Solutions des exercices du polycopié de Python

Cours dispensé à l'université de Rennes 2

Romain Tavenard

Exercice 2.1

```
t = -5
if t<=0:
    print("l'eau est sous forme solide")
elif t<100:
    print("l'eau est sous forme liquide")
else :
    print("l'eau est sous forme gazeuse")
#[Sortie] l'eau est sous forme solide
```

Exercice 2.2

— Version avec utilisation du modulo

```
n=8
i=0
while(i<n):
    if i%2!=0:
        print(i)
    i+=1

#[Sortie] 1
#[Sortie] 3
#[Sortie] 5
#[Sortie] 7
```

— Version sans utilisation du modulo

```
n=8
i=1
while(i<n):
    print(i)
    i+=2

#[Sortie] 1
```

```
#[Sortie] 3
#[Sortie] 5
#[Sortie] 7
```

Exercice 2.3

```
import math

def aire_equi(l):
    base = l
    hauteur = l * math.sin(math.pi / 3)
    return base * hauteur / 2

print(aire_equi(1.))
# [Sortie] 0.4330127018922193
```

Exercice 2.4

— Version itérative (avec une boucle)

```
def affiche_u_n():
    u = 2
    while u < 1000:
        print(u)
        u = u ** 2

affiche_u_n()
# [Sortie] 2
# [Sortie] 4
# [Sortie] 16
# [Sortie] ...
```

— Version récursive (avec des appels de fonction)

```
def affiche_u_n(u=2):
    if u < 1000:
        print(u)
        affiche_u_n(u ** 2)

affiche_u_n()
# [Sortie] 2
# [Sortie] 4
# [Sortie] 16
# [Sortie] ...
```

Ici, on a fixé une valeur par défaut à l'argument `u` correspondant à l'initialisation de la suite, pour que l'appel initial se fasse comme pour la version itérative de la fonction (`affiche_u_n()`)

Exercice 3.1

```
import datetime

interv1_date1 = datetime.datetime(1920, 1, 2, 7, 32)
interv1_date2 = datetime.datetime(1920, 3, 4, 5, 53)
duree1 = interv1_date2 - interv1_date1

interv2_date1 = datetime.datetime(1999, 12, 30, 17, 12)
interv2_date2 = datetime.datetime(2000, 3, 1, 15, 53)
duree2 = interv2_date2 - interv2_date1

if duree1 > duree2:
    print("Le premier intervalle est le plus grand.")
else:
    print("Le second intervalle est le plus grand.")
```

Exercice 3.2

```
import datetime

duree_annee_normale = 365 * 24 * 60 * 60

for annee in range(2010, 2031):
    date_debut = datetime.datetime(annee, 1, 1)
    date_fin = datetime.datetime(annee + 1, 1, 1)
    duree_anne_courante = date_fin - date_debut
    if duree_anne_courante.total_seconds() > duree_annee_normale:
        print(annee, "est bissextile")
    else:
        print(annee, "n'est pas bissextile")
```

Exercice 4.1

```
def argmax(liste):
    i_max = None
    # On initialise elem_max à une valeur
    # qui n'est clairement pas le max
    if len(liste) > 0:
        elem_max = liste[0] - 1
    for i, elem in enumerate(liste):
        if elem > elem_max:
            i_max = i
            elem_max = elem
    return i_max
```

```
print(argmax([1, 6, 2, 4]))
# [Sortie] 1
```

Exercice 4.2

```
def intersection(l1, l2):
    l_intersection = []
    for elem in l1:
        if elem in l2:
            l_intersection.append(elem)
    return l_intersection

print(intersection([1, 6, 2, 4], [2, 7, 6]))
# [Sortie] [6, 2]
```

Exercice 4.3

```
def union_sans_doublon(l1, l2):
    l_union = []
    for elem in l1 + l2:
        if elem not in l_union:
            l_union.append(elem)
    return l_union

print(union_sans_doublon([1, 6, 2, 4], [2, 7, 6, 2]))
# [Sortie] [1, 6, 2, 4, 7]
```

Exercice 5.1

```
def compte_prefix(s, prefix):
    compteur = 0
    for mot in s.split():
        if mot.startswith(prefix):
            compteur += 1
    return compteur

print(compte_prefix("la vie est belle au bord du lac", "la"))
# [Sortie] 2
```

Exercice 5.2

```
def compte_sans_casse(s, mot_cible):
    compteur = 0
    mot_cible_minuscules = mot_cible.lower()
    for mot in s.split():
        if mot.lower() == mot_cible_minuscules:
```

```
            compteur += 1
    return compteur
```

Exercice 6.1

```
def compte_occurrences(s):
    d = {}
    for mot in s.split():
        d[mot] = d.get(mot, 0) + 1
    return d

print(compte_occurrences("la vie est belle c'est la vie"))
# [Sortie] {"c'est": 1, 'la': 2, 'belle': 1, 'est': 1, 'vie': 2}
```

Exercice 6.2

```
def somme_valeurs(d):
    s = 0
    for v in d.values():
        s += v
    return s

print(somme_valeurs({"a": 12, "zz": 1.5, "AAA": 0}))
# [Sortie] 13.5
```

Exercice 7.1

```
import os

def nb_lignes(nom_fichier):
    n = 0
    fp = open(nom_fichier, "r")
    for ligne in fp.readlines():
        n += 1
    return n

def nb_lignes_repertoire(repertoire):
    for nom_fichier in os.listdir(repertoire):
        if nom_fichier.endswith(".txt"):
            nom_complet_fichier = os.path.join(repertoire, nom_fichier)
            n = nb_lignes(nom_complet_fichier)
            print(nom_complet_fichier, n)

nb_lignes_repertoire(".")
```

Exercice 7.2

```
import os

def compte_fichiers(repertoire):
    compteur = 0
    for f in os.listdir(repertoire):
        if os.path.isfile(os.path.join(repertoire, f)):
            compteur += 1
    return compteur

print(nb_lignes_repertoire("."))
```

Exercice 8.1

```
import urllib.request
import json

def temps_trajet(ville_origine, ville_destination, cle_gmaps_api):
    url = "https://maps.googleapis.com/maps/api/directions/json?"
    url += "origin=" + ville_origine
    url += "&destination=" + ville_destination
    url += "&key=" + cle_gmaps_api
    print(url)
    fp = urllib.request.urlopen(url)
    contenu = fp.read().decode("utf-8")
    d = json.loads(contenu)
    secondes = d["routes"][0]["legs"][0]["duration"]["value"]
    return secondes

# Utilisez votre clé Google Maps API
print(temps_trajet("Rennes", "Saint-Malo", "..."))
```

Exercice 9.1

```
def bissextile(annee):
    if annee % 4 == 0 and annee % 100 != 0:
        return True
    elif annee % 400 == 0:
        return True
    else:
        return False

def nb_jours(mois, annee):
    if mois in [1, 3, 5, 7, 8, 10, 12]:
        return 31
    elif mois in [4, 6, 9, 11]:
```

```
        return 30
    else:  # Mois de février
        if bissextile(annee):
            return 29
        else:
            return 28

def lendemain(jour, mois, annee):
    if jour < nb_jours(mois, annee):
        return jour + 1, mois, annee
    elif mois < 12:  # Dernier jour du mois mais pas de l'année
        return 1, mois + 1, annee
    else:  # Dernier jour de l'année
        return 1, 1, annee + 1

# Tests de la fonction bissextile
print(bissextile(2004))  # True car divisible par 4 et non par 100
print(bissextile(1900))  # False car divisible par 100 et non par 400
print(bissextile(2000))  # True car divisible par 400
print(bissextile(1999))  # False car divisible ni par 4 ni par 100

# Tests de la fonction nb_jours
print(nb_jours(3, 2010))  # Mars : 31
print(nb_jours(4, 2010))  # Avril : 30
print(nb_jours(2, 2010))  # Février d'une année non bissextile : 28
print(nb_jours(2, 2004))  # Février d'une année bissextile : 29

# Tests de la fonction lendemain
print(lendemain(12, 2, 2010))  # 13, 2, 2010
print(lendemain(28, 2, 2010))  #  1, 3, 2010
print(lendemain(31, 12, 2010)) #  1, 1, 2011
```