TP d'informatique évalué

1 heure

Consignes

- *La clarté du code sera prise en compte dans l'évaluation :*
  - ▷ *choisissez des noms de variable éclairants et concis ;*
  - ▷ *utilisez des fonctions auxiliaires (bien nommées) si nécessaire ;*
  - ▷ *commentez votre code si nécessaire.*
- *La présentation du code sera prise en compte dans l'évaluation :*
  - ▷ *respectez les conventions de style données en TP ;*
  - ▷ *utilisez les balises* `##` *pour séparer les différentes parties de votre code.*
- *Enregistrez votre travail :*
  - ▷ *dans le répertoire* `TP noté 1`*, dans le sous-répertoire* `GROUPE 1` *(ou* `2`*, ou* `3`*) ;*
  - ▷ *sous le nom* `TP_noté_1_NOM_DE_FAMILLE.py`*.*

Remarques

- *On partira toujours du principe que les fonctions à écrire reçoivent des arguments du type attendu.*
- *Quand une fonction prend en argument un objet de type* `list`*, sauf mention du contraire, on attend que cette fonction traite correctement le cas de la liste vide.*

Q1. (a) Écrire une fonction `isPrime(n)` qui renvoie `True` si l'entier $n \in \mathbb{N}^*$ est premier et qui renvoie `False` sinon.

(b) À la suite de votre code, recopiez le code suivant :

```
## Q1.(b) - Test

n = 1
m = 9
p = 17*23
q = 1999

a = isPrime(n)
b = isPrime(m)
c = isPrime(p)
d = isPrime(q)

print("Q1.(b)")
print(a, b, c, d)
```

Q2. Soit $(u_n)_n \in \mathbb{R}^{\mathbb{N}}$ la suite définie par

$$u_0 \in \mathbb{R} \quad \text{et} \quad \forall n \in \mathbb{N},\ u_{n+1} = \ln(u_n)^2 + 3n.$$

(a) Écrire une fonction `valeur_u(n, u0)` qui prend deux paramètres et qui renvoie le terme u_n d'indice n de la suite $(u_n)_n$.

(b) À la suite de votre code, recopiez le code suivant :

```
## Q2.(b) - Test

n = 100
u0 = 3
u = valeur_u(n, u0)

print("Q2.(b)")
print(u)
```

Q3. (a) Codez une fonction `factorielle(n)` qui renvoie la factorielle $n!$ d'un entier $n \in \mathbb{N}$.
On attend si possible une fonction récursive.

(b) À la suite de votre code, recopiez le code suivant :

```
## Q3.(b) - Test

n = 8
x = factorielle(n)

print("Q3.(b)")
print(x)
```

Q4. (a) Écrire une fonction `moyenne(l)` qui prend en argument une liste de nombres et qui renvoie sa moyenne.

(b) À la suite de votre code, recopiez le code suivant :

```
## Q4.(b) - Test

l1 = [2, 0, 2, 0, 2020, 2222]
m1 = moyenne(l1)
l2 = []
m2 = moyenne(l2)
l3 = [2020]
m3 = moyenne(l3)

print("Q4.(b)")
print(m1, m2, m3)
```

Q5. (a) Écrire une fonction `monMin(l)` qui prend en argument une liste de nombres et qui renvoie son plus petit élément.
Si la liste est vide, par convention, on renverra `None`.

(b) À la suite de votre code, recopiez le code suivant :

```
## Q5.(b) - Test

l1 = [2, 0, 2, 0, -1, 2222]
min1 = monMin(l1)
l2 = []
min2 = monMin(l2)

print("Q5.(b)")
print(min1, min2)
```

Q6. (a) Écrire une fonction `distanceMin(l)` qui renvoie un `tuple (i, j)` tel que

- `i` $\neq$ `j` ;
- la valeur $\left|$`l[i]` $-$ `l[j]`$\right|$ est minimale.

Si la liste `l` vérifie `len(l) <= 1`, par convention, on renverra `None`.

(b) À la suite de votre code, recopiez le code suivant :

```
## Q6.(b) - Test

l1 = [2, 5, 8, 0, -1, 2222, 2222.5]
minDist1 = distanceMin(l1)
l2 = [7]
minDist2 = distanceMin(l2)

print("Q6.(b)")
print(minDist1, minDist2)
```

Q7. (a) Écrire une fonction `index_by_dichotomy(l, a)` qui prend en argument une liste `l` de nombres triés et un objet `a` et qui renvoie un indice `i` tel que `l[i] == a`.
Si `a` n'est pas dans la liste, la fonction renvoie `None`.

(b) À la suite de votre code, recopiez le code suivant :

```
## Q7.(b) - Test

l = [2, 5, 8, 10, 15, 2222]
a1 = 8
a2 = 11
i1 = index_by_dichotomy(l, a1)
i2 = index_by_dichotomy(l, a2)

print("Q7.(b)")
print(i1, i2)
```

Q8. Les matrices sont implémentées en `Python` sous la forme de listes de listes.

(a) Représenter les matrices

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 8 & 9 \end{pmatrix}$$

en `Python` et affecter ces objets aux variables `M1` et `M2`.

(b) Écrire une fonction `matriceEchelonnee(M)` qui renvoie une matrice échelonnée équivalente par lignes à une matrice `M` donnée.

On s'efforcera d'écrire un code modulaire, c'est-à-dire un code structuré sous la forme de modules : votre programme contiendra en plus de la fonction principale autant de fonctions auxiliaires que nécessaire.

(c) À la suite de votre code, recopiez le code suivant :

```
## Q8.(c) - Test

L1 = [2, 5, 8, 10]
L2 = [3, -1, 2, 4]
L3 = [1, 4, 5, 7]
M = [L1, L2, L3]

M_ech = matriceEchelonnee(M)

print("Q8.(c)")
print(M_ech)
```