Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

B216A Développement informatique

Examen Juin 2017

Consignes générales

— Pas de calculatrice.
— Feuilles de brouillon incluses, à rendre avec sa copie.
— Réponses au bic ou stylo, noir ou bleu.

Explications de vos réponses

— Vous pouvez ajouter des commentaires pour expliquer des parties non triviales de vos réponses.
— N'hésitez pas à mentionner toute hypothèse non mentionnée dans l'énoncé que vous auriez prise.
— Parfois un petit schéma vaut mieux qu'un long discours.
— Vous ne devez pas vous soucier des `import` à faire.

Réservé aux correcteurs (questionnaire A)

Q1						/3
Q2						/4
Q3						/3
Q4						/5
Q5						/5
Total						**/20**

Classement : N°1 — **Auditoire** : (siège)

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

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`.

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

(suite de la réponse)

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 arabes 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.

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

(suite de la réponse)

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
```

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= ».

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
```

```python
def log(f):
    def wrapper(*args):
        # À compléter...
    return wrapper

@log
def add(a, b):
    return a + b

print(add(2, 9))
```

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

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
```

```python
def warn(t, msg):
    # À compléter

warn(2, 'COUCOU !')
for i in range(5):
    print(i)
    time.sleep(1)
```

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

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

Brouillon

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

(brouillon suite)

Nom : 10/06/2017 10:30

Matricule : (120 minutes)

B216A — Développement informatique

Commandes Bash de base

Général

`clear`	efface le contenu de la fenêtre
`history`	affiche l'historique des dernières commandes
`exit`	quitte le terminal

Information système

`whoami`	affiche l'identité de l'utilisateur courant
`uptime`	information sur la durée de fonctionnement
`date`	affiche la date courante
`cal`	affiche le calendrier du mois courant
`man <cmd>`	affiche l'aide de la commande `cmd` (*quitter l'aide avec la touche* `q`)
`uname -a`	information sur le système d'exploitation
`w`	liste les utilisateurs connectés
`df`	affiche l'utilisation des disques
`| df -h`	tailles lisibles par un humain

Navigation

`pwd`	affiche le dossier courant
`ls`	liste le contenu du dossier courant
`| ls <path>`	liste le contenu du dossier `path`
`| ls -a`	affiche tous les fichiers et dossiers
`| ls -lh`	liste détaillée et tailles lisibles par un humain
`cd <path>`	change le dossier courant en `path`
`| cd ..`	remonte dans le dossier parent

Manipulation de fichiers

`cat <path>`	affiche le contenu du fichier `path`
`head <path>`	affiche les 10 premières lignes du fichier `path`
`| head -n <nb> <path>`	affiche les `nb` premières lignes du fichier `path`
`tail <path>`	affiche les 10 dernières lignes du fichier `path`
`| tail -n <nb> <path>`	affiche les `nb` dernières lignes du fichier `path`
`mkdir <path>`	crée un nouveau dossier `path`
`touch <path>`	crée un nouveau fichier vide `path`
`cp <path1> <path2>`	copie le fichier `path1` vers le fichier `path2`
`| cp <path1> <path2>/`	copie le fichier `path1` dans le dossier `path2`
`| cp -R <path1> <path2>`	copie le dossier `path1` vers le dossier `path2`
`mv <path1> <path2>/`	déplace le fichier `path1` dans le dossier `path2`
`| mv <path1> <path2>`	déplace le dossier `path1` vers `path2`
`rm <path>`	supprime le fichier `path`
`| rm -i <path>`	demande confirmation avant suppression
`| rm -r <path>`	supprime le dossier `path`
`find <path> -name <name>`	cherche le fichier `name` dans le dossier `path`

Gestion des processus

`ps`	liste statique des processus en cours d'exécution de l'utilisateur courant
`| ps -a`	liste les processus de tous les utilisateurs
`| ps -e`	liste tous les processus
`| ps -f`	affiche toutes les informations des processus
`top`	liste dynamique des processus en cours d'exécution (*quitter la liste avec la touche* `q`)
`kill <pid>`	tue le processus numéro `pid`
`| kill -9 <pid>`	tue violemment le processus

Divers

`diff <path1> <path2>`	affiche la différence entre les deux fichiers
`wc <path>`	affiche le nombre de lignes, de mots et de caractères du fichier `path`
`lpr <path>`	imprime le fichier `path`
`lpq`	affiche les files d'impression
`wget <url>`	télécharge le fichier `url`
`ping <url>`	envoie une requête ping au serveur `url`
`ifconfig`	montre la configuration des interfaces réseau

Cheat Sheet

Nom :
Matricule :

B216A — Développement informatique

10/06/2017 10:30
(120 minutes)

Classement : N°1 — Auditoire : (siège)

Cheat Sheet

Syntaxe des expressions régulières en Python

Il y a 14 *méta-caractères* à précéder d'un backslash pour les capturer :
`.` `^` `$` `*` `+` `?` `{` `}` `[` `]` `\` `|` `(` `)`

Classe de caractères

Représente un ensemble de caractères :

`[abc]`	`a`, `b` et `c`
`[^abc]`	tout sauf `a`, `b` et `c`
`[0-9]`	tous les caractères compris entre `0` et `9`
`[a-zA-Z]`	toutes les lettres de l'alphabet romain

Il existe des classes prédéfinies :

`.`	n'importe quel caractère (sauf retour à la ligne)
`\d`	un chiffre décimal (`[0-9]`)
`\s`	un caractère blanc (`[ \t\n\r\f\v]`)
`\w`	un caractère alphanumérique (`[a-zA-Z0-9_]`)
`\D`, `\S` et `\W`	complémentaires des ensembles précédents

Répétition

Permet de matcher plusieurs fois une même expression régulière R :

R`{n}`	exactement n occurrences
R`{m, n}`	au moins m et au plus n occurrences
R`{m,}`	au moins m occurrences
R`{,n}`	au plus n occurrences

Il existe des répétitions prédéfinies :

R`*`	zéro ou plusieurs occurrences (`{0,}`)
R`+`	une ou plusieurs occurrences (`{1,}`)
R`?`	zéro ou une occurrence (`{0,1}`)

Par défaut, la chaine analysée sera consommée au maximum possible avec les répétitions. Pour ne pas être glouton et consommer le moins possible, on ajoute un `?` en suffixe :
`{m,n}?` `*?` `+?` `??`

Frontière de recherche

`^`	début de la ligne
`$`	fin de la ligne
`\A`	début de la chaine de caractères
`\Z`	fin de la chaine de caractères
`\b`	limite d'un mot (début ou fin)
`\B`	pas la limite d'un mot

Les frontières ne consomment aucun caractères dans la chaine, elles représentent simplement une assertion.

Alternative

R_1`|`R_2	choix entre les deux expressions régulières R_1 et R_2

Groupe

Permet de délimiter une sous-expression régulière R, pour par exemple la répéter ou pour former une alternative :

`(`R`)`	groupe capturant
`(?:`R`)`	groupe non-capturant
`(?P`*name*R`)`	groupe capturant nommé *name*

Il est possible de faire référence à la valeur capturée par un groupe précédent :

`\`i	valeur capturée par le groupe numéro i
`(?P=`*name*`)`	valeur capturée par le groupe nommé *name*

Lookahead

Permet de tester une assertion R, sans avancer dans la chaine :

`(?=`R`)`	succès si R matche à la position courante
`(?!`R`)`	succès si R ne matche pas à la position courante