Exercice 1 — Corrigé

Cet exercice porte sur les structures de données (listes, p-uplets et dictionnaires).

On dispose de la liste `jours` suivante et du dictionnaire `mois` suivant :

```
jours = ["dimanche", "lundi", "mardi", "mercredi", "jeudi",
    "vendredi", "samedi"]
mois = {1 :("janvier",31), 2 :("février",28), 3 :("mars",31),
    4 :("avril",30), 5 :("mai",31), 6 :("juin",30),
    7 :("juillet",31), 8 :("aout",31), 9 :("septembre",30),
    10 :("octobre",31), 11 :("novembre",30), 12 :("décembre",31)}
```

Question 1. a. À partir de la liste `jours`, comment obtenir l'élément `"lundi"` ?

Réponse : `"lundi"` est le deuxième élément de la liste, on utilise donc l'instruction `jours[1]` pour obtenir `"lundi"`.

Question 1. b. On rappelle que l'opérateur % (modulo) renvoie le reste de la division entière (division euclidienne).

Exemple : 7%3 renvoie 1 qui est le reste de la division de 7 par 3.

Que renvoie l'instruction `jours[18%7]` ?

Réponse : on a $18 = 7 \times 2 + 4$, donc le reste de la division euclidienne de 18 par 7 est égal à 4. L'instruction `jours[18%7]` renvoie donc la valeur de `jours[4]`, c'est-à-dire `"jeudi"`.

Question 2. On rappelle que `jours.index(element)` renvoie l'indice de `element` dans la liste `jours`, par exemple `jours.index("mercredi")` renvoie 3. Le nom du jour actuel est stocké dans une variable `j` (par exemple : `j = "mardi"`).

Recopier et compléter l'instruction suivante permettant d'obtenir le numéro du jour de la semaine `n` jours plus tard :

```
numero_jour =(jours.index( ... ) + ... )% ...
```

Réponse

[16]:
```
jours = ["dimanche", "lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"]
mois = {1 :("janvier",31), 2 :("février",28), 3 :("mars",31),
    4 :("avril",30), 5 :("mai",31), 6 :("juin",30),
    7 :("juillet",31), 8 :("aout",31), 9 :("septembre",30),
    10 :("octobre",31), 11 :("novembre",30), 12 :("décembre",31)}
j = "mardi"
n = 15 # par exemple
numero_jour = (jours.index(j) + n)% 7 # Réponse

# vérification
print(numero_jour, jours[numero_jour])
```

```
3 mercredi
```

Question 3. a. À partir du dictionnaire `mois`, comment obtenir le nombre de jours du mois de mars ?

Réponse

[17]:
```
mois[3][1]
```

[17]: 31

Question 3. b. Le numéro du mois actuel est stocké dans une variable `numero_mois`, écrire le code permettant d'obtenir le nom du mois qu'il sera `x` mois plus tard à partir du dictionnaire `mois`.

Par exemple :

- si `numero_mois = 4` et `x = 5`, on doit obtenir `"septembre"`
- si `numero_mois = 10` et `x = 3`, on doit obtenir `"janvier"`

Réponse

[18]:
```
numero_mois, x = 4, 5 # par exemple

mois[(numero_mois + x) % 12][0] # réponse

# Vérification

print(mois[(numero_mois + x) % 12][0])
```

```
septembre
```

Question 4. a. On définit une date comme un tuple : `(nom_jour,numero_jour, numero_mois,annee)`.

Sachant que `date = ("samedi", 21, 10, 1995)`, que renvoie `mois[date[2]][1]` ?

Réponse

`date[2]` renvoie `10`, l'instruction renvoie donc `mois[10][1]`, c'est-à-dire le nombre de jours du mois d'octobre : 31.

[19]:
```
# Vérification
date = ("samedi", 21, 10, 1995)
mois[date[2]][1]
```

[19]: 31

Question 4. b. Écrire une fonction `jour_suivant(date)` qui prend en paramètre une date sous forme de tuple et qui renvoie un tuple désignant la date du lendemain.

Par exemple :

- `jour_suivant( ("samedi", 21, 10, 1995) )` renvoie `("dimanche", 22, 10, 1995)`
- `jour_suivant( ("mardi", 31, 10, 1995) )` renvoie `("mercredi", 1, 11, 1995)`

On ne tient pas compte des années bissextiles et on considère que le mois de février comporte toujours 28 jours.

Réponse

[20]:
```
def jour_suivant(date):
    nom_jour = jours[(jours.index(date[0]) + 1) % 7]
    numero_jour = (date[1] + 1) % mois[date[2]][1]
    if date[1] < mois[date[2]][1]:
        numero_mois = date[2]
    else:
        numero_mois = (date[2] + 1) % 12
    if date[1] == 31 and date[2] == 12:
        annee = date[3] + 1
    else:
        annee = date[3]
    return (nom_jour, numero_jour, numero_mois, annee)

# Vérification

print(jour_suivant( ("samedi", 21, 10, 1995) ))
print(jour_suivant( ("mardi", 31, 10, 1995) ))
print(jour_suivant(("dimanche", 31, 12, 2022)))
```

```
('dimanche', 22, 10, 1995)
('mercredi', 1, 11, 1995)
('lundi', 1, 1, 2023)
```