



- ▶ 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**
 - 1. Les fichiers**
Quiz Échéance le janv 25, 2018 at 23:30 UTC
 - 2. Les tuples**
Quiz Échéance le janv 25, 2018 at 23:30 UTC
 - 3. Tables de hash**
Quiz Échéance le janv 25, 2018 at 23:59 UTC
 - 4. Les dictionnaires**
Quiz Échéance le janv 25, 2018 at 23:59 UTC
 - 5. Les ensembles**

QUIZ 12 - LES TUPLES (4 points possibles)

Tuples (1)

On se donne en entrée

```
triple = (1, 2, 3,)
```

Parmi les expressions et instructions ci-dessous, lesquelles sont valides ?

☐ `triple[0]` ✓

☒ `triple[:]` ✓

☐ `triple[len(triple)]`

☐ `triple[0] = 0`

✗

EXPLANATION

La première expression est correcte et renvoie 1

La seconde expression est correcte et renvoie une copie du tuple

La troisième expression n'est pas valable et renvoie une exception *IndexError*, car `triple` n'a pas de case numéro 3 (*index out of range*)

L'affectation de la quatrième réponse n'est pas autorisée, car un tuple est un objet immuable

Tuples (2)

Quelles sont les expressions qui renvoient `True` ?



6. Les exceptions

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

7. Les références

partagées

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

8. Introduction aux classes

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

- ▶ Semaine 4.
Fonctions et
portée des
variables
- ▶ 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

☒ (1,) == (1)

☐ [(1,)] [0][0] == 1

☒ [(1),] [0][0] == 1

Rechercher un cours



EXPLANATION

Dans la première expression, l'absence de virgule rend les parenthèses vides de sens, cela revient à comparer 1 et 1

Dans la seconde expression la partie droite n'est pas un tuple, mais un simple entier à cause de l'absence de virgule

Dans la troisième expression, on a une liste contenant un tuple contenant l'entier, l'expression renvoie `True`

Dans la quatrième expression, la virgule n'est pas correctement placée et la parenthèse ne crée pas un tuple.

Unpacking (1)

On pose

```
quadruple = (1, [2, 3], True, [ (4,) ] )
```

Quelles sont parmi les affectations suivantes celles qui sont valables, et qui affectent 4 à `four` ?

☐ (one, (two, three), ignored, ((four))) = quadruple

☒ (one, (two, three,), _, ((four,),)) = quadruple

☐ ((one,), (two, three), _, [[four]]) = quadruple



EXPLANATION

Dans la première forme, `four` ne se trouve pas dans un tuple à cause de l'absence de virgule; du coup `four` va valoir `[(4,)]` et non pas `4`

La seconde forme est correcte

Dans la troisième forme, `one` est inclus dans un tuple, ce qui empêche l'affectation de fonctionner

La dernière forme est correcte, même si les parties droite et gauche de l'affectation mélangent listes et tuples.

Unpacking (2)

On cherche à écrire un code qui permette d'intervertir les deux derniers éléments dans une liste. On suppose que la liste en entrée a au moins deux éléments. Quelles sont parmi les variantes suivantes celles qui font bien ce qu'on veut ?

☐ `tmp = liste[-1]; liste[-1] = liste[-2]; liste[-2] = tmp`



☐ `liste.reverse(-2, -1)`

☒ `liste[-2], liste[-1] = liste[-1], liste[-2]`



EXPLANATION

La première formule fonctionne, quoi que pas très "pythonique"

La seconde formule est une invention, `list.reverse()` n'accepte pas d'argument, comme le montrerait `help(list.reverse)`



... , pour retourner des valeurs.

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