



Accueil > Cours > Apprenez à programmer en Python > Quiz : Maîtriser les règles de base du langage Python

Apprenez à programmer en Python

40 heures  Difficile



Mis à jour le 26/06/2019



Maîtriser les règles de base du langage Python

Bravo ! Vous avez réussi cet exercice !

Compétences évaluées

-  Maîtriser les règles qui définissent la portée des variables
-  Créer et éditer des objets

Question 1

Que renverra l'instruction suivante, partant du principe que la variable `chaîne` contient une chaîne de caractères (classe `str`) ?

```
chaîne[:3]
```

- ☐ Le troisième caractère de la chaîne
- ☐ Le quatrième caractère de la chaîne
- ☒ Les trois premiers caractères de la chaîne

La syntaxe utilisant les deux points (`:`) entre crochets est celle d'une sélection. On peut préciser avant les deux points le début de la sélection et après les deux points la fin de la sélection.

Si certaines de ces informations sont omises, le début de la sélection est automatiquement le début de la chaîne et la fin de la sélection est la fin de la chaîne.

Ainsi, ici, on sélectionne du début de la chaîne jusqu'au caractère d'indice 3 non inclus, c'est-à-dire les trois premiers caractères de la chaîne.

Question 2

Appeler une méthode de la classe `str` (chaîne de caractères) modifie-t-elle la chaîne ?

-  ☐ Oui
-  ☒ Non
- ☐ Cela dépend des méthodes.

Les méthodes de la classe `str`, sans exception, travaillent sur la chaîne et renvoient la chaîne modifiée, sans modifier la chaîne d'origine. Ces objets sont immutables, c'est-à-dire qu'ils ne peuvent pas être modifiés. À la place, Python renvoie une copie de la chaîne sur laquelle les modifications ont été appliquées.

Question 3

Peut-on modifier un caractère d'une chaîne de caractères (variable `chaîne` dans cet exemple) grâce à l'instruction suivante ?


```
chaîne[0] = "A"
```

- ☐ Oui
-  ☒ Non

Les chaînes de caractères ne peuvent être modifiées de cette façon, ce sont des objets immutables. Pour modifier le caractère d'une chaîne, on peut utiliser la sélection ou bien la méthode `replace`, en fonction des cas.

Question 4

Que peut contenir une liste (objet de classe `list`) ?

- ☐ Un nombre indéterminé d'objets du même type
-  ☒ Un nombre indéterminé d'objets sans type particulier

- ☐ Un nombre indéterminé d'objets tant qu'aucun ne se trouve plus d'une fois dans la liste

Une liste peut contenir des objets issus de différentes classes sans inconvénient. De plus, une liste peut contenir des doublons (c'est-à-dire qu'elle peut contenir plusieurs fois le même objet).

Question 5

On peut accéder à un élément d'une liste en utilisant les crochets entourant un entier (un indice). Dans quelle circonstance l'indice menant à un élément est-il modifié ?

- ☐ Jamais, les indices ne changent pas après la création d'une liste.
- ✓ ☒ Si on change l'ordre de la liste.
- ☐ Seulement si on supprime certains éléments contenus dans la liste.

Les indices menant à un élément dans la liste sont modifiés si l'ordre de la liste est modifié. Cela peut arriver quand on supprime un élément, mais cela peut arriver dans bien d'autres cas également (si on ajoute des éléments au début de la liste, si on trie la liste, si on l'inverse...).

Question 6

Que renvoie la plupart des méthodes de liste (classe `list`) comme `append`, `insert` ou `remove` ?

- ☐ L'élément ajouté ou retiré de la liste
- ☐ L'indice de l'élément ajouté ou retiré de la liste
- ☐ La liste modifiée
- ✓ ☒ Rien (`None`)

Ces méthodes, comme la plupart des méthodes de liste, travaillent directement sur la liste initiale et ne renvoient rien (`None`). Les listes, à la différence des tuples ou chaînes de caractères, sont en effet des types mutables, que l'on peut modifier, et la plupart des méthodes de la classe `List` modifient directement l'objet d'origine.

Question 7

Peut-on modifier un élément de liste (variable `liste` dans l'exemple) en utilisant l'instruction suivante ?

```
liste[0] = "quelque chose"
```

- ✓ ☒ Oui
- ☐ Non

Les listes, à la différence des tuples ou chaînes de caractères, acceptent cette syntaxe pour remplacer un élément d'une liste par un autre.

Question 8

Quelle est la différence entre les listes (classe `list`) et les tuples (classe `tuple`) ?

- ✓ ☒ On ne peut pas modifier les tuples une fois créés.
- ☐ Les tuples ne peuvent pas contenir plusieurs types d'éléments.
- ☐ On ne peut pas parcourir des tuples.

Les tuples sont des objets immutables, c'est-à-dire qu'ils ne sont pas modifiés une fois créés. Un tuple créé avec un nombre N d'éléments reste donc avec ce nombre N d'éléments.

Bien sûr, il est possible de créer un nouveau tuple à partir du tuple contenant N éléments en en ajoutant un, mais cela ne peut se faire qu'à la création du tuple. Une fois créé, son contenu ne change pas.

Question 9

Si `ma_liste` contient une liste d'éléments (classe `list`), en appelant la fonction `ma_fonction(*ma_liste)`, combien de paramètres seront envoyés à la fonction ?

- ☐ Un, la liste `ma_liste`
- ☐ Un, le premier élément de `ma_liste`
- ✓ ☒ Tous les éléments contenus dans `ma_liste`

En utilisant cette syntaxe (un astérisque `` avant une séquence dans l'appel à une fonction), les éléments contenus dans la liste sont transmis en tant que paramètres de la fonction. Par exemple, si `ma_liste` contient 3 éléments, alors 3 paramètres sont transmis à la fonction.*

Question 10

Après ces instructions, combien d'éléments contiendra la variable `ma_liste` ?

```
ma_liste = [1, 2, 4, 8, 16, 32, 64]
ma_liste = [n for n in ma_liste if n < 16]
```

- ☐ 3
- ✓ ☒ 4
- ☐ 5
- ☐ 6

Cette compréhension de liste (**list comprehension**) filtre la liste en ne sélectionnant que les éléments strictement inférieurs à 16, c'est-à-dire les quatre premiers. Seuls ces quatre éléments se retrouvent donc dans la liste filtrée.

Question 11

Comment sont stockés les éléments dans un dictionnaire (classe `dict`) ?

- ✓ ☒ Sans aucun ordre
- ☐ Sans ordre sauf si on utilise des clés entières
- ✗ ☐ Dans l'ordre dans lequel on les ajoute

Peu importe le type de clé choisi, les dictionnaires ne conservent pas l'ordre d'ajout des éléments. Ce n'est pas un type ordonné.

Le module `collections` contient la définition d'un dictionnaire ordonné, `OrderedDict`, qui propose la structure d'un dictionnaire en conservant une notion d'ordre.

Question 12

Si `adresses` est un objet dictionnaire (classe `dict`), qu'obtiendra-t-on en entrant l'instruction ci-dessous ?

```
for element in adresses:
```

- ✓ ☒ Les clés du dictionnaire
- ☐ Les valeurs du dictionnaire
- ☐ Les clés et valeurs du dictionnaire par paire (`tuple`)

L'instruction donnée parcourra toutes les clés du dictionnaire. Il existe les méthodes `values` et `items` qui permettent, respectivement, de parcourir les valeurs d'un dictionnaire ou de parcourir ses clés et valeurs par couple.

Question 13

Sachant que la variable *dictionnaire* est un dictionnaire (classe `dict`), quelle est la différence entre les deux instructions suivantes ?

```
del dictionnaire["cle"]  
dictionnaire.pop("cle")
```

- ☐ Si la clé n'existe pas, l'exception levée n'est pas la même.
- ✓ ☒ La méthode `pop` retourne la valeur correspondant à la clé supprimée.
- ☐ La méthode `pop` peut aussi être utilisée pour supprimer une valeur du dictionnaire.

La différence entre le mot-clé `del` et la méthode `pop` est que la seconde retourne la valeur correspondant à la clé supprimée. Ceci peut être utile dans certains contextes.

Question 14

A quoi sert le mot-clé `with` utilisé quand on lit ou écrit dans un fichier ?

- ☐ Il permet de capturer toutes les exceptions qui pourraient se produire.
- ✓ ☒ Il s'assure que le fichier est fermé, même s'il se produit des erreurs dans le bloc.
- ☐ Il permet de stocker des objets dans des fichiers.

Le mot-clé `with` ne capture aucune exception. Cependant, il s'assure que le fichier est fermé, même si des erreurs se produisent. En somme, vous pouvez comparer ces deux blocs comme étant identiques :

Avec `with` :

```
with open(fichier, mode) as fichier:  
    # manipulation du fichier
```

Sans `with` :

```
fichier = open(fichier, mode)  
try:  
    # manipulation du fichier  
finally:  
    fichier.close()
```

Question 15

Après le code suivant, combien d'éléments contiendra `liste1` ?

```
liste1 = [1, 2, 3]
liste1.append(8)
liste2 = liste1
liste2.append(19)
```

- ☐ 3
- ☐ 4
- ✓ ☒ 5

On crée une liste avec trois éléments. On ajoute un élément à la fin de la liste. On affecte ensuite la même liste à une seconde variable. Ne vous trompez pas à cette ligne : on ne crée pas une autre liste. Les variables `liste1` et `liste2` possèdent la même référence. Modifier l'un revient à modifier l'autre. Ainsi, la dernière ligne :

```
liste2.append(19)
```

Est strictement équivalente à :

```
liste1.append(19)
```

Au final, `liste1` contient donc 5 éléments au total.

◀ TP : RÉALISEZ UN BON VIEUX PENDU

APPREHENDEZ LES CLASSES ▶

Le professeur

Vincent Le Goff

Découvrez aussi ce cours en...



Livre



PDF

OpenClassrooms

[L'entreprise](#)

[Alternance](#)

[Forum](#)

[Blog](#)

[Nous rejoindre](#)

Entreprises

[Employeurs](#)

En plus


[Devenez mentor](#)

[Aide et FAQ](#)

[Conditions Générales d'Utilisation](#)

[Politique de Protection des Données Personnelles](#)

[Nous contacter](#)

 [Français](#) ▼

