



- ▶ Présentation générale
- ▶ QUESTIONNAIRE : VOS ATTENTES ET MOTIVATIONS
- ▶ Semaine 1. Introduction au MOOC et aux outils Python
- ▶ Semaine 2. Notions de base pour écrire son premier programme en Python
- ▶ Semaine 3. Renforcement des notions de base, références partagées
- ▼ **Semaine 4. Fonctions et portée des variables**

**1. Fonctions**

Quiz Echéance le janv 25, 2018 at 23:30 UTC

**2. Tests if/elif/else et opérateurs booléens**

Quiz Echéance le janv 25, 2018 at 23:59 UTC

**3. Boucles while**

## QUIZ 24 - PASSAGE D'ARGUMENTS ET APPEL DE FONCTIONS

(1/3 points)

### Passage d'arguments par défaut

Quelles sont les manières correctes de déclarer un argument `var` par défaut dans une fonction.

#### Proposition 1

```
def f(a, b, var == 10):  
    print(a, b, var)
```

#### Proposition 2

```
def f(a, b, var=10):  
    print(a, b, var)
```

#### Proposition 3

```
def f(a, var=10, b):  
    print(a, b, var)
```

#### Proposition 4

```
def f(a, var=10, b=30):  
    print(a, b, var)
```

#### Proposition 5

```
def f(a, b, var, var=10):  
    print(a, b, var)
```

Choisissez une ou plusieurs propositions.

☐ Proposition 1

☒ Proposition 2 ✓

☒ Proposition 3



## variables - règle LEGB

Quiz Échéance le janv 25, 2018 at 23:59 UTC

## 5. Modification de la portée avec global et nonlocal

Quiz Échéance le janv 25, 2018 at 23:59 UTC

## 6. Passage d'arguments et appel de fonctions

Quiz Échéance le janv 25, 2018 at 23:59 UTC

- ▶ Semaine 5. Itération, importation et espace de nommage
- ▶ Semaine 6. Conception des classes
- ▶ Semaine 7. L'écosystème data science Python
- ▶ Semaine 8. Programmation asynchrone - asyncio

☐ Proposition 5



### EXPLANATION

La proposition 1 est syntaxiquement incorrecte: on doit déclarer un argument par défaut avec le signe `=` et non `==`.

La proposition 2 est correcte.

La proposition 3 est incorrecte, toujours apparaitre **après** les arguments ordinaires.

La proposition 4 est correcte puisque dans ce cas, `b` est également un argument par défaut.

La proposition 5 est incorrecte, on ne peut pas dupliquer un argument dans l'entête d'une fonction, ici `var` est dupliqué.

Rechercher un cours



## Unpacking des arguments

Comment passer les éléments de la liste suivante

`L = [1, 2, 'a']`

comme arguments de la fonction

```
def f(a, b, c):
    print(a, b, c)
```

Choisissez une ou plusieurs propositions.

☐ `f(L)`

☒ `f(L[0], L[1], L[2])` ✓

☒ `f(*L)` ✓



### EXPLANATION

La proposition 1 est incorrecte parce que `L` correspond à un seul argument alors que `f` attend 3 arguments.

La proposition 2 est correcte, mais pas pythonique. On passe en effet trois arguments à `f`, par contre, on découpe la liste `L` à la main, ce qui est presque toujours le signe d'une mauvaise utilisation de python.

La proposition 3 est correcte et pythonique. On utilise la notion de *unpacking* de liste pour automatiquement passer chaque élément de la liste à un argument de la fonction.

## Arguments variables

```
def f(*args):  
    print(args)  
L = [1, 2, 'a']
```

Sélectionnez tous les appels qui sont valides

☐ `f(1, 2)` ✓

☐ `f('a', 4, 5, 6, 7, [1,2])` ✓

☒ `f(L)` ✓

☒ `f(*L)` ✓



### EXPLANATION

Tous ces appels sont valides, mais par contre, il faut faire attention à bien comprendre leur signification. La fonction `f` accepte un nombre variable d'arguments et va mettre tous ces arguments dans un tuple référencé par la variable `args`.



le dernier appel (1, 2, 3, 4, 5)

*Vous avez utilisé 3 essais sur 3*

[A propos](#)

[Aide](#)

[Contact](#)

[Conditions générales d'utilisation](#)

[Charte utilisateurs](#)

[Politique de confidentialité](#)

[Mentions légales](#)



POWERED BY  
OPENedX