DS 3

Santé publique France

Analyse des données des prélèvements

1 Introduction

2 Tracer du nombre de prélèvements par jour

Pour traiter les données fournies, il est nécessaire de calculer le nombre de prélèvement réalisé chacun des jours.

Question 1 *Donner l'instruction permettant de savoir combien il existe d'éléments dans la liste des jours* `liste_jours`.

Correction
```
>>> len(jours)
```

Question 2 *Écrire la fonction d'en-tête* `def is_test(jours:list, jour:str) -> bool :` *permettant de savoir si le jour* `jour` *est dans la liste des jours* `jours` *et renvoyant un booléen. Vous utiliserez une boucle* `for`.

Correction
```
def is_test(jours:list,jour:str) -> bool :
    for d in dates :
        if d == jour :
            return True
    return False
```

Question 3 *Écrire la fonction d'en-tête* `def is_test(jours:list, jour:str) -> bool :` *permettant de savoir si le jour* `jour` *est dans la liste des jours* `jours` *et renvoyant un booléen. Vous utiliserez une boucle* `while`.

Correction
```
def is_test_while(jours:list,jour:str) -> bool :
    i = 0
    while i<len(jours):
        if jours[i] == jour :
            return True
        i=i+1
    return False
```

Question 4 *Écrire l'instruction permettant de savoir si des tests ont été faits le 13 mai 2020. Que renvoie cette instruction ?*

Correction
```
>>> is_test(liste_jours,"2020-05-13")
```
Cette instruction renvoie le booléen True.

Question 5 *Écrire la fonction d'en-tête* `def indices_jour(jours:list, jour:str) -> list :` *permettant de renvoyer* `liste_indices = list[int]`, *la liste des indices correspondants au jour* `jour`.

Correction

```
def indices_jour(jours:list, jour:str) -> list :
    liste_indices = []
    for i in range(len(jours)):
        if jours[i] == jour :
            liste_indices.append(i)
    return liste_indices
```

Question 6 *Écrire la fonction d'en-tête* `def compte_test_jour(jours:list, jour:str, tests:list) -> int :` *permettant de renvoyer le nombres de tests faits le jour* `jour`.

Correction

```
def compte_test_jour(jours:list, jour:str, tests:list) -> int :
    liste_indices = indices_jour(jours, jour)
    nb_tests = 0
    for i in liste_indices :
        nb_tests = nb_tests + tests[i]
    return nb_tests
```

Question 7 *Écrire les instructions permettant de déterminer le nombre de tests réalisés sur les femmes le 5 novembre 2020 et le nombre de tests positifs dénombrés le même jour chez les hommes.*

Correction

```
>>> compte_test_jour(liste_dates,'2020-11-05',nb_prelev_f)
>>> compte_test_jour(liste_dates,'2020-11-05',nb_cas_pos_h)
```

Question 8 *Écrire la fonction d'en-tête* `def compte_test_jour(jours:list, jour:str, tests:list) -> int :` *permettant de renvoyer le nombres de tests faits le jour* `jour`.

Correction

```
def compte_test_jour(jours:list, jour:str, tests:list) -> int :
    liste_indices = indices_jour(jours, jour)
    nb_tests = 0
    for i in liste_indices :
        nb_tests = nb_tests + tests[i]
    return nb_tests
```

Question 9 *Écrire la fonction d'en-tête* `def creer_liste_jours(jours:list) -> list :` *permettant de renvoyer la liste des jours où des tests ont été faits (éliminant donc les doublons) parmi la liste des jours* `jours`.

Correction

```
def creer_liste_jours(jours:list) -> list :
    liste_jours = []
    for d in jours :
        if not(d in liste_jours) :
            liste_jours.append(d)
    return liste_jours
```

Question 10 *Écrire la fonction d'en-tête* `def creer_liste_test(dates:list, tests:list) -> list :` *permettant de renvoyer la liste du nombre de tests faits chaque jour.*

Correction

```
def creer_liste_test(dates:list, tests:list) -> list :
    liste_j = creer_liste_jours(dates)
    res = []
    for jour in liste_j :
        res.append(compte_test_jour(dates,jour,tests))
    return res
```

Question 11 *Écrire les instructions permettant d'obtenir la liste du nombre de tests réalisés par les femmes chaque jour ainsi que le nombre de cas positifs chez les femmes chaque jour.*

Correction

```
>>> l_test_f = creer_liste_test(liste_jours,nb_prelev_f)
>>> l_pos_f = creer_liste_test(liste_jours,nb_cas_pos_f)
```

Question 12 *Écrire la fonction d'en-tête* `def nb_tests_jour(dates:list, test1:list, test2:list) -> list :` *permettant de renvoyer la liste du nombre de tests faits chaque jour en sommant les tests faits sur les hommes et sur les femmes.*

Correction

```
def nb_tests_jour(dates:list, tests1:list, tests2:list) -> list :
    nb_tests_h = creer_liste_test(dates,tests1)
    nb_tests_f = creer_liste_test(dates,tests2)

    nb_tests = [nb_tests_h[i][1]+nb_tests_f[i][1] for i in range(len(
        nb_tests_h))]
    return nb_tests
```

Question 13 *Donner les instructions permettant de charger les bibliothèques nécessaire au tracer de graphes.*

Correction

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
```

On dipose de la liste du nombre de tests faits chaque jour (`nb_test`) et du nombre de tests positifs chaque jour (`nb_pos`).

Question 14 *Donner les instructions permettant de tracer les courbes suivantes : nombre de tests en fonction du jour, nombre de cas positifs en fonction du jour.*

Correction

```
plt.plot(nb_test,label="Nombre de tests")
plt.plot(nb_pos,label="Nombre de cas positifs")
plt.xlabel('Jour')
plt.legend()
plt.show()
```

3 Traitement des mesures

Question 15 *Écrire la fonction d'en-tête* `def recherche_max(liste:list) -> int :` *permettant de renvoyer le maximum d'une liste. On pourrait ainsi avoir le maximum de tests réalisés en une journée, ou le nombre maximum de cas.* ***On ne pourra pas utiliser la fonction max.***

Correction

Afin de lisser les courbes réalisées précédemment, on se propose de réaliser un lissage sur n jours. Pour cela on fait la moyenne des cas du jour 0 au jour n, puis du jour 1 au jour $n+1$ *etc.* Cette méthode se nomme moyenne glissante.

Question 16 *Compléter la fonction d'en-tête* `def moyenne_glissante(tests:list, nb:int) -> list :` *permettant de réaliser la moyenne glissante sur n jours.*

Correction

```
def moyenne_glissante(tests,nb):
    res = []
    for i in range(len(tests)-nb):
        s = 0
        for j in range(i,i+nb):
            s = s+tests[j]
        res.append(s/nb)
    return res
```

Question 17 *Quelles conditions doit-il y avoir sur la liste* `tests` *pour pouvoir faire une recherche dichotomique?*

Correction
Il est nécessaire `tests` soit une liste d'entiers triés.

Question 18 *Écrire la fonction d'en-tête* `def recherche_dicho(tests:list, nb:int) -> bool :` *permettant de savoir si un nombre de tests n b a déjà été fait. On utilisera donc une recherche par dichotomie.*

Correction

```
def appartient_dicho(e,t):
    """Renvoie un booléen indiquant si e est dans t
       Préconditions : t est un tableau de nombres trié par ordre croissant
                       e est un nombre"""
    g = 0 # Limite gauche de la tranche où l'on recherche e
    d = len(t)-1 # Limite droite de la tranche où l'on recherche e
    while g <= d: # La tranche où l'on cherche e n'est pas vide
        m = (g+d)//2 # Milieu de la tranche où l'on recherche e
        pivot = t[m]
        if e == pivot: # On a trouvé e
            return True
        elif e < pivot:
            d = m-1 # On recherche e dans la partie gauche de la tranche
        else:
            g = m+1 # On recherche e dans la partie droite de la tranche
    return False
```