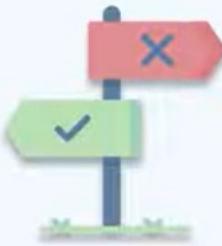


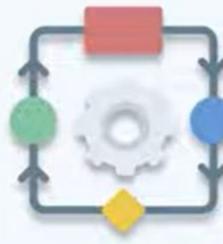
Plan de cours



Instructions
de base



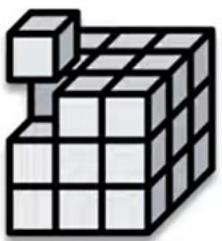
Structures
conditionnelles



Structures
répétitives



Fonctions
et modules



Structures
de données



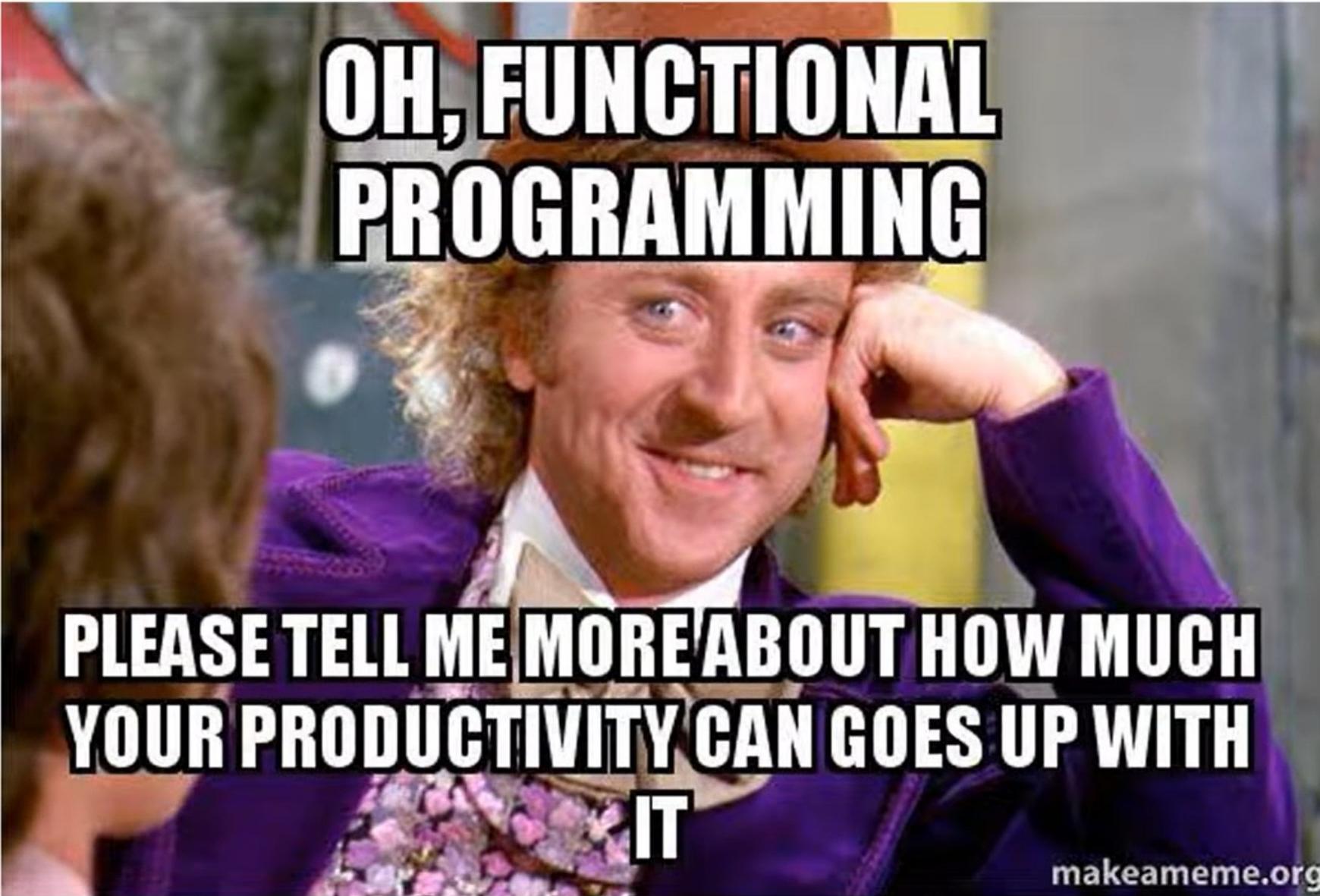
Chaine
de caractères



Gestion
des fichiers



Concepts
avancés



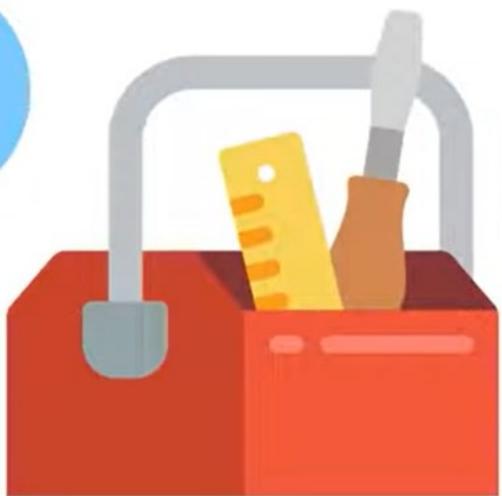
**OH, FUNCTIONAL
PROGRAMMING**

**PLEASE TELL ME MORE ABOUT HOW MUCH
YOUR PRODUCTIVITY CAN GOES UP WITH
IT**

makeameme.org

Fonction dans langage Python

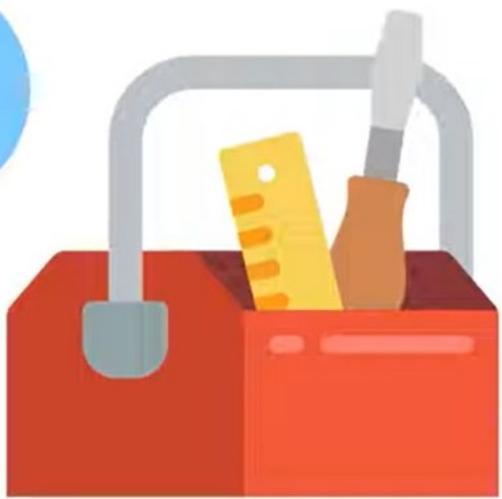
1



Définir la
fonction

Fonction dans langage Python

1



Définir la
fonction

2



Appeler la
fonction

**Pourquoi ma fonction
ne rien fournir en sortie ?**



**Pourquoi ma fonction
ne rien fournir en sortie ?**



**Oh! je n'ai jamais appelé
la fonction.**

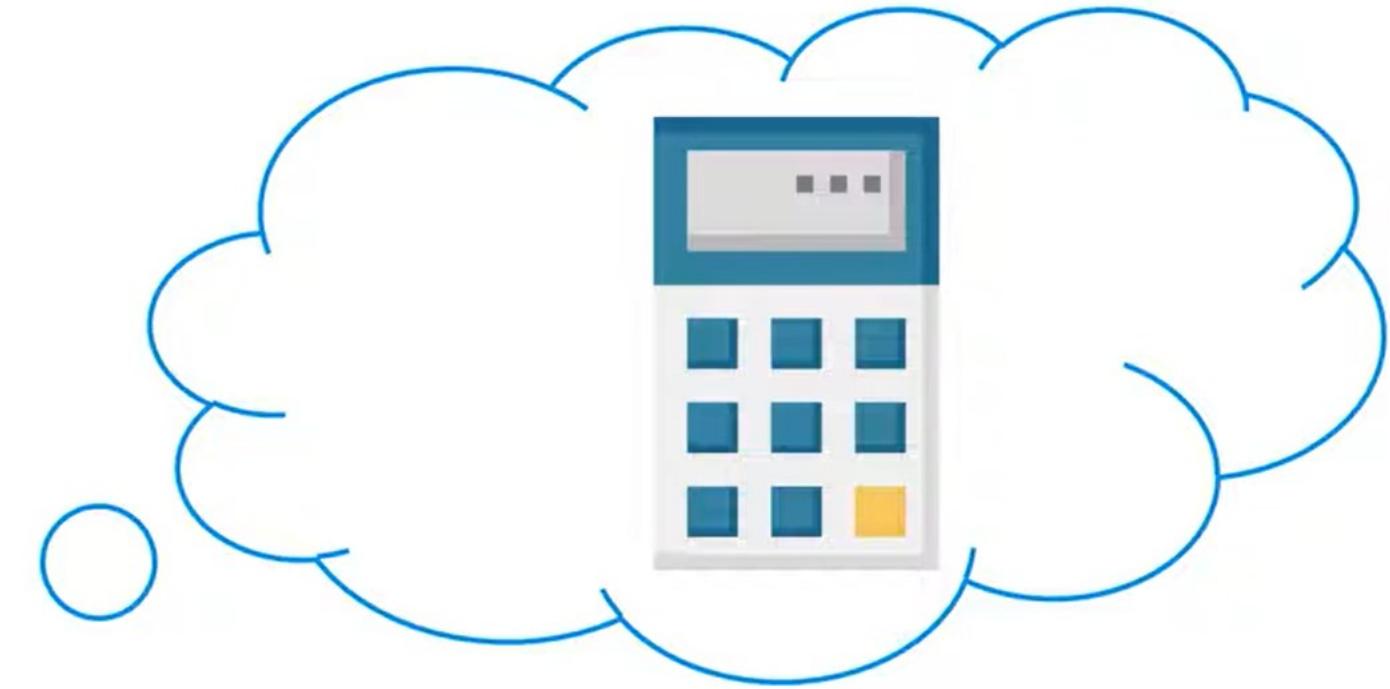
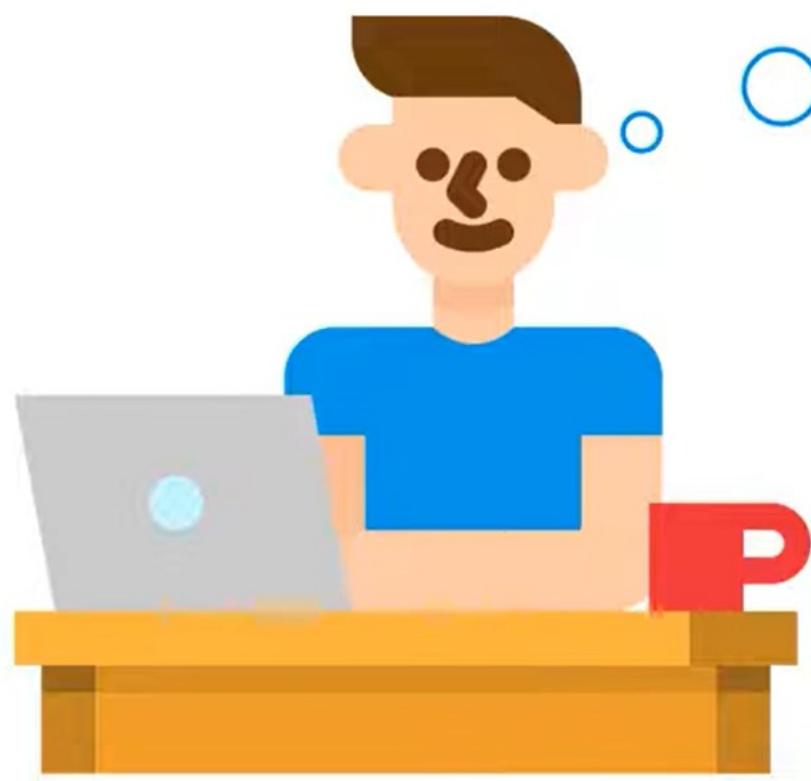


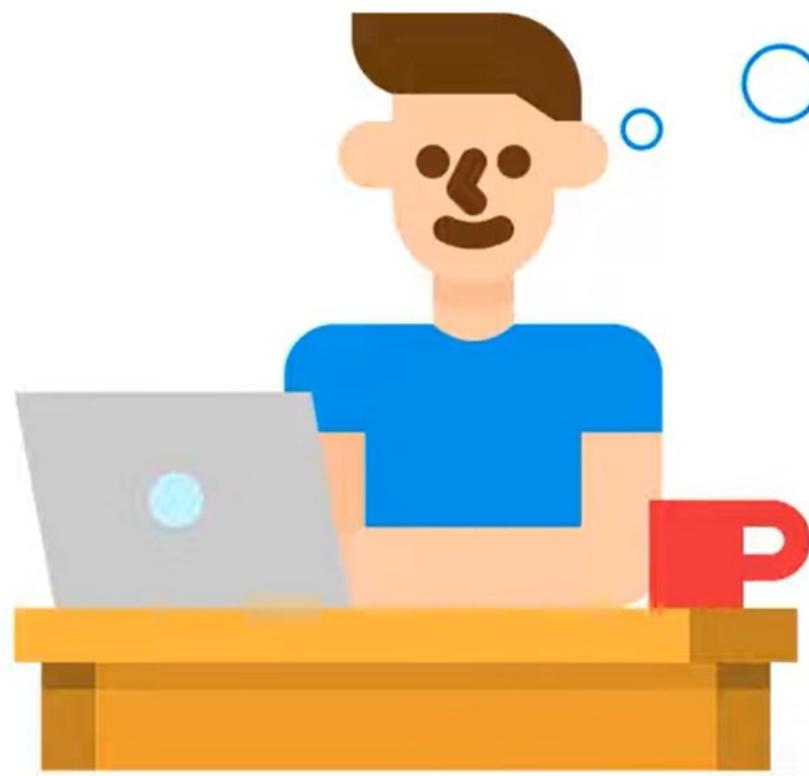
Variables

- Locales
- Globales
- Mot-clé global

Types de paramètres

- **Arguments requis.**
- **Arguments par défaut**
- **Arguments avec étiquettes (mot-clé)**





- Somme
- Soustraction
- Multiplication
- Division

Calculatrice

```
def somme( A , B ):
```

```
    C = A + B
```

```
    print ( " A + B = " , C )
```

```
def soustraction( A , B ):
```

```
    C = A - B
```

```
    print ( " A - B = " , C )
```

```
def multiplication( A , B ):
```

```
    C = A * B
```

```
    print ( " A * B = " , C )
```

```
def division( A , B ):
```

```
    if B != 0 :
```

```
        print ( " A / B = " , A / B )
```

```
    else :
```

```
        print ( " Impossible " )
```

Calculatrice

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )
```

```
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A / B )  
    else :  
        print ( " Impossible " )  
  
somme( 50 , 10 )  
soustraction( 15 , 35 )  
multiplication( 7 , 3 )  
division( 4 , 2 )
```

- Somme
- Soustraction
- Multiplication
- Division

```
def somme(A, B):
    C = A + B
    print("A+B=" , C)

def soustraction(A, B):
    C = A - B
    print("A-B=" , C)

def multiplication(A, B):
    C = A * B
    print("A*B=" , C)

def division(A, B):
    C = A / B
    print("A/B=" , C)
```

Calculatrice

```
def division(A, B):
    if B == 0:
        print("impossible")
    else:
        print("A/B=" , A / B)

somme(50, 10)
soustraction(15, 35)
multiplication(3, 2)
division(4, 2)
```

- Somme
- Soustraction
- Multiplication
- Division
- Puissance
- Racine carré
- Factoriel
- Exponentielle
- ...

Calculatrice

```
def somme(A, B):  
    print("Somme = A + B")  
  
def soustraction(A, B):  
    print("Soustraction = A - B")  
  
def multiplication(A, B):  
    print("Multiplication = A * B")  
  
def division(A, B):  
    if B == 0:  
        print("Division impossible")  
    else:  
        print("Division = A / B")  
  
somme(50, 10)  
soustraction(15, 35)  
multiplication(3, 2)  
division(10, 2)
```

- Somme
- Soustraction
- Multiplication
- Division
- Puissance
- Racine carré
- Factoriel
- Exponentielle
- ...

Calculatrice

```
def somme(A, B):  
    print("Somme : ", A + B)  
  
def soustraction(A, B):  
    print("Soustraction : ", A - B)  
  
def multiplication(A, B):  
    print("Multiplication : ", A * B)  
  
def division(A, B):  
    if B == 0:  
        print("Division impossible")  
    else:  
        print("Division : ", A / B)
```



Complexité

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )  
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A /  
B )  
    else :  
        print ( " Impossible " )  
somme( 50 , 10 )  
soustraction( 15 , 35 )  
multiplication( 7 , 3 )  
division( 4 , 2 )
```

Calculatrice.py

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )  
somme( 9 , 150 )  
soustraction( 88 , 2 )  
multiplication( 52 , 21 )  
somme( 69 , 5 )  
soustraction( 44 , 41 )  
multiplication( 125 , 32 )  
somme( 9 , 150 )
```

operations.py

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )  
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A /  
B )  
    else :  
        print ( " Impossible " )  
somme( 50 , 10 )  
soustraction( 15 , 35 )  
multiplication( 7 , 3 )  
division( 4 , 2 )
```

Calculatrice.py

```
def multiplication( A , B ):
    C = A * B
    print ( " A * B = " , C )
def division( A , B ):
    if B != 0 :
        print ( " A / B = " , A / B )
    else :
        print ( " Impossible " )
multiplication( 14 , 31 )
division( 100 , 24 )
```

calculs.py

```
def somme( A , B ):
    C = A + B
    print ( " A + B = " , C )
def soustraction( A , B ):
    C = A - B
    print ( " A - B = " , C )
def multiplication( A , B ):
    C = A * B
    print ( " A * B = " , C )
somme( 9 , 150 )
soustraction( 88 , 2 )
multiplication( 52 , 21 )
somme( 69 , 5 )
soustraction( 44 , 41 )
multiplication( 125 , 32 )
somme( 9 , 150 )
```

operations.py

```
def somme( A , B ):
    C = A + B
    print ( " A + B = " , C )
def soustraction( A , B ):
    C = A - B
    print ( " A - B = " , C )
def multiplication( A , B ):
    C = A * B
    print ( " A * B = " , C )
def division( A , B ):
    if B != 0 :
        print ( " A / B = " , A /
B )
    else :
        print ( " Impossible " )
somme( 50 , 10 )
soustraction( 15 , 35 )
multiplication( 7 , 3 )
division( 4 , 2 )
```

Calculatrice.py

- Somme
 - Soustraction
 - Multiplication
 - Division
 - Puissance
 - Racine carré
 - Factoriel
 - Exponentielle
 - ... `calculs.py`
- Programme 1
 - Programme 2
 - Programme 3
 - Programme 4
 - Programme 5
 - ...



Complexité

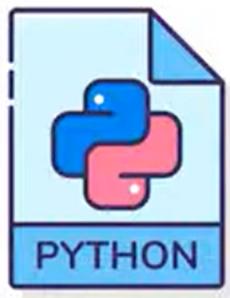


Duplication

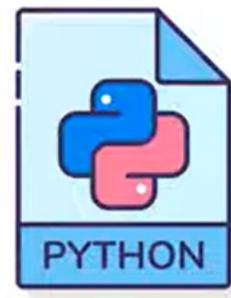
Calculatrice

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )
```

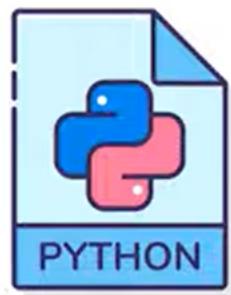
```
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A / B )  
    else :  
        print ( " Impossible " )  
  
somme( 50 , 10 )  
soustraction( 15 , 35 )  
multiplication( 7 , 3 )  
division( 4 , 2 )
```



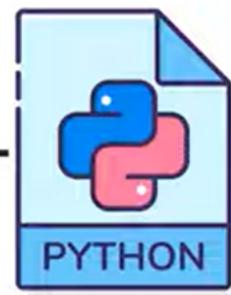
Calculatrice.py



fonctions.py



Calculatrice.py



fonctions.py



Calculatrice

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )  
  
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A / B )  
    else :  
        print ( " Impossible " )
```

fonctions.py

Calculatrice

somme(50 , 10)

soustraction(15 , 35)

multiplication(7 , 3)

division(4 , 2)

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )  
  
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A / B )  
    else :  
        print ( " Impossible " )
```

Calculatrice.py

fonctions.py

Calculatrice

```
from fonctions import *
somme( 50 , 10 )
soustraction( 15 , 35 )
multiplication( 7 , 3 )
division( 4 , 2 )
```

Calculatrice.py

```
def somme( A , B ):
    C = A + B
    print( " A + B = " , C )
def soustraction( A , B ):
    C = A - B
    print( " A - B = " , C )
def multiplication( A , B ):
    C = A * B
    print( " A * B = " , C )
def division( A , B ):
    if B != 0 :
        print( " A / B = " , A / B )
    else :
        print( " Impossible " )
```

fonctions.py



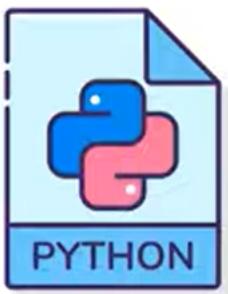
Simple



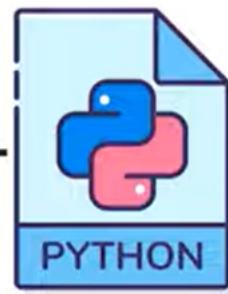
Complexité

Calculatrice.py

fonctions.py

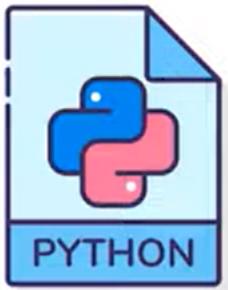


calculatrice.py

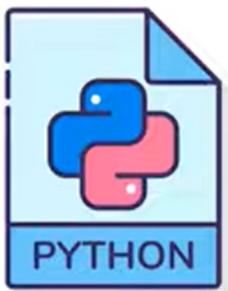


fonctions.py

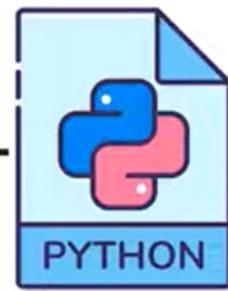




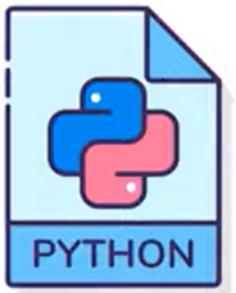
operations.py



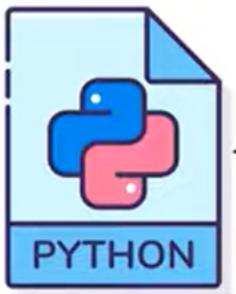
calculatrice.py



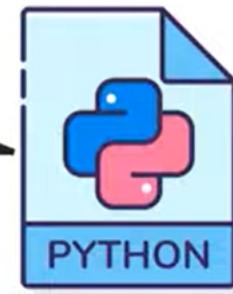
fonctions.py



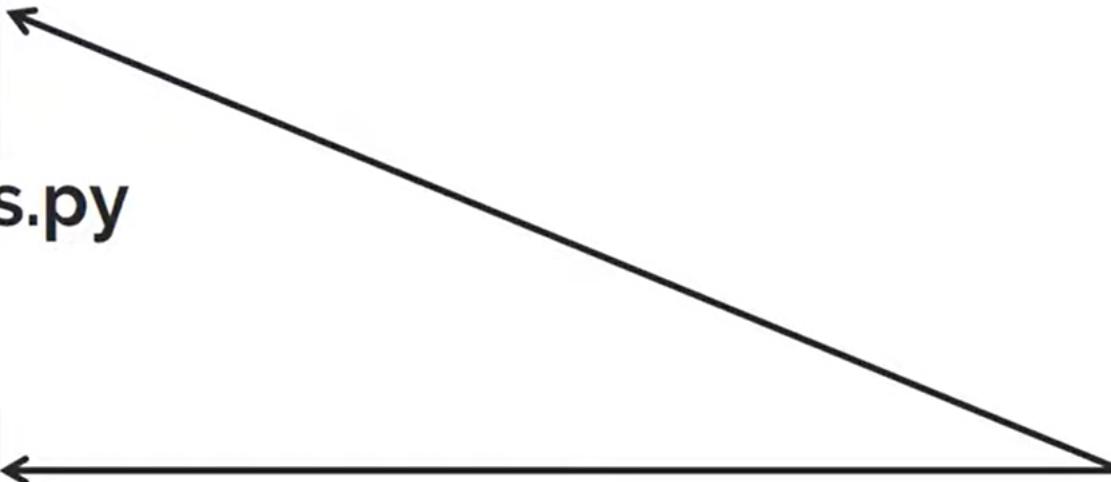
operations.py

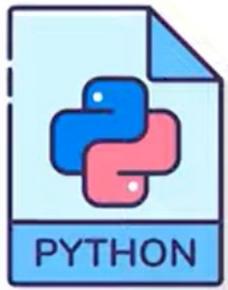


calculatrice.py

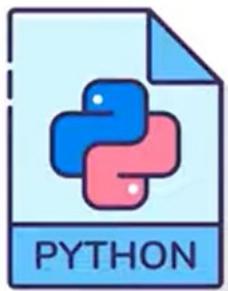


fonctions.py

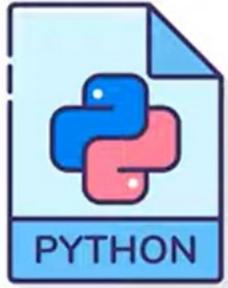




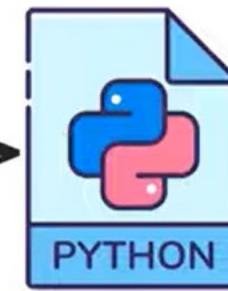
operations.py



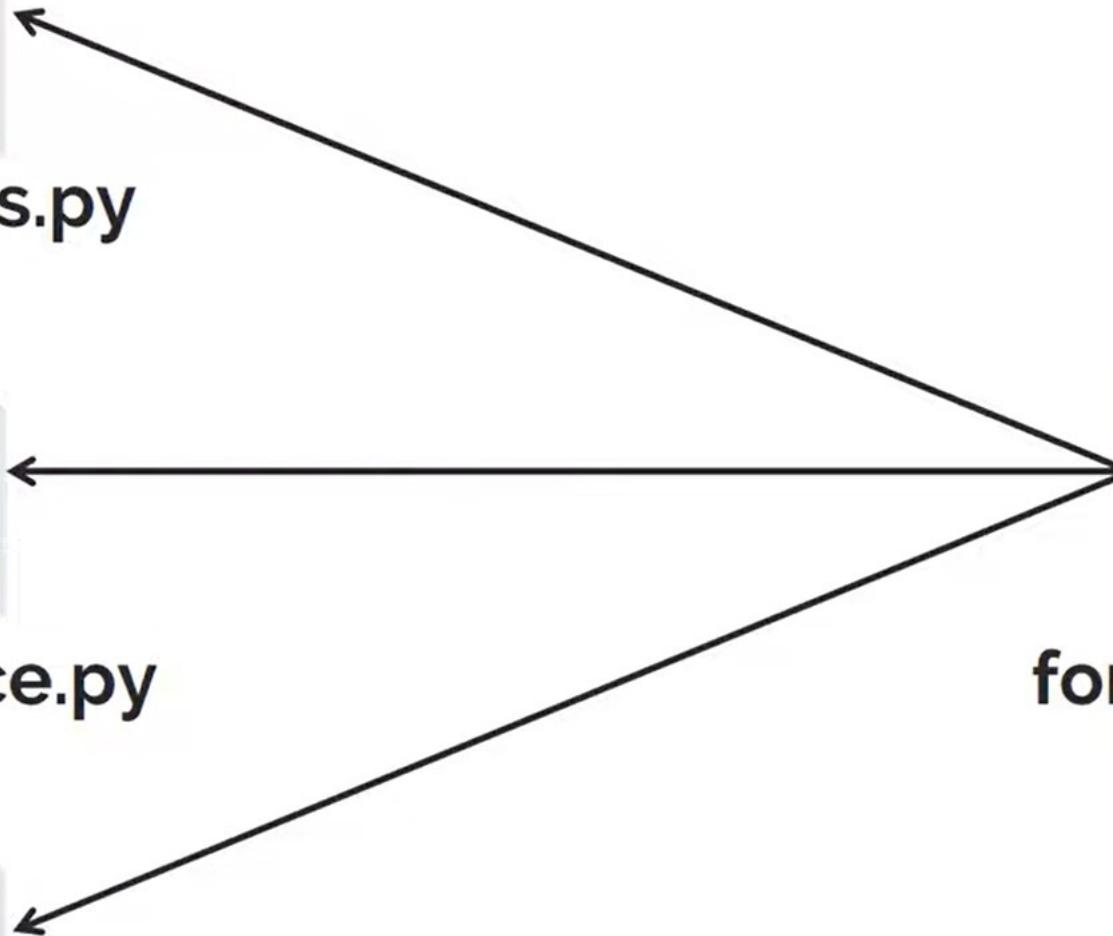
calculatrice.py



calculs.py



fonctions.py



```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )  
  
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A / B )  
    else :  
        print ( " Impossible " )
```

fonctions.py

```
from fonctions import *
somme( 50 , 10 )
soustraction( 15 , 35 )
multiplication( 7 , 3 )
division( 4 , 2 )
```

calculatrice.py

```
from fonctions import *
somme( 9 , 150 )
soustraction( 88 , 2 )
multiplication( 52 , 21 )
division( 43 , 22 )
somme( 69 , 5 )
soustraction( 44 , 41 )
multiplication( 125 , 32 )
division( 42 , 15 )
somme( 9 , 150 )
```

operations.py

```
def somme( A , B ):
    C = A + B
    print( " A + B = " , C )
def soustraction( A , B ):
    C = A - B
    print( " A - B = " , C )
def multiplication( A , B ):
    C = A * B
    print( " A * B = " , C )
def division( A , B ):
    if B != 0 :
        print( " A / B = " , A / B )
    else :
        print( " Impossible " )
```

fonctions.py

```
from fonctions import *
somme( 50 , 10 )
soustraction( 15 , 35 )
multiplication( 7 , 3 )
division( 4 , 2 )
```

calculatrice.py

```
from fonctions import *
multiplication( 14 , 31 )
division( 100 , 24 )
```

calculs.py

```
from fonctions import *
somme( 9 , 150 )
soustraction( 88 , 2 )
multiplication( 52 , 21 )
division( 43 , 22 )
somme( 69 , 5 )
soustraction( 44 , 41 )
multiplication( 125 , 32 )
division( 42 , 15 )
somme( 9 , 150 )
```

operations.py

```
def somme( A , B ):
    C = A + B
    print( "A + B = " , C )
def soustraction( A , B ):
    C = A - B
    print( "A - B = " , C )
def multiplication( A , B ):
    C = A * B
    print( "A * B = " , C )
def division( A , B ):
    if B != 0 :
        print( "A / B = " , A / B )
    else :
        print( "Impossible " )
```

fonctions.py



complexité



Simple



Duplication



Complexité



Duplication



Simple



Réutilisable

Modules

- Les modules sont utilisés pour décomposer les grands programmes en petits fichiers gérables et organisés. De plus, les modules permettent la réutilisation du code.

- Les modules sont utilisés pour décomposer les grands programmes en petits fichiers gérables et organisés. De plus, les modules permettent la réutilisation du code.
- Nous pouvons définir nos fonctions les plus utilisées dans un module et l'importer, au lieu de copier leurs définitions dans différents programmes.

Importation

Exemple :

Méthode 1 : Importer toutes les fonctions

Exemple :

```
from fonctions import *

somme( 50 , 10 )

soustraction( 15 , 35 )

multiplication( 7 , 3 )

division( 4 , 2 )
```

calculatrice.py

```
def somme( A , B ):

    C = A + B

    print ( " A + B = " , C )

def soustraction( A , B ):

    C = A - B

    print ( " A - B = " , C )

def multiplication( A , B ):

    C = A * B

    print ( " A * B = " , C )

def division( A , B ):

    if B != 0 :

        print ( " A / B = " , A / B )

    else :

        print ( " Impossible " )
```

fonctions.py

*Note : L'importation de toutes les fonctions avec * est déconseillée.*

Importation

Méthode 2 : Importer une ou plusieurs fonctions

Exemple :

calculatrice.py

```
def somme(A, B):  
    C = A + B  
    print("A + B =", C)  
  
def soustraction(A, B):  
    C = A - B  
    print("A - B =", C)  
  
def multiplication(A, B):  
    C = A * B  
    print("A * B =", C)  
  
def division(A, B):  
    if B != 0:  
        print("A / B =", A / B)  
    else:  
        print("Impossible")
```

fonctions.py

Méthode 2 : Importer une ou plusieurs fonctions

Exemple :

```
from fonctions import somme , division  
  
somme( 50 , 10 )  
  
division( 4 , 2 )
```

calculatrice.py

```
def somme( A , B ):  
    C = A + B  
    print( " A + B = " , C )  
  
def soustraction( A , B ):  
    C = A - B  
    print( " A - B = " , C )  
  
def multiplication( A , B ):  
    C = A * B  
    print( " A * B = " , C )  
  
def division( A , B ):  
    if B != 0:  
        print( " A / B = " , A / B )  
    else:  
        print( " Impossible " )
```

fonctions.py

Importation

Méthode 3 : Importer le module

Exemple :

```
import fonctions  
  
fonctions.somme( 50 , 10 )  
  
fonctions.soustraction( 15 , 35 )  
  
fonctions.multiplication( 7 , 3 )  
  
fonctions.division( 4 , 2 )
```

calculatrice.py

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )  
  
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A / B )  
    else :  
        print ( " Impossible " )
```

fonctions.py

Note : N'oubliez pas de spécifier le nom du module devant la fonction.

Importation

Méthode 4 : Importer un module et donnez-lui un alias.

Exemple : **import fonctions as f**

calculatrice.py

```
def somme(A, B):  
    C = A + B  
    print("A + B =", C)  
  
def soustraction(A, B):  
    C = A - B  
    print("A - B =", C)  
  
def multiplication(A, B):  
    C = A * B  
    print("A * B =", C)  
  
def division(A, B):  
    if B != 0:  
        print("A / B =", A / B)  
    else:  
        print("Impossible")
```

fonctions.py

Importation

Méthode 4 : Importer un module et donnez-lui un alias.

Exemple :

```
import fonctions as f  
  
f.somme( 50 , 10 )  
  
f.soustraction( 15 , 35 )  
  
f.multiplication( 7 , 3 )
```

calculatrice.py

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )  
  
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A / B )  
    else :  
        print ( " Impossible " )
```

fonctions.py

Importation

Méthode 4 : Importer un module et donnez-lui un alias.

Exemple :

```
import fonctions as f  
  
f.somme( 50 , 10 )  
  
f.soustraction( 15 , 35 )  
  
f.multiplication( 7 , 3 )  
  
f.division( 4 , 2 )
```

calculatrice.py

```
def somme( A , B ):  
    C = A + B  
    print ( " A + B = " , C )  
  
def soustraction( A , B ):  
    C = A - B  
    print ( " A - B = " , C )  
  
def multiplication( A , B ):  
    C = A * B  
    print ( " A * B = " , C )  
  
def division( A , B ):  
    if B != 0 :  
        print ( " A / B = " , A / B )  
    else :  
        print ( " Impossible " )
```

fonctions.py

Note : N'oubliez pas de spécifier le nom de l'alias devant la fonction.

Importation

Méthode 5 : Importer une fonction depuis un module et donnez-lui un alias.

Exemple : `from fonctions import somme as s`

calculatrice.py

```
def somme(A, B):  
    C = A + B  
    print("A + B =", C)  
  
def soustraction(A, B):  
    C = A - B  
    print("A - B =", C)  
  
def multiplication(A, B):  
    C = A * B  
    print("A * B =", C)  
  
def division(A, B):  
    if B != 0:  
        print("A / B =", A / B)  
    else:  
        print("Impossible")
```

fonctions.py

Importation

Méthode 5 : Importer une fonction depuis un module et donnez-lui un alias.

Exemple :

```
from fonctions import somme as s
s(50, 10)
```

calculatrice.py

```
def somme(A, B):
    C = A + B
    print("A + B =", C)

def soustraction(A, B):
    C = A - B
    print("A - B =", C)

def multiplication(A, B):
    C = A * B
    print("A * B =", C)

def division(A, B):
    if B != 0:
        print("A / B =", A / B)
    else:
        print("Impossible")
```

fonctions.py

Exercice 1

1- Créer un projet (dossier) contenant deux fichiers: fonctions et main.

Le fichier **fonctions** est un module qui contient la définition des fonctions suivantes:

- Une fonction qui retourne si les valeurs de A et B sont de même signe ou non.
- Une fonction qui renvoie le minimum de A et B.
- Une fonction qui renvoie le maximum de A et B.

main.py

fonctions.py X

fonctions.py > signe > ✘

```
1 def signe(nombre1, nombre2):
2     if nombre1 * nombre2 < 0 :
3         print(f"{nombre1} et {nombre2} sont de signe contraire")
4     else:
5         print(f"{nombre1} et {nombre2} sont de meme signe")
6
7 def minimum(A, B):
8     min = A
9     if A > B :
10        min = B
11    print(f"Le minimum entre {A} et {B} est : {min}")
12
13 def maximum(A, B):
14     max = A
15     if B > A :
16        max = B
17    print(f"Le maximum entre {A} et {B} est : {max}")
```

main.py X

fonctions.py

main.py

```
1  from fonctions import*
2  signe(1, -6)
3  minimum(1, -6)
```

Explorer

...

main.py X

fonctions.py

pythonTest

> _pycache_

> codes

> packages

fonctions.py

main.py

main.py

2 signe(1, -6)

3 minimum(1, -6)

4 maximum(1, -6)

Problems

Output

Debug Console

Terminal

Ports

GitLens

Spell Checker 15

Code Reference Log

● PS D:\pythonTest> python main.py

1 et -6 sont de signe contraire

Le minimum entre 1 et -6 est : -6

Le maximum entre 1 et -6 est : 1

○ PS D:\pythonTest> █

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "operations" and the file "operations.py" as the active tab. The left sidebar shows the project structure under "v12": "venv library root" contains "fonctions.py" and "main.py"; "operations.py" is also listed. Below that are "External Libraries" and "Scratches and Consoles". The main editor window contains the following Python code:

```
import fonctions as f
f.signe(55, 66)
f.signe(50, -2)
f.minimum(50, 2)
f.maximum(50, 2)
```

Run: operations

C:\Users\P50\PycharmProjects\v12\venv\Scripts\python.exe C:/Users/P50/PycharmProjects/v12/operations.py

A et B ont le même signe

A et B ont deux signe différents

Le minimum est : 2

Le maximum est : 50

Process finished with exit code 0

SUPER MARIO FAMILY MUSEUM



© Nintendo





niveau_1.py

niveau_2.py

niveau_3.py

image.py

son.py

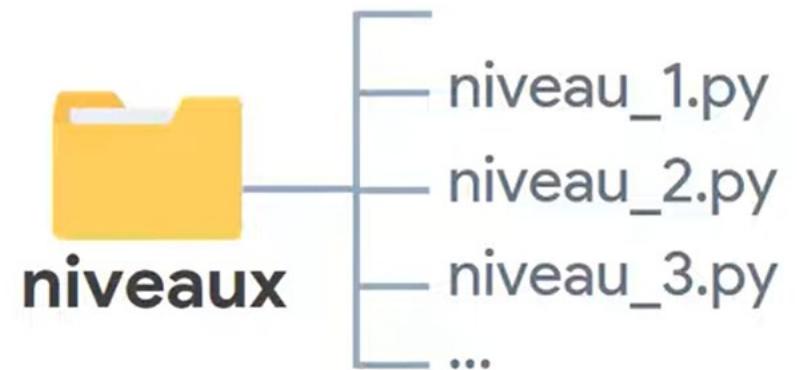
score.py

bonus.py

...

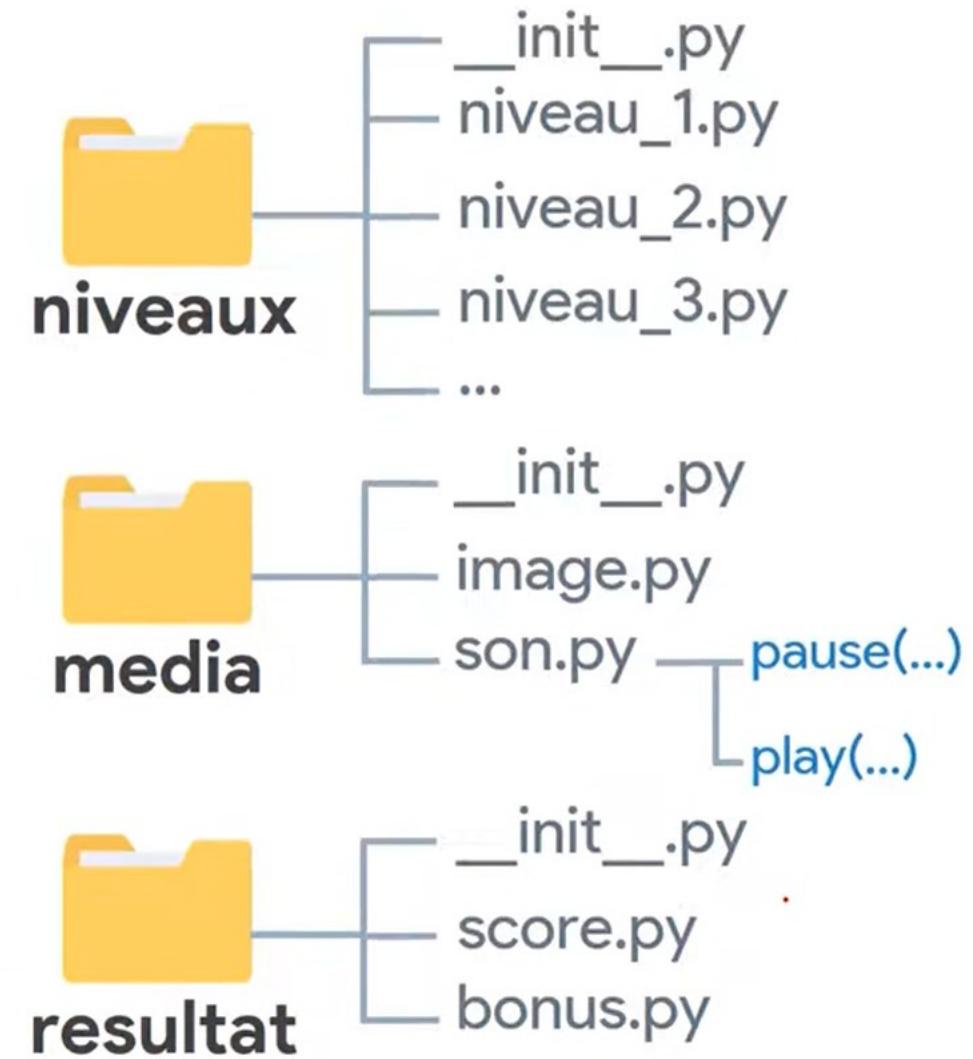
...





Utilisation :

```
from media.son import play  
...  
mario.py
```



Paquet (package)

- Un paquet est un ensemble de plusieurs modules regroupés dans le même dossier.
- Un paquet est importé et utilisé de la même manière que les modules.
- Un paquet doit contenir un fichier nommé `__init__.py` pour que Python le considère comme un paquet . Ce fichier peut simplement être un fichier vide.

Écosystème

Disponibilité des bibliothèques



[Tous](#)[Images](#)[Vidéos](#)[Actualités](#)[Plus](#)[Paramètres](#)[Outils](#)

Recherche dans les pages en Français ▾

Date indifférente ▾

Tous les résultats ▾

Effacer

<https://docs.python.org> › library ▾

[La bibliothèque standard — Documentation Python 3.9.5](#)

La bibliothèque contient des **modules** natifs (écrits en C) exposant les fonctionnalités du système telles que les interactions avec les fichiers qui autrement ne ...

[Importer des modules](#) · [Modules de traitement XML](#) · [Shelve — Objet Python...](#) · [Os](#)

<https://docs.python.org> › tutorial › modules ▾

[6. Modules — Documentation Python 3.9.5](#)

Voir [Modules standards](#) pour plus d'informations. 6.1.3. Fichiers **Python** « compilés ». Pour accélérer le chargement des **modules**, **Python** cache une version ...

<https://www.pierre-giraud.com> › module-paquet ▾

[Les modules et paquets Python - Pierre Giraud](#)

En **Python**, on peut distinguer trois grandes catégories de **module** en les classant selon leur éditeur : Les **modules standards** qui ne font pas partie du langage en ...

<https://informatique-python.readthedocs.io> › Cours › st... ▾

[5. Bibliothèque standard — Documentation Analyse ...](#)

Python dispose d'une très riche bibliothèque de **modules** étendant les capacités du langage dans de nombreux domaines: nouveaux types de données, ...

Autres questions posées

What are standard modules in Python?

Recherches associées

[modules python liste](#)[liste des modules python pdf](#)[python liste modules installés](#)[python module list](#)

Table des matières

Fonctions mathématiques — `math`

- Fonctions arithmétiques et de représentation
- Fonctions logarithme et exponentielle
- Fonctions trigonométriques
- Conversion angulaire
- Fonctions hyperboliques
- Fonctions spéciales
- Constantes

Sujet précédent

`numbers` — Classes de base abstraites numériques

Sujet suivant

Fonctions mathématiques pour nombres complexes — `cmath`

Cette page

Signalement de bogue

Montrer le code source

Fonctions mathématiques — `math`

Ce module fournit l'accès aux fonctions mathématiques définies par la norme C.

Ces fonctions ne peuvent pas être utilisées avec les nombres complexes ; si vous avez besoin de la prise en charge des nombres complexes, utilisez les fonctions du même nom du module `cmath`. La séparation entre les fonctions qui gèrent les nombres complexes et les autres vient du constat que tous les utilisateurs ne souhaitent pas acquérir le niveau mathématique nécessaire à la compréhension des nombres complexes. Recevoir une exception plutôt qu'un nombre complexe en retour d'une fonction permet au programmeur de déterminer immédiatement comment et pourquoi ce nombre a été généré, avant que celui-ci ne soit passé involontairement en paramètre d'une autre fonction.

Les fonctions suivantes sont fournies dans ce module. Sauf mention contraire explicite, toutes les valeurs de retour sont des flottants.

Fonctions arithmétiques et de représentation

`math.ceil(x)`

Renvoie la partie entière par excès de x , le plus petit entier supérieur ou égal à x . Si x est un flottant, délégué à `x.__ceil__()`, qui doit renvoyer une valeur `Integral`.

`math.comb(n, k)`

Renvoie le nombre de façons de choisir k éléments parmi n de manière non-ordonnée et sans répétition.

Vaut $n! / (k! * (n - k)!)$ quand $k \leq n$ et zéro quand $k > n$.

Aussi connue sous le nom de « coefficient binomial » car c'est la valeur du coefficient du k^{e} terme dans le développement polynomial de l'expression `(1+x) ** n`.

Lève une `TypeError` si un des paramètres n'est pas un entier. Lève une `ValueError` si un des paramètres est négatif.

Nouveau dans la version 3.8.

`math.copysign(x, y)`

```
1 import math
2
3 math.|
```

f isinf(_x) math
f isnan(_x) math
f isqrt(_n) math
f lcm(integers) math
f ldexp(_x, _i) math
f lgamma(_x) math
f log(x, base) math
f log1p(_x) math
f log2(_x) math
f log10(_x) math
v nan math

Press Ctrl+ to choose the selected (or first) suggestion and insert a dot afterwards. Next Tip : :

Exercice 2

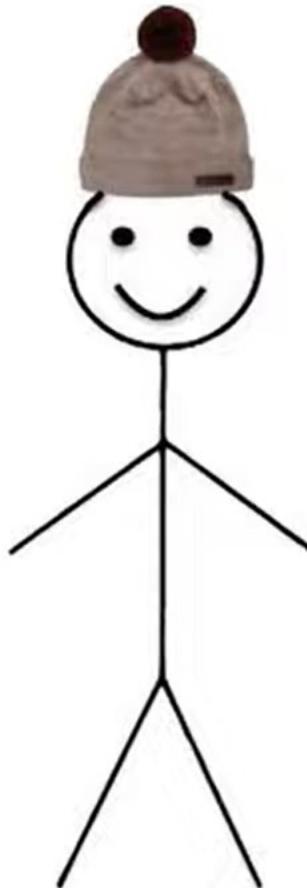
En utilisant les modules standards de python, écrire les programmes suivants :

- Un programme qui calcule la factorielle d'un nombre,
- Un programme qui affiche un nombre impair aléatoire entre 1 et 101,
- Un programme qui détermine le mode et l'écart type d'une série statistique,
- Un programme qui affiche la date d'aujourd'hui,
- Un programme qui ouvre le site Google.com,
- Un programme qui convertit des bitcoins en euro,

This is Bill

**Bill doesn't read
Documentation**

Don't be like Bill



Examples

Basic examples:

```
>>> random()                                     # Random float: 0.0 <= x < 1.0
0.37444887175646646

>>> uniform(2.5, 10.0)                         # Random float: 2.5 <= x <= 10.0
3.1800146073117523

>>> expovariate(1 / 5)                          # Interval between arrivals averaging 5 seconds
5.148957571865031

>>> randrange(10)                               # Integer from 0 to 9 inclusive
7

<<
>>> randrange(0, 101, 2)                         # Even integer from 0 to 100 inclusive
26

>>> choice(['win', 'lose', 'draw'])              # Single random element from a sequence
'draw'

>>> deck = 'ace two three four'.split()
>>> shuffle(deck)                                # Shuffle a list
>>> deck
['four', 'two', 'ace', 'three']

>>> sample([10, 20, 30, 40, 50], k=4)          # Four samples without replacement
[40, 10, 50, 30]
```

Les codes de formatage se référant aux heures, minutes ou secondes auront pour valeur 0. Pour une liste complète des directives de formatage, voir [Comportement de strftime\(\) et strptime\(\)](#).

date.__format__(format)

Identique à `date.strftime()`. Cela permet de spécifier une chaîne de formatage pour un objet `date` dans une chaîne de formatage littérale et à l'utilisation de `str.format()`. Pour une liste complète des directives de formatage, voir [Comportement de strftime\(\) et strptime\(\)](#).

Exemple d'utilisation de la classe date :

Exemple de décompte des jours avant un évènement :

```
>>> import time
>>> from datetime import date
>>> today = date.today()
>>> today
datetime.date(2007, 12, 5)
>>> today == date.fromtimestamp(time.time())
True
<<
>>> my_birthday = date(today.year, 6, 24)
>>> if my_birthday < today:
...     my_birthday = my_birthday.replace(year=today.year + 1)
>>> my_birthday
datetime.date(2008, 6, 24)
>>> time_to_birthday = abs(my_birthday - today)
>>> time_to_birthday.days
202
```

Plus d'exemples avec la classe date :

```
>>> from datetime import date
>>> d = date.fromordinal(730920) # 730920th day after 1. 1. 0001
>>> d
datetime.date(2002, 3, 11)

>>> # Methods related to formatting string output
>>> d.isoformat()
'2002-03-11'
>>> d.strftime("%d/%m/%y")
'11/03/02'
>>> d.strftime("%A %d %B %Y")
```

You can convert amount from one currency to other.

Currency Symbols

Currency names

Lets now look at some examples

Get conversion rate

We will get the conversion rate from USD to Euros, Pounds, Canadian dollars and Japanese Yen

```
from forex_python.converter import CurrencyRates

curr = CurrencyRates()
print (curr.get_rate('USD', 'EUR') )
print (curr.get_rate('USD', 'GBP') )
print (curr.get_rate('USD', 'CAD') )
print (curr.get_rate('USD', 'JPY') )
```

When run you will something like this, the values will be different

```
C:\Python38-32>python curr1.py
0.9005763689
0.8021433718
1.3858969741
107.2676512968
```

conversion output

Basic conversions

