Classement : N° — **Auditoire** : (siège) **0**

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

PI2T Développement informatique

Examen Juin 2016

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.

Réservé aux correcteurs (questionnaire A)

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

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — 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/ Avec Git, plusieurs copies intégrales d'un projet informatique peuvent exister en même temps sur plusieurs machines.	☐	☐
B/ Un test fonctionnel a pour but de vérifier que l'utilisateur d'un programme parvienne à retrouver facilement et rapidement les différentes fonctions du programme.	☐	☐
C/ Un service comme BitTorrent se base sur une architecture client-serveur.	☐	☐
D/ Deux applications connectées qui communiquent sur un réseau sont chacune identifiée par une adresse IP et un numéro de port.	☐	☐
E/ Lorsqu'on utilise un système de gestion de versions, on est limité à un seul commit par jour.	☐	☐

1.2 Algorithmique

Affirmation	**Vrai**	**Faux**
A/ Pour un algorithme donné, il n'existe qu'un et un seul programme Python correspondant.	☐	☐
B/ Toute fonction récursive finit toujours par s'arrêter.	☐	☐
C/ Tous les nœuds d'un arbre possèdent un parent.	☐	☐
D/ La postcondition d'une fonction représente les conditions qui seront toujours satisfaites après exécution de la fonction.	☐	☐
E/ Une pile est une file.	☐	☐

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — 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 4.
`x = lambda x: x + 2`
`x = x(2)`
`print(x)`
	☐	☐

C/ La fonction prédéfinie sum permet de concaténer tous les éléments d'une liste de chaines de caractères.	☐	☐
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/ Une fonction Python est un objet que l'on peut stocker dans une variable.	☐	☐

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

2 Théorie (4 points)

(Compréhension d'un concept informatique)

Expliquez le concept de **système de gestion de versions** et son intérêt dans le cadre du développement d'un programme informatique, seul et à plusieurs.

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

(suite de la réponse)

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — 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 un code postal constitué de quatre ou cinq chiffres arabes, et tel que le premier (celui tout à gauche) est non nul.
2. Recherche de mots répétés à l'intérieur d'une chaine de caractères donnée (deux mots exactement les mêmes qui se suivent).
3. Une chaine de caractères donnée doit représenter un prix, c'est-à-dire un nombre naturel, suivi éventuellement d'un point suivi d'une séquence de chiffres (de 0 à 9).

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

4 Trouver l'erreur (5 points) (Niveau moyen)

(A) Lorsqu'on exécute le programme suivant, l'interpréteur Python indique un problème au niveau de l'expression régulière.

```
1 import re
2
3 pattern = '[A-Z]:(?:\\[^<>:"/\|?*]+)*\\?'
4 p = re.compile(pattern)
5
6 print(p.match("C:\\Users\\lur\\Desktop"))  # devrait matcher
7 print(p.match("/home/lur/Desktop"))        # ne devrait pas matcher
```

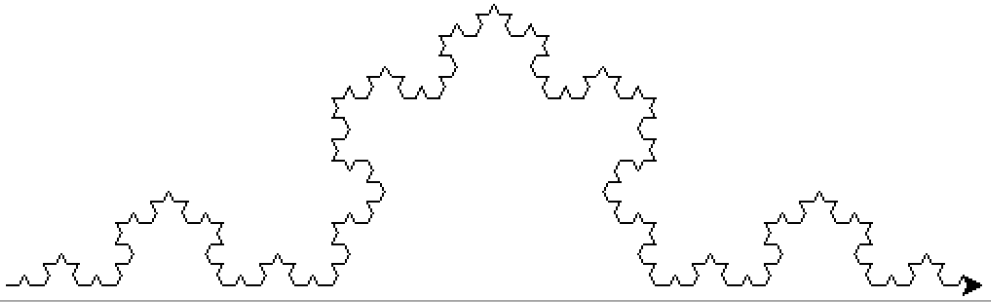
Cette expression régulière doit matcher les *« chemins Windows »*. Sachant que les noms de fichiers Windows ne peuvent pas contenir les caractères `<>:"/\|?*`, identifiez le problème et corrigez-le. Justifiez votre réponse.

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

(B) La fonction récursive suivante devrait dessiner la fractale représentée ci-dessous, mais l'interpréteur Python indique un problème de récursion infinie. Expliquez à quoi est due cette erreur et comment on pourrait la corriger.



```
1  from turtle import *
2
3  def vonkoch(dist):
4      if dist == 5:
5          forward(dist)
6      else:
7          vonkoch(dist/3)
8          left(60)
9          vonkoch(dist/3)
10         right(120)
11         vonkoch(dist/3)
12         left(60)
13         vonkoch(dist/3)
14
15 vonkoch(200)
16 done()
```

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

5 Multiplier des matrices (5 points) (Niveau avancé)

Le programme suivant sert a multiplier des matrices. Il le fait en utilisant la fonction map qui, comme vous le savez, prend une fonction en paramètre. Dans ce programme, c'est la fonction getRowMultiplier qui génère la fonction passée en paramètre à map.

```
1  # In this program, a matrix is a list of rows and a row
2  # is a list of values
3
4  # As a matrix is a list of row
5  # We use the transpose operation to get a list of columns.
6  # That will be useful in the mult function
7  #
8  # - params: a matrix as described before
9  # - return: the transposed matrix (aka. the list of columns)
10 def transpose(matrix):
11     trans = []
12     for j in range(len(matrix[0])):
13         trans.append([])
14         for i in range(len(matrix)):
15             trans[j].append(matrix[i][j])
16     return trans
17
18 # This function return a function that multiply by a single row
19 # It is used with a map function for matrix multiplication
20 #
21 # - params: a row
22 # - return: a function that take a column (list of values) and
23 #           return the result of (row * col)
24 def getRowMultiplier(row):
25     # À compléter...
26
27 # In a matrix multiplication each row of the first matrix
28 # multiply each column of the second one. We can multiply a row
29 # by all the columns at once by using a map function. Each call
30 # to map need a function that multiply by a specific row.
31 # The getRowMultiplier function generate these functions.
32 #
33 # - params: two matrix, the width of the first must match the
34 #           height of the second
35 # - return: a matrix that is the product of the two params
36 # - raise: ArithmeticError if matrix sizes are incompatible
37 def mult(A, B):
38     if len(A[0]) != len(B):
39         raise ArithmeticError("Incompatible Matrix sizes")
40     res = []
41     columnsOfB = transpose(B) # get the list of columns
42     for row in A:
43         res.append(list(map(getRowMultiplier(row), columnsOfB)))
44     return res
45
46
47 matrix = [
48     [1,2],
49     [3,4]
50 ]
51 print(mult(matrix, matrix))
```

Vous devez écrire la fonction getRowMultiplier.

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

```
def getRowMultiplier(row) :
```

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

(suite de la réponse)

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

Brouillon

Nom : 07/06/2016 13:30

Matricule : (120 minutes)

PI2T — Développement informatique

(brouillon suite)

Cheat Sheet

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

Nom :
Matricule :

PI2T — Développement informatique

07/06/2016 13:30
(120 minutes)

Classement : N° **— Auditoire :** (siège) **0**

Nom :
Matricule :

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

PI2T — Développement informatique

07/06/2016 13:30
(120 minutes)

0

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