BAC Pro CIEL	Python – Module 5 Les fonctions	Année 2025/2026

A. Fonctions

Une fonction est un bloc de code organisé et réutilisable permettant d'effectuer une action précise. Les fonctions permettent de décomposer des problèmes complexes en petits morceaux gérables, d'éviter la répétition de code et de rendre les programmes plus lisibles et faciles à maintenir.

Pour définir une fonction en Python, on utilise le mot-clé def, suivi du nom de la fonction, des parenthèses qui peuvent contenir des paramètres et des deux points. Le corps de la fonction doit être indenté.

La syntaxe est la suivante :

```
def nom_de_la_fonction(parametre1, parametre2,…) :
    # Corps de la fonction
    # Instructions à exécuter lorsque la fonction est appelée
    return valeur_a_retourner
```

avec :

- ⇨ def : mot-clé pour définir la fonction
- ⇨ nom_de_la_fonction : un nom descriptif
- ⇨ ( ) : contient les paramètres (arguments) que la fonction peut recevoir (optionnel)
- ⇨ : : Marque le début du bloc de code de la fonction
- ⇨ return : Utilisé pour renvoyer une valeur de la fonction. Etant un paramètre optionnel, il n'est pas obligatoire de le mettre. Si la fonction ne possède pas de return, la fonction ne renverra rien (None)

Exemples :

Sans paramètre et sans retour

```
def saluer() :
      print("Bonjour tout le monde")
```

Lorsque "saluer()", sera appelé dans un programme, elle affichera le message « Bonjour tout le monde ».

Exemple d'utilisation de la fonction saluer() :

```
saluer()
```

Résultat :

```
Bonjour tout le monde
```

Avec paramètre et sans retour

```
def saluer_personne(nom) :
      print(f"Bonjour, {nom} !")
```

Exemple d'utilisation de la fonction saluer_personne :

```
saluer_personne("Jack")
saluer_personne("Emilie")
```

Résultat :

```
Bonjour, Jack !
Bonjour, Emilie !
```

Avec plusieurs paramètres et retour

```
def addition(a,b) :
    somme = a + b
    return somme
```

Exemple d'utilisation de la fonction addition(a,b) :

```
resultat = addition(5,8)
print(f"La somme est {resultat}")
print(f"La somme suivante est {addition(10,4)}")
```

Résultat :

```
La somme est 13
La somme suivante est 14
```

Avec plusieurs paramètres et retour de plusieurs valeurs

```
def operations(x,y) :
    somme = x + y
    produit = x * y
    return somme, produit
```

Exemple d'utilisation de la fonction operations(x,y) :

```
s, p = operations(4,3)
print(f"Somme : {s}, Produit : {p}")
```

Résultat :

```
Somme : 7, Produit : 12
```

Avec plusieurs paramètres dont un par défaut et sans retour

```
def saluer_avec_langue(nom, langue="français") :
      if langue == "français" :
            print(f"Bonjour, {nom} !")
      elif langue == "anglais" :
            print(f"Hello, {nom} !")
```

Exemples d'utilisation de la fonction saluer_avec_langue(nom, langue="français") :

```
saluer_avec_langue("Alice")
saluer_avec_langue("John","anglais")
```

Résultat :

```
Bonjour, Alice !
Hello, John !
```

B. Commentaires de fonctions

Les commentaires de fonctions, appelés aussi doctrings, sont des chaînes de caractères sur plusieurs lignes placées comme première instruction à l'intérieur de modules, fonctions ou méthodes. Ils sont utilisés pour documenter l'objet Python qu'ils suivent et sont accessibles via l'attribut __doc__ de l'objet, ainsi que par des outils de documentation (comme Sphinx).

Exemple :

```
def ma_fonction(param1 , param2) :
      """
      Cette fonction fait quelque chose d'incroyable
      Args :
            param1 (int) : le premier paramètre entier
            param2 (str) : le deuxième paramètre est une chaîne de
      caractères

      Returns :
            bool : True si l'opération a réussi, False sinon
      """

      return True
print(ma_fonction.__doc__)
```

Résultat :

```
    Cette fonction fait quelque chose d'incroyable

    Args :

      param1 (int) : le premier paramètre entier

      param2 (str) : le deuxième paramètre est une chaîne de
caractère

    Returns :

      bool : True si l'opération a réussi, False sinon
```

C. Fonction main

Une astuce utile en Python est de déclarer le programme principal comme une fonction.

Cette astuce permet ainsi de fermer le programme via « return ».

Exemple :

```
def main() :
      for i in range (25) :
            print(f"Le nombre est : {i}")
            if i == 8 :
                  return

main()
```

Dans cet exemple, la fonction principale main() va s'exécuter de 0 à 8 mais elle s'arrêtera à 8 grâce à l'instruction return.

La fonction main() a été définie et elle sera notre programme principal. La dernière ligne de ce programme (main()) appelle cette fonction faisant office de programme principal.

D. Erreurs fréquentes

Oublier les deux-points :

```
def saluer()
    print("Bonjour")
```

```
SyntaxError: invalid syntax
```

Mauvaise indentation :

```
def saluer() :
print("Bonjour")
```

```
IndentationError: expected an indented block
```

Oublier d'appeler la fonction :

```
def saluer() :
    print("Bonjour")
```

Résultat : rien ne s'affiche tant que saluer() n'est pas appelé

Confondre print() et return :

```
def addition(a, b):
    print(a + b)

resultat = addition(2, 3)
print(resultat)
```

Résultat :

```
5
None
```

E. A retenir

def permet de définir des fonctions

une fonction doit être appelée pour s'exécuter

les paramètres permettent d'envoyer des informations à la fonction

return permet de renvoyer une valeur

une fonction sans return renvoie None

l'indentation est obligatoire

F. Synthèse