Initiation à Python — Examen

LP ESSIG

2013 – 2014

Note

Certaines questions sont à choix multiple. Dans ce cas, une seule réponse est attendue. Mais la réponse attendue peut aussi être *toutes les propositions* ou bien *aucune des propositions*.

Première partie

Cours

1. Quelle est l'effet du mot-clé `pass` ?

2. Qu'affiche le code suivant :

```
squared = [x**2 for x in range(5)]
if len(squared) > 3:
  print(squared[3])
else:
  print("Input error")
```

   A – Input error B – 3 C – 16 D – 4

3. Comment définit-on une *docstring* ? Quel est sont rôle ? Pour quelles constructions du langage peut-on l'utiliser ?

4. Par quelle instruction peut-on remplacer l'extrait de code suivant :

```
r = []
for elem in sac:
  if elem.isBlack():
    result.append(elem)
```

   A – `r = [b for b in sac if b.isBlack()]`
   B – `r = [elem for elem in sac and elem.isBlack()]`
   C – `r = [elem.isBlack() for elem in sac]`

5. Quel est le problème avec l'extrait de code suivant :

```
tel = {
  "code": 3420,
  "numero": 1234567890,
```

```
  "marque": "samsung",
}
print(tel["numéro"])
```

6. Quel est l'instruction à privilégier pour répéter une action jusqu'à ce qu'une condition arrête cette répétition ?

7. Comment définit-on une fonction qui prend trois entiers en paramètre ?
   A – `def f(3,int):` B – `def f(a,b,c):` C – `def f(int,int,int):`

8. Quel est l'équivalent de l'expression suivante :

   ```
   bool(len(x) != 0 and y == 0)
   ```

   A – `bool(not (not x or y))`
   B – `bool(x and not y)`
   C – `bool(y and False or x)`

9. Quelles sont les lignes de code à utiliser pour importer le module `time` et utiliser la fonction `localtime` qu'il contient (elle ne prend pas de paramètres).

10. Quelles sont les deux façons de définir un attribut dans une classe ?

11. Comment faire pour qu'une fonction retourne plusieurs valeurs à la fois ?
    A – Mettre plusieurs `return` à la suite, un pour chaque valeur.
    B – Retourner un tuple de ces valeurs.
    C – Utiliser le mot-clef `yield`.

12. Expliquez le rôle d'une exception. Comment est-elle définie ? Comment est-elle déclenchée ?

13. La ligne de code suivante est-elle correcte ? Pourquoi ? Si elle ne l'est pas, proposez une alternative.

    ```
    a, b = 1, 10, 42
    ```

Deuxième partie

Exercices

1. Écrire une fonction `sommeCube`. Cette fonction prend une liste en paramètre. La fonction renvoie la somme du cube (puissance 3) de chaque élément de la liste. Toute valeur non numérique est ignorée. Si la liste ne contient aucune valeur numérique, la fonction renvoie 0.
   Écrire ensuite quelques lignes de code pour tester cette fonction avec différentes listes correspondant aux différents cas de figure (que des nombres, mélange de nombres et d'autres choses, aucun nombre).

2. Écrire une classe `Domino`. Cette classe définit un domino qui contient deux valeurs, une pour chacun de ses côtés. Ces valeurs doivent être initialisées à la création des objets issus de cette classe.

   Cette classe comporte une méthode `affiche` qui dessine le domino en question à l'écran. La méthode `__repr__` est surchargée pour retourner les valeurs contenu dans le domino.

   Ajoutez les méthode `__eq__` et `__gt__` pour comparez deux dominos entre eux (ils sont égaux si les deux valeurs sont identiques et l'un est supérieur à l'autre s'il a au moins une de ses valeurs supérieure aux deux valeurs de l'autre).

   Ajoutez une méthode `placerACote` qui prend un autre domino en paramètre et qui retourne si, oui ou non, les deux dominos peuvent être placés l'un à côté de l'autre.

   Un exemple d'utilisation de la classe pourrait être le suivant :

```
>>> d = Domino(4,6)
>>> d.affiche()
+---+---+
| 4 | 6 |
+---+---+
>>> print(d)
Domino(4, 6)
>>> d == Domino(5,5)
False
>>> d > Domino(5,5)
True
>>> d.placerACote(Domino(4,1))
True
```

3. Soit la classe :

```
class SecurityError(Exception):
    pass

class CompteEnBanque:
    def __init__(self, nom, argent):
        self.proprietaire = nom
        self.balance = argent

    def depot(self, argent):
        if argent < 0:
            raise SecurityError("Dépot négatif non autorisé")
        self.balance += argent

    def retrait(self, argent):
        if argent < 0:
            raise SecurityError("Retrait négatif non autorisé")
        self.balance -= argent

    def decouvert(self):
        return self.balance < 0
```

   Utiliser l'héritage pour étendre les fonctionnalités de la classe précédente en une classe

`CompteSecurise`. Ce compte sécurisé possède, en plus de la quantité d'argent disponible, un code de sécurité sous forme de chaîne de caractères. Cette chaîne est initialement `"abcd"` pour tous les comptes créés.

Modifier la méthode `retrait` pour demander à l'utilisateur (et vérifier) le code avant tout retrait d'argent. Ajouter une méthode `nouveauCode` qui prend en paramètre le nouveau code de sécurité pour le compte qui fait appel à cette méthode.

Écrire ensuite quelques lignes de code pour tester cette classe. Ce test devra créer un compte sécurisé pour `toto` possédant 50 unités d'argent, changer le code de sécurité de ce compte pour une valeur que vous choisirez et tenter de retirer 15 unités d'argent sur ce compte.