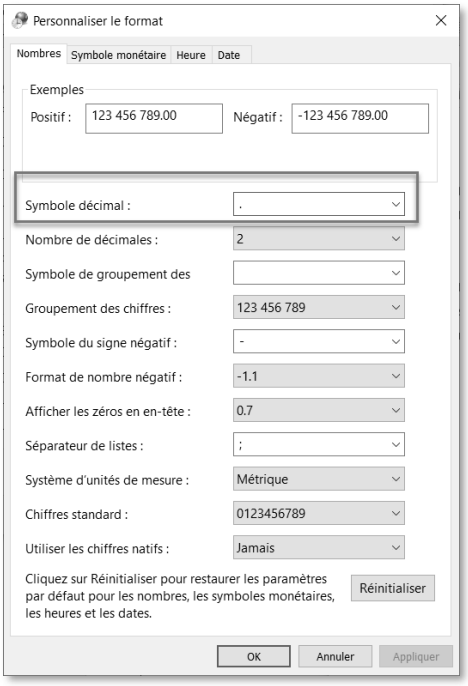
Cliquez sur « Paramètres »



On sélectionne alors « Région » :



On choisit le symbole décimal « **.** » ou « **,** » :



L'affectation et les types de variables

1- L'affectation

L'instruction d'affectation permet d'attribuer une valeur à une variable. Par exemple :
`Age = 25` affecte la valeur **25** à la variable `Age`.
Le type de données doit être défini en fonction de la valeur prise par la variable.

```
' Déclaration des variables
Dim Age As Integer
Dim Message as String
Dim Rayon As Double
Dim Valeur As Boolean

' Affectation de valeurs
Age = 25
Message = "Bonjour"
Rayon = 12.4568
Valeur = False
```

2- Les types de variables

Variant :
Peut contenir tous les types de données.
Toutes les variables sont converties en type Variant si aucun autre type de données n'est explicitement déclaré.

Boolean :
Données pouvant prendre exclusivement les valeurs logiques True et False.

Integer :
Données contenant des nombres entiers, de 16 bits, compris entre -32 768 et 32 767.
Remarque :

Long :
Nombre entier de 32 bits dont la valeur est comprise entre -2 147 483 648 et 2 147 483 647.
Si vous lui affectez un nombre décimal, la valeur de la variable est arrondie à l'entier le plus proche.

Currency :
Données de 64 bits dont la plage de valeurs s'étend de -922 337 203 685 477,5808 à 922 337 203 685 477,5807.
Ce type de donnée est utilisé dans les calculs monétaires ou dans les calculs à virgule fixe pour lesquels une grande précision est requise.

Single :
Type de données, sur 32 bits, qui regroupe des variables à virgule flottante en simple précision sous forme de nombres à virgule flottante, dont la valeur est comprise entre -3,402823E38 et -1,401298E-45 pour les valeurs négatives, et entre 1,401298E-45 et 3,402823E38 pour les valeurs positives.

Double :
Type de données, sur 64 bits, stockant les nombres à virgule flottante en double précision compris entre -1,79769313486231E308 et -4,94065645841247E-324 pour les valeurs négatives, et entre 4,94065645841247E-324 et 1,79769313486232E308 pour les valeurs positives.

Date :
Type de données, sur 64 bits, utilisé pour stocker les dates et les heures.

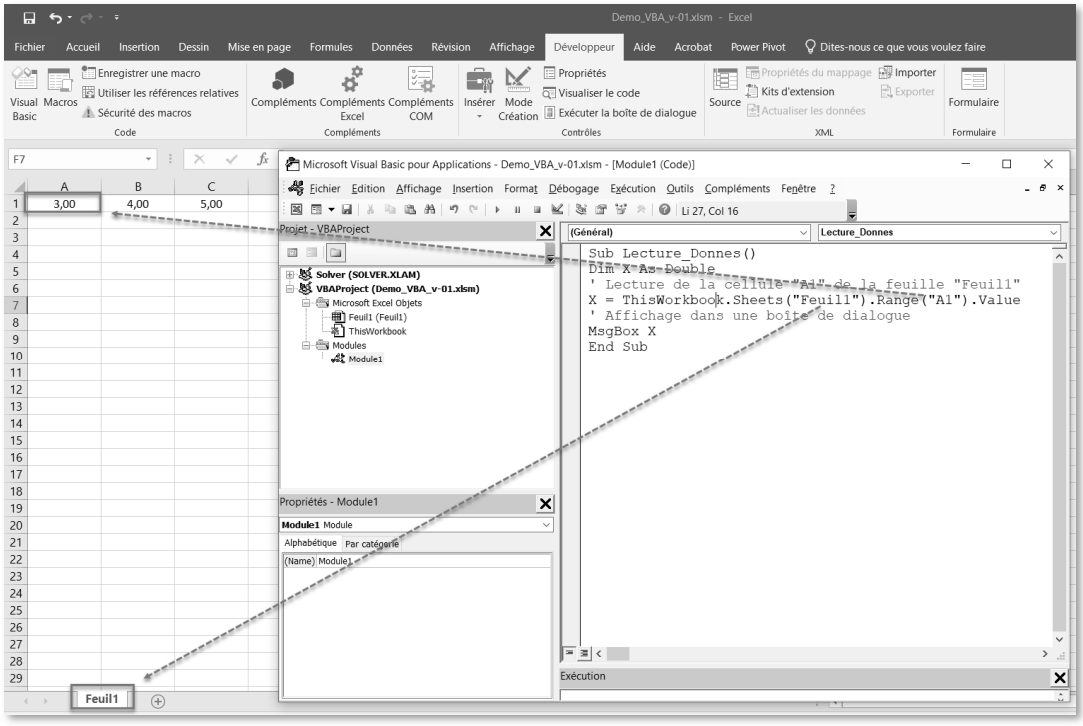
String :
Type de données, sur 64 bits, utilisé pour stocker des chaînes de caractères.

Les instructions d'entrée-sortie

1- Les instructions d'entrée des données à partir d'une feuille de calcul

La procédure « Lecture_Donnees() » permet de lire la valeur présente dans la cellule « A1 » de la feuille « Feuil1 » du classeur :

```
1       Sub Lecture_Donnes()
2       Dim X As Double
3       ' Lecture de la cellule "A1" de la feuille "Feuil1"
4       X = ThisWorkbook.Sheets("Feuil1").Range("A1").Value
5       ' Affichage dans une boîte de dialogue
6       MsgBox X
7       End Sub
```



Les procédures et les fonctions

1- Les procédures

Une procédure exécute une action sans fournir de valeur explicite. Sa syntaxe est la suivante :

```
1      Sub DemoSub()
2      Dim X1 As Double
3      Dim X2 As Double
4      Dim LH As Double
5      ' Affectation de valeurs
6      X1 = 14.56
7      X2 = 31.19
8      ' Calcul de l'hypoténuse LH
9      LH = Hypotenuse(X1, X2)
10     ' Affichage de LH dans une boîte de dialogue
11     MsgBox LH
12     End Sub
```

Cet exemple définit une procédure appelée « **DemoSub()** » qui contient du code accessible depuis tout point du programme. L'appel s'effectue en insérant le nom de la procédure à l'endroit adapté du programme.
Cette procédure appelle la fonction « **Hypotenuse** » pour calculer la longueur d'une hypoténuse qui sera ensuite affichée dans une boîte de dialogue **Msgbox**.

2- Les fonctions

Une fonction, tout comme une procédure, regroupe un bloc d'instructions à exécuter en une unité logique. Cependant, contrairement à une procédure, **une fonction retourne une valeur**.

Exemple : Fonction **Hypotenuse** :

```
1      Function Hypotenuse(x As Double, y As Double) As Double
2      ' Cette fonction renvoie la longueur de l'hypoténuse
3      ' dans un triangle rectangle.
4      Dim R As Double
5      R = ((x ^ 2) + (y ^ 2)) ^ 0.5
6      Hypotenuse = R
7      End Function
```

La valeur de retour est assignée par une simple affectation : **Hypotenuse = R**

Les instructions conditionnelles

Les instructions conditionnelles permettent de n'exécuter un bloc de code que lorsqu'une condition particulière est remplie. La structure de contrôle If-Then-Else et If-ElseIf constitue l'un des piliers fondamentaux de la programmation, permettant aux développeurs de prendre des décisions conditionnelles au sein de leurs scripts. En VBA, cette structure est particulièrement puissante et tout à fait indispensable.

Intérêt de la Structure If-Then-Else :

La structure If-Then-Else offre la possibilité d'exécuter des blocs de code conditionnels en fonction de l'évaluation d'une expression booléenne. Si l'expression est vraie, le bloc de code dans la clause "Then" est exécuté. Dans le cas contraire, le bloc de code dans la clause "Else" prend le relais. Cette flexibilité permet de construire des programmes capables de s'adapter dynamiquement aux différentes situations rencontrées lors de l'exécution.

1- Instruction If...Then...Else...End If

```
1      Dim B As Integer
2      B = 5
3      If A > 3 Then
4          B = 2
5      Else
6          B = 0
7      End If
```

Dans cet exemple, la valeur 2 est assignée à la variable **B** si la variable **A** est supérieure à 3. Dans le cas contraire (donc si **A** est inférieure ou égale à 3), la valeur 0 est assignée à **B**.

Pour les situations plus complexes, on peut imbriquer plusieurs instructions `If`, par exemple :

```
1      Dim B As Integer
2      B = 5
3      If A = 0 Then
4          B = 0
5      ElseIf A < 3 Then
6          B = 1
7      Else
8          B = 2
9      End If
```

Si la valeur de la variable **A** est égale à zéro, alors la valeur 0 est assignée à **B**. Si la valeur de **A** est inférieure à 3 (mais non égale à zéro), alors la valeur 1 est assignée à **B**. Dans tous les autres cas (c'est-à-dire si la valeur de **A** est supérieure ou égale à 3), la valeur 2 est assignée à **B**.

Les boucles bornées

Une boucle répète l'exécution d'un bloc de code un nombre de fois donné. Certaines boucles peuvent également se répéter indéfiniment.

1- Boucle For…Next

La boucle **`For Next`** est utilisée en VBA pour répéter un ensemble d'instructions un certain nombre de fois. Voici un exemple concret pour illustrer son utilisation :

Supposons que l'on souhaite afficher les carrés des nombres de 1 à 5. On peut utiliser une boucle **`For Next`** pour accomplir cela de manière efficace. Voici le code :

```
1       Sub AfficherCarres()
2           Dim i As Integer
3
4       ' Utilisation de la boucle For Next pour afficher les carrés des
        nombres de 1 à 5
5           For i = 1 To 5
6               MsgBox "Le carré de " & i & " est : " & i ^ 2
7           Next i
8       End Sub
```

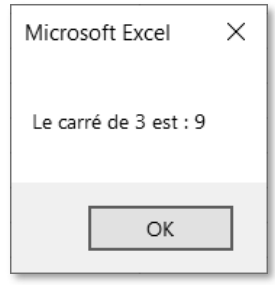
Dans ce code :

- Dim i As Integer déclare une variable i de type entier.
- For i = 1 To 5 initialise la boucle For Next en disant que i doit prendre des valeurs de 1 à 5 (inclus).
- Next i marque la fin de la boucle. Après l'exécution du code entre For et Next, i est incrémenté automatiquement de 1, et la boucle est répétée jusqu'à ce que i atteigne la valeur spécifiée dans To, dans ce cas, 5.

À chaque itération de la boucle, une boîte de dialogue s'affiche avec le message indiquant le carré du nombre actuel dans la boucle (i). Cela se poursuit jusqu'à ce que tous les nombres de 1 à 5 aient été traités.

Le résultat des carrés affichés est :

- Le carré de 1 est : 1
- Le carré de 2 est : 4
- Le carré de 3 est : 9
- Le carré de 4 est : 16
- Le carré de 5 est : 25

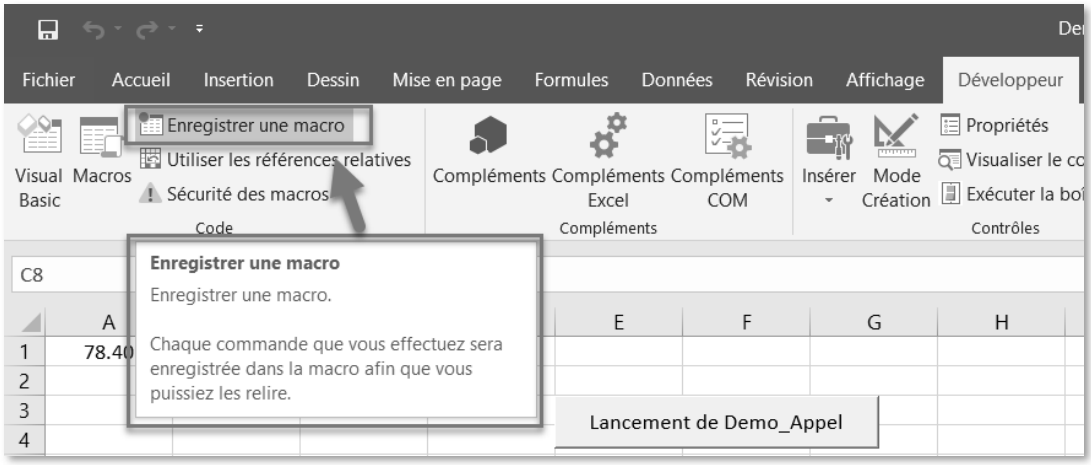


La boucle `For Next` est donc une structure de contrôle très utile pour automatiser des tâches répétitives en exécutant un ensemble d'instructions un certain nombre de fois.

Comment enregistrer une macro ?

1- Lancement de l'enregistreur de macros

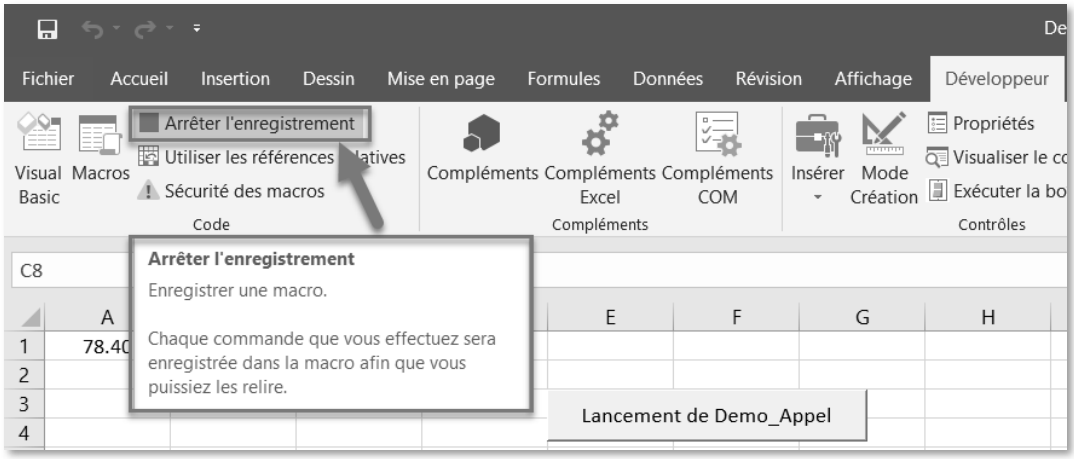
Pour lancer l'enregistrement d'une macro, sélectionner le menu :
Développeur/Enregistrer une macro



Il faut ensuite choisir le nom de la macro et son emplacement dans le module de votre choix. Il est possible à ce stade de créer un nouveau module. Valider ensuite par « Enregistrer ».

2- Arrêt de l'enregistreur de macros et sauvegarde de la macro

Pour mettre fin à l'enregistrement, cliquer sur « Arrêter l'enregistrement ».



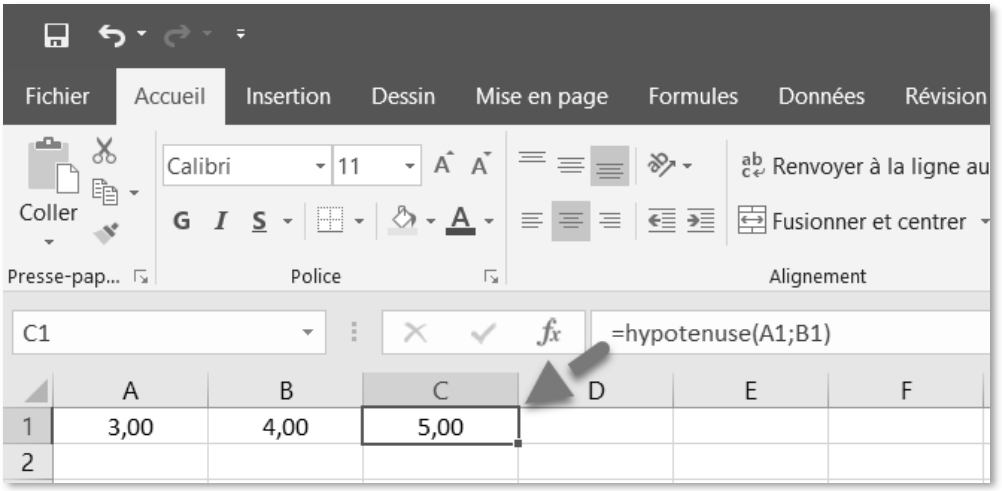
Comment exécuter une macro ?

1- Exécuter une macro à l'aide d'une formule dans le tableur

Cette macro « **Hypotenuse** » permet de calculer la longueur de l'hypoténuse dans un triangle rectangle

```
1       Function Hypotenuse(x As Double, y As Double) As Double
2       ' Cette fonction renvoie la longueur de l'hypoténuse
3       ' dans un triangle rectangle.
4       Dim R As Double
5       R = ((x ^ 2) + (y ^ 2)) ^ 0.5
6       Hypotenuse = R
7       End Function
```

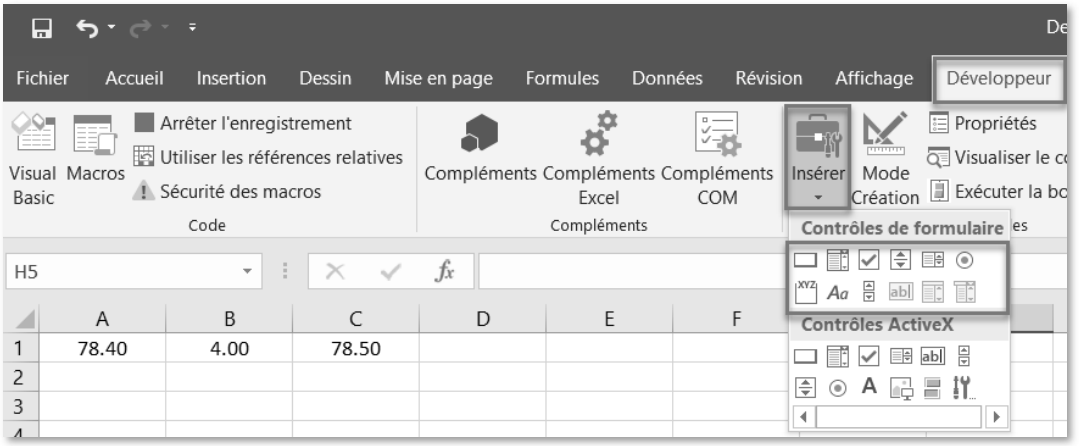
Après avoir créé une fonction, on peut l'utiliser dans une formule au sein du tableur Excel :



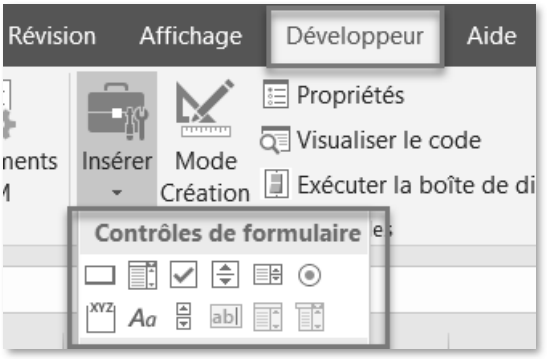
Les contrôles de formulaire

Dans un classeur Excel, les contrôles de formulaire permettent de préciser les choix de l'utilisateur. On ne considère dans cet ouvrage que les contrôles suivants :

- Boutons d'option ;
- Case à cocher ;
- Bouton ;
- Liste déroulante (Données/Validité).



Ces contrôles peuvent être placés dans des boîtes de dialogue ou dans des formulaires au sein d'une feuille de calcul. On se limitera dans ce guide aux contrôles de formulaires placés directement dans une feuille de calcul.



Utilisation du solveur d'Excel

1- Définition du solveur d'Excel

Le Solveur d'Excel est un outil d'optimisation qui fait partie du logiciel Microsoft Excel. Il permet de résoudre des problèmes complexes en ajustant les valeurs de certaines cellules pour optimiser une formule ou une fonction objectif, tout en respectant des contraintes spécifiées. Le Solveur est particulièrement utile pour les problèmes de programmation linéaire, de programmation non linéaire, d'optimisation et d'analyse de scénarios.

Principales caractéristiques du Solveur d'Excel :

1. **Optimisation :** Le Solveur aide à trouver la meilleure valeur pour une cellule objectif tout en satisfaisant un ensemble de contraintes. Cela peut être utilisé pour maximiser ou minimiser une fonction, par exemple, maximiser les profits ou minimiser les coûts.
2. **Contraintes :** Il permet de définir des contraintes sur les variables ajustées, ce qui permet de refléter des conditions du monde réel dans le modèle.
3. **Variables :** Les variables représentent les cellules dont les valeurs peuvent être modifiées par le Solveur pour optimiser la fonction objectif.
4. **Fonction objectif :** C'est la cellule ou la formule que vous souhaitez maximiser ou minimiser.
5. **Rapport de résolution :** Une fois le Solveur exécuté, il fournit un rapport détaillé indiquant les valeurs optimales des variables, la valeur de la fonction objectif optimisée, ainsi que d'autres informations pertinentes.

Principales utilisations du Solveur d'Excel :

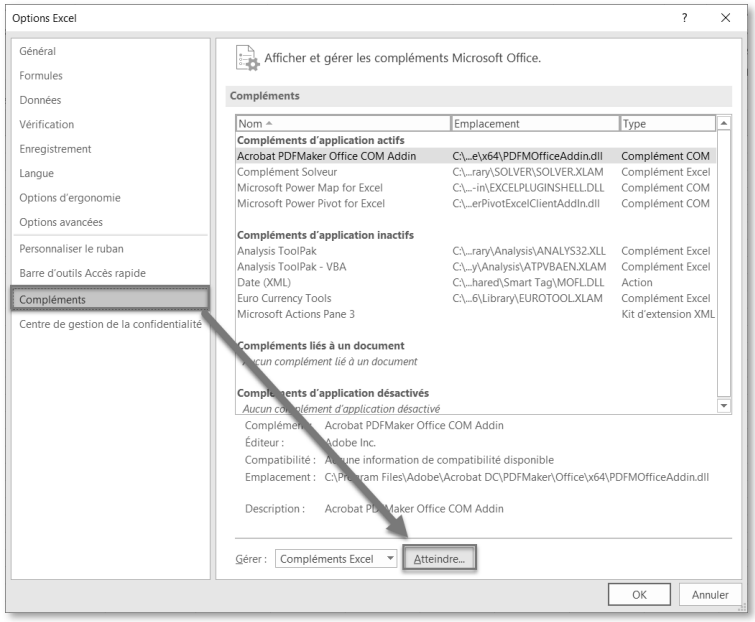
1. **Planification de la production :** Optimiser les niveaux de production pour maximiser les profits tout en respectant les contraintes de ressources.
2. **Allocation de ressources :** Affecter des ressources limitées (argent, personnel, matériaux) de manière optimale pour atteindre des objectifs spécifiques.
3. **Gestion de portefeuille :** Optimiser un portefeuille d'investissements en maximisant les rendements tout en respectant des contraintes de risque.
4. **Problèmes de transport et d'affectation :** Trouver la meilleure manière de transporter des biens d'un endroit à un autre ou d'affecter des tâches à des ressources.
5. **Analyse de scénarios :** Tester différentes conditions et voir comment elles affectent la solution optimale.

Le Solveur d'Excel offre une approche puissante pour résoudre divers problèmes d'optimisation sans avoir à écrire de code complexe. Il est particulièrement utile pour les professionnels de la finance, de la logistique, de la gestion de projet et d'autres domaines où des décisions complexes doivent être prises pour maximiser les résultats tout en respectant des contraintes spécifiques.

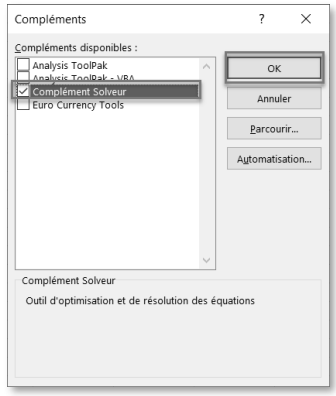
2- Installation du Solveur d'Excel

Le solveur d'Excel n'est pas installé par défaut, il faut afficher son icône en procédant comme indiqué ci-après.

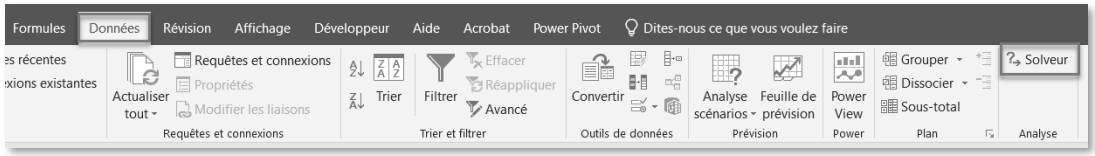
Cliquez sur **Fichier/Options** puis Compléments. Cliquez ensuite sur « **Atteindre** ».



Sélectionner alors le « Complément Solveur » puis valider par « OK ».



L'icône du solveur apparaît alors dans le ruban « Données ».



Utilisation des fonctions intégrées à Excel dans des macros VBA

1- Introduction

L'utilisation de VBA (Visual Basic for Applications) dans Excel offre une puissante possibilité d'automatiser des tâches, de créer des fonctionnalités personnalisées et d'étendre les capacités du logiciel de tableur. Dans cette partie, on explore comment intégrer les fonctions du tableur Excel directement dans des macros VBA pour optimiser les processus et améliorer l'efficacité.

Lorsque l'on travaille avec VBA dans Excel, la méthode **`WorksheetFunction`** est une ressource précieuse qui permet d'accéder à une grande variété de fonctions intégrées à Excel directement depuis les macros. Cette méthode fournit une interface pour exécuter des fonctions Excel standard, offrant ainsi une puissance de calcul étendue au-delà des capacités de base de VBA.

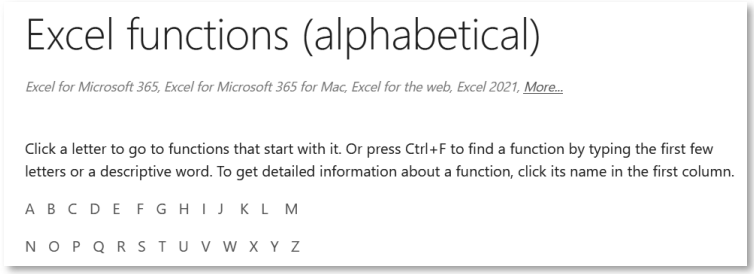
Avantages et Limitations :

- **Avantages :**
  - Accès à une large gamme de fonctions Excel sans avoir à réinventer la roue.
  - Facilité d'utilisation pour des tâches courantes telles que le calcul de statistiques, de fonctions mathématiques, etc.
- **Limitations :**
  - Certaines fonctions peuvent générer des erreurs si les conditions nécessaires ne sont pas remplies.

En utilisant judicieusement la méthode **`WorksheetFunction`**, on peut exploiter la richesse des fonctions Excel directement dans vos macros VBA, simplifiant ainsi le processus de développement et améliorant la lisibilité du code.

Remarque importante :
En VBA, lorsque vous utilisez la méthode **`WorksheetFunction`** pour appeler une fonction Excel, il faut utiliser le nom de la fonction en anglais, même si on utilise la version française d'Excel. Cela est dû au fait que VBA utilise les noms de fonction en anglais pour accéder aux fonctions Excel intégrées.

Liste des fonctions classées dans l'ordre alphabétique :
`Lien` : `https://bit.ly/liste-fonctions-excel`


Excel functions (alphabetical)

Excel for Microsoft 365, Excel for Microsoft 365 for Mac, Excel for the web, Excel 2021, More...

Click a letter to go to functions that start with it. Or press Ctrl+F to find a function by typing the first few letters or a descriptive word. To get detailed information about a function, click its name in the first column.

A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

TP 1 : Convertisseur de températures

1- Présentation du travail pratique

Ce travail pratique est consacré à la création d'un convertisseur de températures en utilisant le langage Visual Basic for Applications (VBA) pour Excel. Ce projet vous offre une excellente occasion de mettre en pratique vos connaissances récemment acquises dans le domaine de la programmation VBA tout en explorant des concepts clés liés à la manipulation de données et aux opérations arithmétiques.

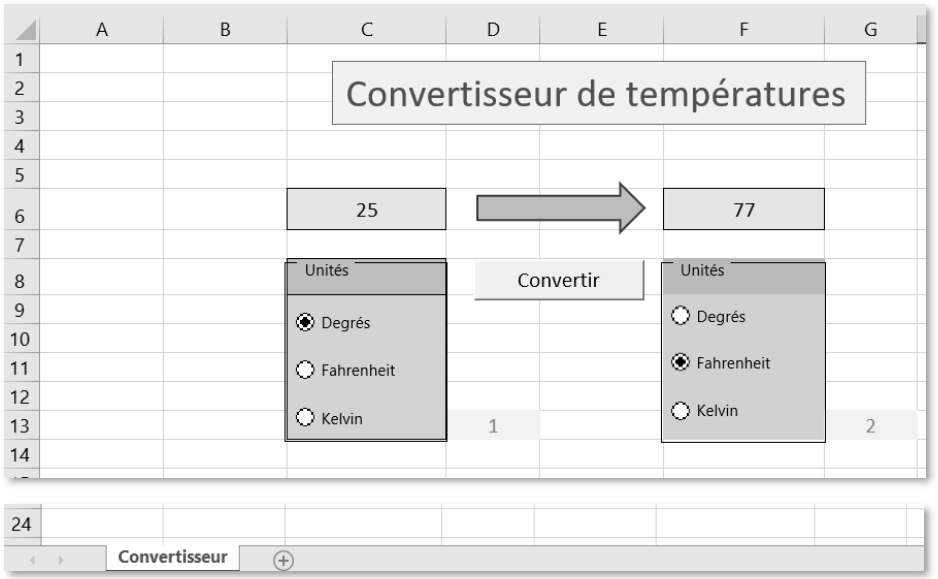
Objectif : L'objectif de ce travail pratique est de concevoir un convertisseur de températures polyvalent qui permettra à l'utilisateur de saisir une valeur dans l'une des échelles de température (Degrés Celsius, Kelvin ou Fahrenheit) et d'obtenir instantanément la température équivalente dans l'autre échelle qui a été sélectionnée.

2- Travail demandé

Il s'agit de créer une interface de calcul ainsi que les macros nécessaires pour effectuer les calculs des différentes températures en prenant en compte le choix de l'unité de la température initiale et celle de la température finale.

3- Exemple d'interface

Voici un exemple de présentation de l'interface :



4- Rappels théoriques

4-1. Le Fahrenheit

Fahrenheit est une échelle de température thermodynamique, où le point de congélation de l'eau est à 32 degrés Fahrenheit (°F) et le point d'ébullition à 212 °F (sous une pression atmosphérique normale).
Cela sépare les points d'ébullition et de congélation de l'eau d'exactement 180 degrés. Par conséquent, un degré sur l'échelle Fahrenheit représente 1/180 de l'intervalle entre le point de congélation et le point d'ébullition de l'eau.

4-2. Le Celcius et le Kelvin

Bien qu'initialement défini comme le point de congélation de l'eau (et plus tard le point de fusion de la glace), l'échelle Celsius est maintenant officiellement une échelle dérivée, définie par rapport à la l'échelle de température **Kelvin**.

Sur l'échelle Celsius le zéro (0 °C) est maintenant défini comme égal à 273,15 K, avec une différence de température de 1 °C équivalent à une différence de 1 K, c'est-à-dire que la taille de l'unité sur chaque échelle est la même. Cela signifie que 100 °C, préalablement défini comme le point d'ébullition de l'eau, est maintenant défini comme l'équivalent de 373,15 K.

4-3. Conversions

Comment obtenir des degrés Kelvin : K

$$K = 273.15 + C$$

$$K = 273.15 + (F - 32)/1.8$$

Comment obtenir des degrés Celcius : C

$$C = K - 273.15$$

$$C = (F - 32)/1.8$$

Comment obtenir des degrés Fahrenheit : F

$$F = 32 + C * 1.8$$

$$F = 32 + (K - 273.15) * 1.8$$

Rappel : le symbole décimal en VBA est le point « . ».

5- Solution

Les codes VBA solutions sont donnés ci-après. Vous avez également à votre disposition le classeur Excel « solution » relatif à ce travail pratique : la page **163** détaille la procédure à suivre pour obtenir les classeurs Excel corrigés.

```
1       Sub Convertisseur()
2       Dim TempI As Double
3       Dim TempF As Double
4       Dim TypeTempI As Integer
5       Dim TypeTempF As Integer
6       '-----------------------
7       ' On lit la valeur de la température dans la cellule C6
8       TempI = ThisWorkbook.Sheets("Convertisseur").Range("C6").Value
9       '---
10      ' On lit la valeur référentielle des boutons options pour l'unité
        initiale
11      TypeTempI = ThisWorkbook.Sheets("Convertisseur").Range("D13").Value
12
13      ' On lit la valeur référentielle des boutons options pour l'unité
        finale
14      TypeTempF = ThisWorkbook.Sheets("Convertisseur").Range("G13").Value
15
16      Select Case TypeTempI
17          Case 1:  'en "Degrés"
18              If TypeTempF = 1 Then '"Degrés"
19                  TempF = TempI
20              ElseIf TypeTempF = 2 Then '"Fahrenheit"
21                  TempF = 32 + TempI * 1.8
22              Else 'Kelvin
23                  TempF = 273.15 + TempI
24              End If
25          Case 2: ' "Fahrenheit"
26              If TypeTempF = 1 Then '"Degrés"
27                  TempF = (TempI - 32) / 1.8
28              ElseIf TypeTempF = 2 Then '"Fahrenheit"
29                  TempF = TempI
30              Else 'Kelvin
31                  TempF = 273.15 + (TempI - 32) / 1.8
32              End If
33          Case 3: '"Kelvin"
34              If TypeTempF = 1 Then '"Degrés"
35                  TempF = TempI - 273.15
36              ElseIf TypeTempF = 2 Then '"Fahrenheit"
37                  TempF = 32 + (TempI - 273.15) * 1.8
38              Else 'Kelvin
39                  TempF = TempI
40              End If
41
42      End Select
44      ' On affiche le résultat dans la cellule F6
45      ThisWorkbook.Sheets("Convertisseur").Range("F6").Value = TempF
48      End Sub
```

Les liens utiles

1- Tutoriels vidéo : Excel - Bureautique

Un ensemble de tutoriels vidéo sur Excel dans le cadre d'une utilisation bureautique.

	https://bit.ly/jo-excel-bureautique

2- Tutoriels vidéo : Excel – Programmation

Un ensemble de tutoriels vidéo sur la programmation sous l'environnement Excel et le langage VBA.

	https://bit.ly/jo-excel-programmation

3- Tutoriels vidéo : Excel – Développements

Un ensemble de tutoriels vidéo sur les développements réalisés sous l'environnement Excel et en utilisant le langage VBA.

	https://bit.ly/jo-excel-developpements

4- Tutoriels vidéo : Excel – Power Query et Power Pivot

Un ensemble de tutoriels vidéo sur Excel et sur l'utilisation de Power Query et Power Pivot.

	https://bit.ly/jo-excel-power

5- Liste des fonctions du tableur Excel

La liste exhaustive des fonctions du tableur Excel pour une utilisation en VBA avec la méthode **`WorksheetFunction`**.

	https://bit.ly/liste-fonctions-excel

6- La chaîne Mathématiques Magiques

La chaîne YouTube « Mathématiques Magiques »

La chaîne YouTube « Mathématiques Magiques » propose un ensemble de ressources pédagogiques portant sur des domaines très variés.

Lien : https://www.youtube.com/c/MathématiquesMagiques

7- Le site de ressources pédagogiques joseouin.fr

Le site Internet « joseouin.fr »

Le site Internet « joseouin.fr » propose un ensemble de tutoriels et de travaux pratiques (Excel, Word, Python, Scilab, Visual Basic, Moodle, Ubuntu, LaTeX, etc.). On y trouve également des énigmes, des QCM en ligne ainsi que des présentations de matériels (tablettes graphiques) et d'ouvrages pédagogiques.

Lien : https://www.joseouin.fr

Cet ouvrage a été achevé en mars 2024
Dépôt légal : mars 2024
Déposé auprès de la BnF (Bibliothèque Nationale de France)