TP 1 : Quelques bases en Python

I Opérations élémentaires

Exercice 1 :

1. Taper dans la console les lignes de commandes suivantes :

```
6/2   6//2   7/2   7//2   2**3   3**2   16%3   17%3
```

2. Expliquer par une courte phrase ce que fait chacune des opérations / , // , ** et % .

> / correspond à la division, le résultat est du type **float** (c'est un réel).
>
> // correspond à la division entière, c'est le quotient dans la division euclidienne. C'est du type **int** (entier).
>
> ** correspond à la puissance.
>
> % correspond au reste de la division euclidienne.

II La boucle for

Exercice 2 :
Que font les six instructions Python suivantes :

1.
```
for i in range(1, 12, 1):
    print(i)
```

2.
```
for i in range(1, 12):
    print(i)
```

3.
```
for i in range(0, 13, 3):
    print(i)
```

4.
```
for i in range(6):
    print(i)
```

5.
```
for i in range(16,6,-1):
    print(i)
```

6.
```
for i in range(19,-7,-4):
    print(i)
```

> Il faut bien comprendre ici les rôles de debut,fin et pas dans la commande `range(debut,fin,pas)` . Si on ne mentionne pas le pas, il prend la valeur 1 par défaut. Si on ne mentionne pas debut il prend la valeur 0 par défaut.

Exercice 3 :

Quel programme écrire pour obtenir l'affichage suivant :

```
3
7
11
15
19
```

> Il faut faire attention dans une boucle for la valeur de "fin" n'est jamais atteinte.
>
> ```
> for i in range(3,20,4):
>     print(i)
> ```

III La boucle while

Exercice 4 :

Expliquer en une phrase que fait ce programme :

```
u=1
while u<500:
   print(u)
   u=u*2
```

Ce programme affiche les puissance de 2 plus petites que 500.

Exercice 5 :

Écrire un programme similaire pour obtenir la liste des puissances de 3 inférieures ou égales à 2000.

```
u=1
while u<=2000:
   print(u)
   u=u*3
```

IV Fonction en python

Exercice 6 :

Voici le code d'une fonction python :

```
def opp(a):
    return -a
```

où a est une nombre réel, appelé paramètre de la fonction.

1. Que renvoie opp(-3) ?

opp(-3) renvoie 3

2. Que fait la fonction `opp` en général ?

La fonction `opp` renvoie l'opposé du réel a, c'est à dire $-a$.

3. Écrire une fonction `inv` qui renvoie l'inverse d'un réel non nul.

```
def inv(a):
    return 1/a
```

4. Tester la fonction `inv` avec le paramètre 5, puis le paramètre $\frac{1}{3}$. Donner les valeurs obtenues.

```
>> inv(5)
0.2
>> inv(1/3)
3.0
```

Exercice 7 :

Écrire une fonction `perim` qui prend en paramètre la longueur et la largeur d'un rectangle et renvoie son périmètre.

```
def perim(L,l):
    #L représente la longueur
    #l représente la largeur
    return 2*(L+l)
```

Exercice 8 :

1. Soit f la fonction définie sur $\mathbb{R}$ par $f(x) = x^3 - 5x^2 + 3$.
   La fonction python suivante `f` de paramètre `x` permet de renvoyer la valeur de $f(x)$.

```
def f(x):
    return x**3-5*x**2+3
```

   Utiliser cette fonction pour donner $f(-1)$, $f(1)$, $f(1,5)$ et $f\left(\frac{5}{2}\right)$.

```
>> f(-1)
-3
>> f(1)
-1
>> f(1.5)
-4.875
>> f(5/2)
-12.625
```

   On obtient $f(-1) = -3$, $f(1) = -1$, $f(1,5) = -4,875$ et $f\left(\frac{5}{2}\right) = -12,625$

2. Soit g la fonction définie sur $\mathbb{R}$ par $g(x) = 2x^3 + 3x - 5$.
   (a) Écrire une fonction python `g` de paramètre `x` qui renvoie la valeur de $g(x)$.

```
def g(x):
    return 2*x**3+3*x-5
```

   (b) Utiliser cette fonction pour compléter le tableau de valeur suivant :

x	-2	-1	0	1	2	3
$g(x)$	-27	-10	-5	0	17	58

V Structure conditionnelle

Exercice 9 :

Voici le code d'une fonction python :

```
def f(a):
    if a<0:
        return -a
    else:
        return a
```

1. Que renvoie `f(2)` et `f(-3)` ?

   `f(2)` renvoie 2 et `f(-3)` renvoie 3

2. Que fait la fonction `f` en général ?

   La fonction `f` renvoie la valeur absolue de a

VI Pour aller plus loin : Syracuse

Exercice 10 : Cet exercice porte sur la suite de Syracuse à partir d'un entier positif x, on construit à la main une suite (u_n) de la manière suivante :

- $u_0 = x$
- $u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$

1. Prenons $x = 5$, calculer les valeurs de u_1 , u_2 , u_3 , u_4, ..., u_9

$u_1 = 3 \times 5 + 1 = 16$

$u_2 = \dfrac{16}{2} = 8$

$u_3 = \dfrac{8}{2} = 4$

$u_4 = \dfrac{4}{2} = 2$

$u_5 = \dfrac{2}{2} = 1$

$u_6 = 3 \times 1 + 1 = 4$

$u_7 = 2$

$u_8 = 1$

$u_9 = 4$

2. On considère l'algorithme ci-dessous,

```
Algo
Variables : x,n  nombres entiers naturels
Traitement :
                Entrée x
                n prend la valeur 0
                Tant que x différent de 1 faire
                        Si x est un nombre pair alors
                                x prend la valeur x/2
                        Sinon:
                                x prend la valeur 3*x+1
                        n prend la valeur n+1
                Fin Tant que
                Renvoyer n
Fin
```

Exécuter l'algorithme à la main en complétant le tableau suivant pour les valeurs de x suivantes :

Pour $x = 3$:

	x	n	$x \neq 1$
Initialisation	3	0	Vrai
	10	1	Vrai
	5	2	Vrai
	16	3	Vrai
	8	4	Vrai
	4	5	Vrai
	2	6	Vrai
	1	7	Faux

Pour $x = 6$:

	x	n	$x \neq 1$
Initialisation	6	0	Vrai
	3	1	Vrai
	10	2	Vrai
	5	3	Vrai
	16	4	Vrai
	8	5	Vrai
	4	6	Vrai
	2	7	Vrai
	1	8	Faux

Pour $x = 9$:

	x	n	$x \neq 1$
Initialisation	9	0	Vrai
	28	1	Vrai
	14	2	Vrai
	7	3	Vrai
	22	4	Vrai
	11	5	Vrai
	34	6	Vrai
	17	7	Vrai
	52	8	Vrai
	26	9	Vrai
	13	10	Vrai
	40	11	Vrai
	20	12	Vrai
	10	13	Vrai
	5	14	Vrai
	16	15	Vrai
	8	16	Vrai
	4	17	Vrai
	2	18	Vrai
	1	19	Faux

3. Écrire la fonction correspondante en Python de paramètre x et la tester avec les valeurs de x ci-dessus ainsi que pour $x = 127$

```
def syracuse(x):
    n=0
    while x!=1:
        print(x)
        if x%2==0:
            x=x//2
        else:
            x=3*x+1
        n=n+1
    return n
```

syracuse(5) renvoie 5.
syracuse(3) renvoie 7.
syracuse(6) renvoie 8.
syracuse(9) renvoie 9.
syracuse(127) renvoie 46.

Remarque :
La valeur n renvoyée par cet algorithme s'appelle *le temps de vol* de la suite de Syracuse commençant pas $u_0 = x$