Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

B216A Développement informatique

Examen Juin 2017

Solutionnaire

Les réponses aux exercices dont la réponse est un code à fournir sont données à titre d'exemple, plusieurs solutions pouvant en effet répondre aux problèmes posés.

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

1 Vrai ou faux (3 points)

(Connaissances générales)

Pour chacune des affirmations suivantes, indiquez si elle est vraie ou fausse. *Si vous avez répondu correctement à toutes les affirmations d'un bloc, vous obtenez un point, si vous avez fait une faute vous obtenez un demi-point et sinon vous avez zéro (une abstention compte comme une faute).*

1.1 Gestion de projet et réseau

Affirmation	**Vrai**	**Faux**
A/ Lorsque l'on utilise Git, il faut absolument avoir une connexion à internet pour pouvoir faire des commits sur sa machine.	☐	☑
B/ Un test fonctionnel a pour but de vérifier que le code d'une fonction respecte bien ses spécifications.	☑	☐
C/ Un service comme BitTorrent se base sur une architecture peer-to-peer (P2P).	☑	☐
D/ Deux applications connectées qui communiquent sur un réseau avec le protocole TCP ou UDP sont chacune identifiée par une adresse IP et un numéro de port.	☑	☐
E/ Un système de gestion de versions moderne comme Git permet de travailler facilement, en même temps, sur plusieurs versions d'un même programme.	☑	☐

1.2 Algorithmique

Affirmation	**Vrai**	**Faux**
A/ Un programme Python donné peut correspondre à un ou plusieurs algorithmes différents.	☐	☑
B/ Une fonction récursive sans cas de base finit toujours par s'arrêter sans erreur.	☐	☑
C/ Le nombre de nœuds d'un arbre binaire avec n *« niveaux »* vaut $2^0 + 2^1 + 2^2 + ... + 2^{n-1}$.	☐	☑
D/ La postcondition d'une fonction représente les conditions qui seront toujours satisfaites après exécution de la fonction pour autant que la précondition était satisfaite.	☑	☐
E/ Une file est une pile.	☐	☑

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

1.3 Programmation concurrente et fonctionnelle

Affirmation	**Vrai**	**Faux**
A/ Concurrence et parallélisme sont deux synonymes.	☐	☑

B/ L'exécution du programme suivant affiche False.
`x = lambda y, x: x > y`
`y = x(2, 4)`
`print(y)`
	☐	☑

C/ La fonction prédéfinie sum peut être utilisée pour faire la somme d'une liste de nombres complexes.	☑	☐
D/ Il est possible d'exécuter des programmes de manière concurrente sur une machine qui ne peut exécuter physiquement qu'une seule instruction à la fois.	☑	☐
E/ On peut combiner l'utilisation d'un thread et de la fonction sleep pour exécuter une tâche de fond à intervalles réguliers.	☑	☐

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

2 Théorie (4 points)

(Compréhension d'un concept informatique)

Définissez complètement la notion de **fonction** Python. Illustrez votre définition avec un simple exemple commenté d'une fonction fun qui renvoie une fonction et d'un code qui appelle cette fonction fun.

Une fonction permet de rassembler une séquence d'instructions en une entité qu'il est possible d'appeler, ce qui revient à exécuter les instructions de la fonction. Elle est caractérisée par trois éléments :

- *un nom ;*
- *une liste de paramètres (avec leurs types) ;*
- *une valeur de retour (avec son type).*

De plus, une fonction est un objet de type function et on peut les stocker dans des variables, les recevoir en paramètres, renvoyer une fonction comme valeur de retour, etc.
Pour ce faire, on peut, par exemple, définir une fonction imbriquée (c'est-à-dire définie dans une autre et donc invisible de l'extérieur) et la renvoyer :

```
def fun():
    # Définition d'une fonction imbriquée f
    # qui fait la somme de deux nombres entiers a et b
    def f(a, b):
        return a + b
    # Renvoi de la fonction f
    return f

# Appel de la fonction fun, stockage de la valeur renvoyée dans v
v = fun()
# Appel de la fonction contenue dans v, pour faire la somme de 12 et -2
result = v(12, -2)
print(result)
```

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

3 Expression régulière (3 points)

(Niveau élémentaire)

Définissez une expression régulière permettant de vérifier les motifs suivants :

1. Une chaine de caractères donnée doit être une note sur 100 valide pour un examen, sachant qu'on admet au maximum deux chiffres après la virgule, *par exemple* 64*,* 27.5*,* 0 *et* 74.05.
2. Une chaine de caractères donnée doit être une plaque de voiture valide, c'est-à-dire trois lettres romaines suivies de trois chiffres arabes non nuls ou l'inverse, *par exemple* *KDJ283 ou 381DBS*.
3. Une chaine de caractères donnée doit représenter le nom d'une gate d'un aéroport régional, c'est-à-dire la lettre A ou B suivie d'un ou deux chiffres non nuls, *par exemple* *A9 ou B12*.

Expliquez brièvement les différentes parties de vos expressions régulières.

1. Pour faciliter les choses, on va séparer le cas du 100 *car il est clair qu'on n'aura jamais de chiffres après la virgules, excepté* *0 ou 00*, *dans ce cas. Il s'agit de la première expression régulière. Remarquez l'utilisation d'un groupe non-capturant, déclaré avec le préfixe* *?:*, *afin de rendre la partie décimale optionnelle. La seconde expression régulière représente les autres cas, à savoir un seul chiffre avant la virgule (pouvant être nul) ou deux chiffres dont celui à gauche doit être non nul et une partie décimale optionnelle comme pour le premier cas.*

```
100(?:\.[0]{1,2})?|[1-9]?[0-9](?:\.[0-9]{1,2})?
```

2. On a un choix entre deux expressions régulières que l'on va donc déclarer et séparer avec *|*. *Dans le premier cas, les trois premiers caractères sont des lettres romaines majuscules de* *A à Z et les trois derniers des chiffres arabes non nuls de* *1 à 9*. *Dans le second cas, c'est juste l'inverse.*

```
[A-Z]{3}[1-9]{3}|[1-9]{3}[A-Z]{3}
```

3. Le premier caractère est un *A ou un B que l'on fait suivre par un ou deux chiffres non nuls.*

```
[AB][1-9]{1,2}
```

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

4 Trouver l'erreur (5 points)

(Niveau moyen)

(A) L'exécution du programme suivant, qui comporte une fonction récursive permettant de calculer la factorielle d'un nombre naturel, se déroule sans erreur, mais n'affiche pas les réponses attendues. Identifiez **le problème**, expliquez-le et corrigez-le. Justifiez votre réponse.

```
1  def fact(n):
2      result = 1
3      if n >= 0:
4          prev = fact(n - 1)
5          result = n * prev
6      return result
7
8  print(fact(2))                          # Réponse attendue 2
9  print(fact(4))                          # Réponse attendue 24
10 print(fact(5))                          # Réponse attendue 120
```

La fonction telle que définie renvoie toujours la valeur 0. *En effet, si on examine attentivement la définition de la fonction, on voit que le cas récursif correspond à* $n \geq 0$. *Le cas de base est donc* $n < 0$, *en l'occurrence il se produira lorsque* n *vaut* -1 *puisque* n *est décrémenté de* 1 *à chaque appel récursif. Dans ce cas, la fonction renvoie donc* 1, *ce qui ne pose a priori pas de soucis.*

Le problème se situe dans le dernier appel récursif, c'est-à-dire lorsque n *vaut* 0. *On va en effet multiplier le résultat de l'appel récursif stocké dans* prev *par* 0, *ce qui fera toujours* 0.

Pour résoudre le problème, il suffit de corriger le cas de base qui est lorsque n *vaut* 0 *:*

```
if n > 0:
```

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

(B) L'exécution du programme suivant, qui comporte une fonction permettant d'extraire des informations d'une chaine de caractères, se déroule sans erreur, mais n'affiche pas la réponse attendue. Identifiez **les problèmes**, expliquez-les et corrigez-les. Justifiez votre réponse.

```
1  import re
2
3  def price(items):
4      result = []
5      pattern = re.compile(r'.*P=(.*);.*\.', re.MULTILINE)
6      for m in pattern.finditer(items):
7          result.append(m.group(0))
8      return result
9
10 print(price('''C=182738;P=12;Q=2;S=29.
11 C=273616;P=24;Q=7;S=11.'''))                    # Réponse attendue [12, 24]
```

Hint : La fonction doit extraire le nombre naturel se trouvant juste après « P= ».

Il y a deux problèmes dans ce code. Le premier concerne la méthode group *qui récupère la valeur capturée par un groupe. Avec « 0 », on récupère la chaine originale, avec « 1 » le premier groupe, etc. Ici, il faut donc mettre « 1 » puisque l'expression régulière ne contient qu'un groupe qui est celui que l'on veut :*

```
result.append(m.group(1))
```

Ensuite, l'utilisation de « . » dans le groupe capturé est trop large car il se peut du coup que le groupe capturé contienne des « ; » ce qu'on ne veut pas puisqu'il doit justement s'arrêter au premier « ; ». Plusieurs solutions peuvent être mises en œuvre. On peut par exemple remplacer le « . » par des chiffres :*

```
pattern = re.compile(r'.*P=([0-9]+);.*.', re.MULTILINE)
```

Ce qu'on peut aussi faire, c'est de rendre le « . » fainéant à l'aide d'un point d'interrogation, l'arrêtant ainsi au premier « ; » trouvé :*

```
pattern = re.compile(r'.*P=(.*?);.*.', re.MULTILINE)
```

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

5 Compléter le code (5 points) (Niveau avancé)

(A) On souhaite développer un décorateur log qui permet d'affiche à l'écran la valeur des paramètres et la valeur renvoyée par une fonction lorsqu'on l'appelle. Le programme suivant, que vous devez compléter, devrait afficher ce qui suit lors de son exécution :

```
Args   : (2, 9)
Result: 11
11
```

```
1  def log(f):
2      def wrapper(*args):
3          # À compléter...
4      return wrapper
5
6  @log
7  def add(a, b):
8      return a + b
9
10 print(add(2, 9))
```

Hint : La décoration log *ne peut évidemment être utilisée que sur des fonctions renvoyant une valeur.*

On commence par afficher les paramètres reçus qui se trouvent dans args *qui est un tuple. On appelle ensuite la fonction décorée* f *en lui donnant les paramètres avec l'opérateur de déballage et on stocke le résultat dans une variable* result*. On peut ainsi afficher la valeur renvoyée puis la renvoyer :*

```
def log(f):
    # Fonction définissant le comportement après décoration
    def wrapper(*args):
        print('Args : '.format(args))
        # Appel de la fonction qui est décorée
        result = f(*args)
        print('Result: '.format(result))
        return result
    return wrapper
```

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

(B) On souhaite développer une fonction warn d'avertissement qui permet d'afficher une phrase à l'écran après un certain temps, peu importe ce que le programme est en train de faire. Le programme suivant, que vous devez compléter, pourrait afficher ce qui suit lors de son exécution :

```
0
1
COUCOU !
2
3
4
```

```
1 def warn(t, msg):
2     # À compléter
3
4 warn(2, 'COUCOU !')
5 for i in range(5):
6     print(i)
7     time.sleep(1)
```

Hint : Un thread est créé et démarré avec « threading.Thread(target=fct).start() ».

On veut déclencher une action, à savoir afficher un message, après qu'un certain temps se soit écoulé. Pour cela, on va définir une fonction imbriquée qui commence par attendre grâce à time.sleep puis affiche le message. Cette fonction sera lancée dans un thread, de sorte que le reste du programme continue de s'exécuter en parallèle, en attendant que le timer arrive à échéance :

```
def warn(t, msg):
    # Fonction d'affichage d'un message après un certain délai
    def doaction():
        time.sleep(t)
        print(msg)
    # Exécution de la fonction 'doaction' dans un thread parallèle
    th.Thread(target=doaction).start()
```