Traitement de l'information
– Évaluation –

30/11/2018

Documents, téléphones et moyens de communication divers et variés interdits.
Toutes les informations nécessaires sont rappelées dans le sujet.
Les exercices sont indépendants les uns des autres.

Nom : _____

Exercices

1 Questions théoriques

1.1 Quelles sont les différences entre scrapping et utilisation d'une API ?

1.2 Quel est l'indicateur le plus stable : *médiane* ou *moyenne* ? Pourquoi ?

1.3 Dessinez une boîte à moustaches et expliquez comment elle s'interprête.

1.4 J'ai $R^2 = 0,9$ pour une régression linéaire entre les variables A et B. Puis-je en déduire que A cause B ? Pourquoi ?

2 Manipulation d'un fichier JSON

Fichier `équipe.json` :

```json
{
  "nomEquipe": "Super hero squad",
  "ville": "Metro City",
  "formation": 2016,
  "baseSecrete": "Super tower",
  "active": true,
  "membres": [
    {
      "nom": "Molecule Man",
      "age": 29,
      "nomCivil": "Dan Jukes",
      "pouvoirs": [
        "Résistance aux radiations",
        "Changement de taille",
        "Rafale de radiations"
      ]
    },
    {
      "nom": "Madame Uppercut",
      "age": 39,
      "nomCivil": "Jane Wilson",
      "pouvoirs": [
        "Coup de poing spécial",
        "Résistance aux dégats",
        "Réflexes surhumains"
      ]
    },
    {
      "nom": "Eternal Flame",
      "age": 1000000,
      "nomCivil": "Inconnu",
      "pouvoirs": [
        "Immortalité",
        "Immunité à la chaleur",
        "Inferno",
        "Téléportation",
        "Voyage interdimensionnel"
      ]
    }
  ]
}
```

```python
import json
json_file = open("équipe.json", "r")
squad = json.load(json_file)
json_file.close()
```

```python
# Nom de la base secrète
print(

)
```

```python
# Âge de Madame Uppercut
print(

)
```

```python
# Premier pouvoir d'Eternal Flame
print(

)
```

```python
# Nombre de membres de l'équipe
print(

)
```

3 Expressions régulières

Cochez les propositions correspondant aux expressions régulières suivantes :

`ab+c`

☐ abc
☐ ac
☐ abbb
☐ bbc

`a.[bc]+`

☐ abc
☐ abbbbbbbb
☐ azc
☐ abcbcbcbc
☐ ac
☐ asccbbbbcbcccc

`[a-z]+[\.\?!]`

☐ battle !
☐ Hot
☐ green
☐ swamping.
☐ jump up.
☐ undulate ?
☐ is. ?

`[a-zA-Z]*[^,]=`

☐ Butt=
☐ BotHEr,=
☐ Ample
☐ FIdDlE7h=
☐ Brittle =
☐ Other.=

`[a-z][\.\?!]\s+[A-Z]`

☐ A. B
☐ c ! d
☐ e f
☐ g. H
☐ i ? J
☐ k L

Rappels

Groupes et intervalles		Quantifieurs	
`.`	N'importe quel caractère à part `\n`	`*`	0 ou plus
`a|b`	a ou b	`*?`	0 ou plus (lazy)
`[abc]`	a, b ou c	`+`	1 ou plus
`[^abc]`	Tout sauf a, b ou c	`+?`	1 ou plus (lazy)
`[a-z]`	Lettre entre a et z	`?`	0 ou 1
`[A-Z]`	Lettre entre A et Z	`{3}`	Exactement 3 fois
`[0-9]`	Chiffre entre 0 et 9	`{3,}`	3 ou plus
`(...)`	Groupe nommé	`{3,5}`	3, 4 ou 5
`(mi|fa)`	« mi » ou « fa »		
Classes de caractères			
`\s`	Caractère espace	`\S`	Tout sauf un caractère espace
`\d`	Chiffre	`\D`	Tout sauf un chiffre
`\w`	Caractère alphanumérique	`\W`	Tout sauf un caractère alphanumérique
Caractères spéciaux		Ancres	
`\n`	Nouvelle ligne	`^`	Début de chaîne
`\r`	Retour chariot	`$`	Fin de chaîne
`\t`	Tabulation	`\b`	Limite de mot
`\`	Caractère d'échappement	`\B`	Tout sauf une limite de mot

4 Compréhension de code Python

Un développeur a écrit un script `todo_viewer.py` afin de l'aider à ne pas oublier les tâches récurrentes de son travail. Écrivez ci-dessous la sortie de ce script si vous l'exécutiez maintenant :

Fichier `todo_viewer.py` :

```
import csv
import datetime


def print_todo(tasks, importance_level):
    for task in tasks:
        if importance_level == 1:
            print("--- {}".format(task.lower()))
        elif importance_level == 2:
            print("*** {}".format(task.title()))
        else:
            print("/!\ {}".format(task.upper()))


current_day = datetime.date.today().strftime("%a")
todo_to_print = {}

todofile = open("tasks.csv", "r")
for row in csv.DictReader(todofile, delimiter=","):
    if row["Day"] == current_day or row["Day"] == "all":
        if row["Level"] in todo_to_print.keys():
            todo_to_print[row["Level"]].append(row["Task"])
        else:
            todo_to_print[row["Level"]] = [row["Task"]]
todofile.close()

for level in sorted(todo_to_print.keys(), reverse=True):
    print_todo(todo_to_print[level], int(level))
```

Fichier `tasks.csv` :

```
Level,Day,Task
2,Tue,Code review with the intern (10am)
1,all,Update Redmine timeline
1,Tue,Project E meeting (8am)
3,Fri,Run non regression tests
3,Mon,Project B meeting (2pm)
3,Mon,Check week-end backup logs
2,all,Check Jenkins build log
1,Fri,Run static analysis tools
2,Thu,Team meeting (9am)
```

Documentation

Fonctions de base

`sorted(`***iterable***`[, `***cmp***`[, `***key***`[, `***reverse***`]]])` Renvoie une nouvelle liste triée à partir des éléments dans *iterable*. Les arguments *cmp*, *key* et *reverse* sont facultatifs.
cmp spécifie une fonction de comparaison personnalisée à deux arguments qui doit renvoyer un nombre négatif, positif ou zéro, selon que le premier argument est considéré comme inférieur, égal ou supérieur au second argument. La valeur par défaut est `None`.
key spécifie une fonction à un argument utilisée pour extraire une clé de comparaison pour chaque élément de la liste. La valeur par défaut est `None` (comparaison directe des éléments).
reverse est une valeur booléenne. Si la valeur est `True`, les éléments de la liste sont triés comme si chaque comparaison était inversée.

Bibliothèque `datetime`

`date.today()` Renvoie la date locale actuelle.

`date.strftime(`***format***`)` Renvoie une chaîne représentant la date, contrôlée par une chaîne de format.

	Sens	Exemple
`%a`	Jour de la semaine abrégé.	Sun, Mon, . . . , Sat
`%A`	Jour de la semaine.	Sunday, Monday, . . . , Saturday
`%w`	Jour de la semaine sous forme d'entier, 0 étant le dimanche et 6 le samedi.	0, 1, . . . , 6
`%d`	Jour du mois sous forme d'entier à 2 positions.	01, 02, . . . , 31
`%b`	Mois abrégé.	Jan, Feb, . . . , Dec
`%B`	Mois.	January, February, . . . , December
`%m`	Mois sous forme d'entier à 2 positions.	01, 02, . . . , 12
`%y`	Année sous forme d'entier à 2 positions (sans siècle).	00, 01, . . . , 99
`%Y`	Année sous forme d'entier (avec siècle)	1970, 1988, 2001, 2013
`%H`	Heure sous forme d'entier à 2 positions (24 heures).	00, 01, . . . , 23
`%I`	Heure sous forme d'entier à 2 positions (12 heures).	01, 02, . . . , 12
`%M`	Minute sous forme d'entier à 2 positions.	00, 01, . . . , 59
`%S`	Second sous forme d'entier à 2 positions.	00, 01, . . . , 59
`%f`	Microseconde sous forme d'entier à 6 positions.	000000, 000001, . . . , 999999
`%j`	Jour de l'année sous forme d'entier à 3 positions.	001, 002, . . . , 366
`%%`	Le caractère '%'.	%