TP2 - Structures algorithmiques

1 Alternative

1.1 Lecture

Calculer la valeur de la variable indiquée après l'exécution du code, puis vérifier avec Pyzo.

Question 1. Que vaut x ?

```
x = -2
if not (x>0 and x<5) :
    x = -x
else :
    x = 2*x
```

Question 2. Que vaut z ?

```
x,y = 1,2
if not x>y :
    z = x
else :
    z = y
```

Question 3. Que vaut x ?

```
x = -2
if not (x>0 and x<5) :
    x = -x
if  x>0 and x<5 :
    x = -2*x
```

Question 4. Que vaut y ?

```
x = -3
if  x<-4 :
    y = 0
elif  x<-3 :
    y = 4-x
elif x<-1 :
    y = x*x+6*x+8
elif x<3 :
    y = 2-x
else :
    y = -2
```

Question 5. Que vaut k ?

```
x,y = 2,3
d,c = 5,4
if x>0 and x<d :
    if y>0 and y<c :
        k = y
    else :
        k = x
else :
    k = 1
```

Question 6. Que vaut y ?

```
x,y = 3,-2
if x<y :
    y = y-x
elif x==y :
    y = 0
else :
    y = x-y
```

1.2 Programmes utilisant des alternatives

Répondre aux questions dans l'éditeur de code, penser à exécuter le fichier en appuyant sur **F5** et à tester vos programmes.

Question 7. Ecrire le code de la fonction `racine` qui prend en paramètre un nombre `x` et qui retourne la racine si celui-ci est positif ou le booléen `False` dans le cas contraire.

Question 8. Ecrire le code de la fonction `bissextile` prenant en paramètre une année (donc un entier) et retournant un booléen indiquant si celle-ci est bissextile ou non.

On rappelle que depuis octobre 1582, une année n est bissextile si et seulement si n est divisible par 4, sauf si n est divisible par 100 mais pas par 400. On rappelle également qu'avant 1582, les années bissextiles étaient exactement les années multiples de 4.

Question 9. Ecrire le code de la fonction `sol2ndDegre` prenant en paramètres les coefficients d'une équation du second degré. La fonction retournera soit un couple de solutions, soit une seule solution, soit le booléen `False` s'il n'y a pas de solutions.

Exercice 1 (Jeu de devinette). On souhaite mettre au point un petit jeu dont le principe est le suivant : l'utilisateur doit penser à un animal parmi les cinq suivants hirondelle, dauphin, chat, diplodocus et tyrannosaure, et l'ordinateur doit deviner duquel il s'agit en utilisant uniquement les questions ci-dessous :

- Est-ce que l'animal existe encore aujourd'hui ?
- Est-ce que l'animal vole ?
- Est-ce que l'animal nage ?
- Est-il carnivore ?

1. Ecrire une fonction `jeuDevinette` prenant en entrées quatre booléens `rep1`, `rep2`, `rep3` et `rep4` correspondants aux réponses données à ces quatre questions et renvoyant l'animal recherché sous forme de chaîne de caractères.
2. Ecrire une fonction `jouerJeu` ne prenant aucun paramètre d'entrée, qui pose les questions au joueur et affiche la réponse de l'ordinateur.

   On donne ci-dessous le squelette de cette fonction à compléter. La fonction `input` permet de poser une question et la fonction `int` transforme la réponse de l'utilisateur en entier.

```
def jouerJeu () :
    rep1 = int(input("Est-ce que l'animal existe encore aujourd'hui ?"))==1
    #....
    reponse = jeuDevinette(...)
    print(reponse)
```

Pour jouer avec l'ordinateur, il suffit d'appeler la fonction `jouerJeu ()` dans la console.

2 Boucles

2.1 Lecture

Pour chaque morceau de code ci-dessous, anticiper le résultat avant de vérifier avec Pyzo.

On rappelle que la fonction `print` permet d'afficher un message à l'écran comme le contenu d'une variable.

Question 10.

```
i = 5
while i>=0 :
    i = i-1
    print(i)
```

Question 11.

```
i = 2
while i<6 :
    i = i+1
    if i%3 == 0 :
        print("x")
    print(i)
```

Question 12.

```
i = 5
while i>=0 :
    i = i-1
print(i)
```

Question 13.

```
i = 0
while i<=7 :
    i = i+1
    if i%3 == 0 :
        i = i+1
    else :
        print(i)
```

Question 14.

```
a = 1
b = 1
for i in range (1,11) :
    print(a)
    c = a+b
    a = b
    b = c
```

Question 15.

```
x = 11
for n in range (0,10) :
    print(n,x)
    if x%2 == 0 :
        x = x//2
    else :
        x = 3*x+1
```

2.2 Programmes utilisant des boucles

Répondre aux questions en utilisant une boucle `for` lorsque c'est possible et en utilisant une boucle `while` sinon.

Question 16. Ecrire le code de la fonction `sommeN` qui prend en paramètre un entier `n` et qui renvoie la somme des entiers de 1 à n. Pour $n = 1\,000$, vous devez obtenir 500 500.

Il est interdit d'utiliser la formule de Gauss.

Question 17. Ecrire le code de la fonction `sommeNbImpairs` qui prend en paramètres les entiers $n1$ et $n2$ tels que $n1 \leq n2$ et qui retourne la somme des entiers impairs compris entre $n1$ et $n2$ inclus.

Question 18. Ecrire le code de la fonction `sommeK` prenant en paramètres deux entiers n et k et renvoyant la somme des puissances $k^{ièmes}$ des entiers de 1 à n.

Question 19. Ecrire le code de la fonction `sommeMultiples35` prenant en paramètre un entier naturel n et renvoyant la somme de tous les multiples de 3 ou de 5 strictement inférieurs à n. Par exemple, pour $n = 1\,435$, la fonction renvoie 480 248.

Question 20. Ecrire le code de la fonction `bricole` qui prend en paramètre un entier naturel n à 6 chiffres (pas besoin de le vérifier) et qui renvoie le résultat issu des manipulations suivantes :

1. on coupe le nombre en deux parties : les 3 premiers chiffres forment le nombre `deb` et les 3 derniers le nombre `fin` ;
2. il faut répéter `fin` fois les deux opérations suivantes :
   (a) multiplier `deb` par 13
   (b) et conserver les 3 derniers chiffres comme nouvelle valeur de `deb`.
3. on renvoi la valeur alors contenue dans la variable `deb`.

Par exemple, avec $n = 317\,010$, la fonction renvoie 133.

Exercice 2 (Palindromes). Un nombre palindrome est un nombre qui se lit dans les deux sens, par exemple 1221. Le nombre palindrome le plus grand issu du produit de deux entiers à deux chiffres est $9009 = 91 \times 99$. On cherche le nombre palindrome le plus grand issu du produit de deux entiers à trois chiffres.

Noter que comme on ne représente pas les zéros devant le premier chiffre de gauche d'un nombre, les nombres ayant un ou des zéros à droite ne seront pas des palindromes.

1. Ecrire le code de la fonction `retourner` prenant en paramètre un entier naturel n et renvoyant un autre entier dont l'écriture est l'inverse de n. Par exemple, pour $n = 123$, la fonction retourne 321.
   Il est interdit d'utiliser des fonctions de conversions vers un autre type. Il faut utiliser avantageusement les opérateurs `%` et `//`.

2. En déduire une fonction `estPalindrome` prenant en paramètre un entier naturel n et retournant un booléen indiquant si le nombre passé en paramètre est un nombre palindrome.
3. Ecrire une fonction `plusHautPalindrome` ne prenant pas de paramètres et retournant le nombre palindrome le plus grand issu du produit de deux entiers à trois chiffres.
   Le résultat est 906 609.

Exercice 3 (Suite de Syracuse). La suite de Syracuse est une suite d'entiers naturels définie de façon récurrente de la façon suivante : soit $N \in \mathbb{N}$, on pose $u_0 = N$ et pour tout $n \geq 0$:

$$\begin{cases} u_{n+1} = \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ u_{n+1} = 3 * u_n + 1 & \text{si } u_n \text{ est impair.} \end{cases}$$

La conjecture affirme que pour tout N, il existe un indice n tel que $u_n = 1$. On cherche la valeur de N comprise entre 1 et 1 million donnant la plus longue suite de valeurs avant d'atteindre 1. Par exemple, avec $N = 30$, la suite donne les valeurs suivantes : 15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2 et 1. Le premier 1 est atteint en 19 étapes puis le cycle 4, 2 et 1 se répète ensuite indéfiniment.

1. Ecrire le code de la fonction `decompteSyracuse` prenant en paramètre un entier N et renvoyant le nombre de termes de la suite de Syracuse permettant d'atteindre 1 en partant de N.
2. Ecrire la fonction `plusLongueSyracuse` ne prenant aucun paramètre et retournant le couple constitué de la valeur de N comprise entre 1 et 1 000 000 ainsi que le nombre d'étapes pour laquelle la suite est la plus longue pour $u_0 = N$.
   La fonction doit renvoyer $(837\,799, 525)$.