

OBJECTIFS : L'objectif de ce TP est de continuer à écrire des programmes, en utilisant à présent davantage les fonctions.

DOCUMENT A RENDRE : Ce travail est évalué. Vous en rédigerez un compte-rendu sous le nom « *tp4_nomfamille.doc* » ou « *tp4_nomfamille.odt* » et vous le transférez en fin d'activité par l'intermédiaire de l'onglet transfert du site https://nsibrantly.fr/transferts_premiere.php en utilisant le code : **tp4** . Ce compte-rendu contiendra :

- les réponses aux différentes questions posées,
- les captures d'écran **des morceaux de codes** écrits et celles **des résultats des exécutions** données dans le shell. Pour faire ces captures, utiliser *l'Outil Capture d'écran* de Windows.

1. Code qui calcule son âge en fonction de son année de naissance :

Le code ci-dessous est incomplet :

```
# Fonctions
def mon_age(annee) :
    if annee > 2020 :
        return "pas encore né"
    else :
        age = 2020 - annee
        return age

# Main
a = mon_age(2004)
b = mon_age(2022)
c = mon_age(2000)
print(a)
print(b)
print(c)
```

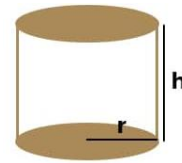
En l'exécutant, cela donne :

```
>>> (executing file "ex_1.py")
16
pas encore né
20
```

⇒ Recopier et compléter ce code pour obtenir le même résultat à l'exécution

2. Code qui calcule le volume d'un cylindre

Le volume d'un cylindre est donné par la relation ci-contre avec $\pi \approx 3.14$.



$$\text{volume} = \pi \times r^2 \times h$$

Le code ci-dessous permet de calculer ce volume en m^3 :

```
# Fonctions
def volume_cylindre(r,h) :
    if r < 0 or h < 0 :
        return "les valeurs doivent être positives"
    else :
        volume = 3.14 * r**2 * h
        message = str(volume) + " m3"
        return message
```

Main

```
message1 = volume_cylindre(-1,2)
message2 = volume_cylindre(1,-2)
message3 = volume_cylindre(1,1)
v = volume_cylindre(1,2)
print(message1)
print(message2)
print(message3)
print(v)
```

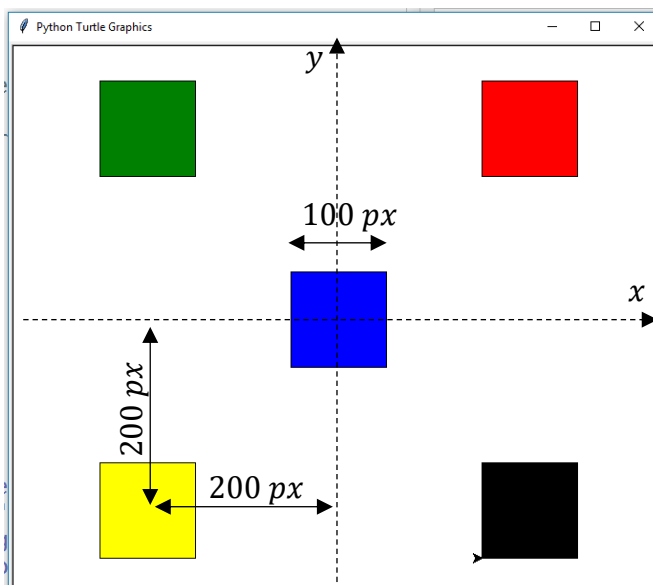
```
>>> (executing file "ex_2.py")
les valeurs doivent être positives
les valeurs doivent être positives
3.14 m3
6.28 m3
```

Son exécution donne :

⇒ Compléter ce code pour obtenir le même résultat.

3. Code qui trace des carrés de couleur :

L'exécution du code ci-contre donne le tracé ci-dessous (le repère a été rajouté, il n'est pas tracé) :



⇒ Compléter ce code afin d'obtenir le même résultat.

```
1 from turtle import *
2
3 # Fonctions
4 def carre(a,x,y,couleur) :
5     up()
6     fillcolor(couleur)
7     x = x - a/2
8     y = y - a/2
9     goto(x,y)
10    down()
11    begin_fill()
12    for i in range(4) :
13        forward(a)
14        left(90)
15    end_fill()
16
17 # Main
18 carre(100,0,0,"blue")
19 carre(100,200,200,"red")
20 carre(100,-200,-200,"yellow")
21 carre(100,-200,200,"green")
22 carre(100,200,-200,"black")
23 exitonclick()
```

4. Code qui compte le nombre de caractères d'un string – Mode Débogage de Pyzo :

⇒ Enregistrer le code ci-contre dans un fichier nommé *ex_4.py* :

⇒ Que retourne la fonction *nb_caractere* après exécution ?

```
exercice_1.py
1 # Fonctions
2 def nb_caracteres(chaine) :
3     nb = 0
4     for l in chaine :
5         nb = nb + 1
6     return nb
7
8 # Main
9 n = nb_caracteres("Vacances")
10 print(n)
```

⇒ Compléter ci-contre le tableau (à reproduire) en donnant les valeurs des variables étape par étape :

<i>nb</i>	<i>chaine</i>	<i>l</i>
0	Vacances	V
1	Vacances	a
2	Vacances	c
3	Vacances	a
4	Vacances	n
5	Vacances	c
6	Vacances	e
7	Vacances	s

5. Code qui inverse l'ordre des lettres dans une chaîne de caractère :

⇒ Enregistrer le code ci-contre dans un fichier nommé *ex_5.py* :

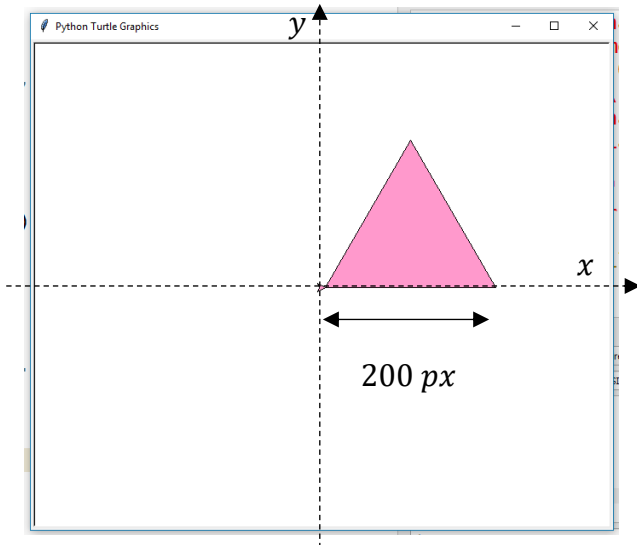
```
1 # Fonctions
2 def nb_caracteres(chaine) :
3     a_l_envers = ""
4     for l in chaine :
5         a_l_envers = l + a_l_envers
6     return a_l_envers
7
8 # Main
9 mot_1 = nb_caracteres("Nadal")
10 mot_2 = nb_caracteres(mot_1)
11 print(mot_1)
12 print(mot_2)
```

⇒ Compléter le tableau ci-dessous, donnant la valeur des variables au cours de l'exécution :

	<i>mot_1</i>	<i>mot_2</i>	<i>chaine</i>	<i>a_l_envers</i>	<i>l</i>
<i>Etape 1</i>			<i>Nadal</i>		<i>N</i>
<i>Etape 2</i>			<i>Nadal</i>	<i>N</i>	<i>a</i>
<i>Etape 3</i>			<i>Nadal</i>	<i>aN</i>	
<i>Etape 4</i>			<i>Nadal</i>		
<i>Etape 5</i>			<i>Nadal</i>		
<i>Etape 6</i>			<i>Nadal</i>		
<i>Etape 7</i>	ladaN		ladaN		l
<i>Etape 8</i>	ladaN		ladaN	l	a
<i>Etape 9</i>	ladaN		ladaN	al	d
<i>Etape 10</i>	ladaN		ladaN	dal	a
<i>Etape 11</i>	ladaN		ladaN	adal	N
<i>Etape 12</i>	ladaN		ladaN	Nadal	N
<i>Etape 13</i>	ladaN	Nadal	ladaN	Nadal	N

6. Code qui trace un triangle équilatéral en couleur :

L'exécution du code ci-contre donne le tracé ci-dessous (le repère a été rajouté, il n'est pas tracé) :



```

1 from turtle import *
2
3 # Fonctions
4 def triangle(a,x,y,r,g,b) :
5     up()
6     goto(x,y)
7     fillcolor(r,g,b)
8     down()
9     begin_fill()
10    for i in range(3) :
11        forward(a)
12        left(120)
13    end_fill()
14
15 # Main
16 triangle(200,0,0,1,0.6,0.8)
17
18 exitonclick()

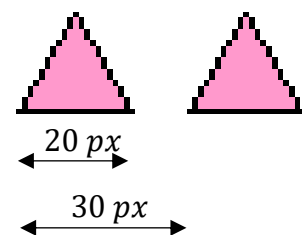
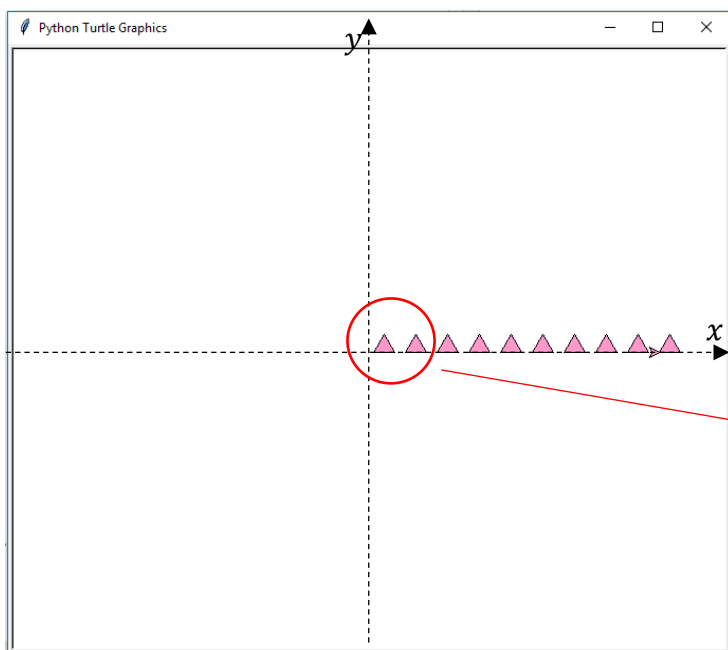
```

⇒ Compléter ce code afin d'obtenir le même résultat.

(**INFO** : si $r = 1$ et $g = b = 0$, on a du rouge
 si $r = b = 0$ et $g = 1$, on a du green
 si $r = g = 0$ et $b = 1$, on a du blue)

7. Code qui trace plusieurs triangles roses

⇒ Faire un copier/coller de la fonction triangle dans un nouveau fichier `ex_7.py`. Compléter alors ce code en utilisant cette fonction, afin d'obtenir le dessin ci-dessous à son exécution :



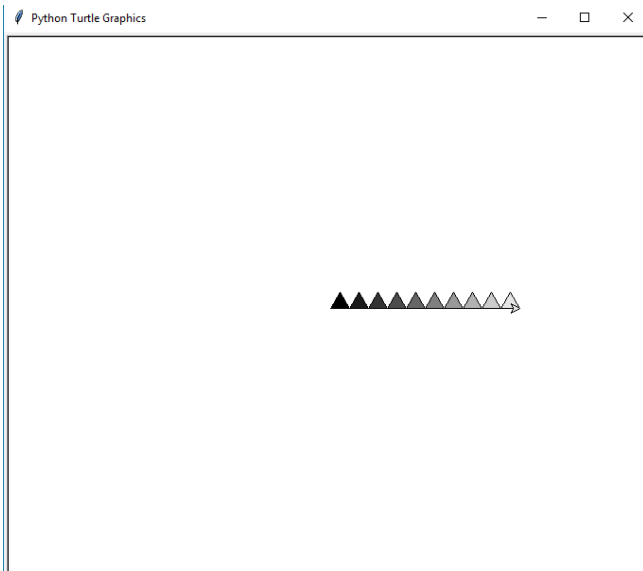
```
1 from turtle import *
2
3 # Fonctions
4 def triangle(a,x,y,r,g,b) :
5     up()
6     goto(x,y)
7     fillcolor(r,g,b)
8     down()
9     begin_fill()
10    for i in range(3) :
11        forward(a)
12        left(120)
13    end_fill()
14
15 # Main
16 for i in range(10) :
17     x = 30*i
18     y = 0
19     triangle(20,x,y,1,0.6,0.8)
20
21 exitonclick()
```

8. Code qui trace encore des triangles, mais autrement :

⇒ On donne une autre version de la fonction *triangle* . Ici, les coordonnées x,y ont été enlevées. Après tracé du triangle, la tortue se déplace sur le sommet de droite.

```

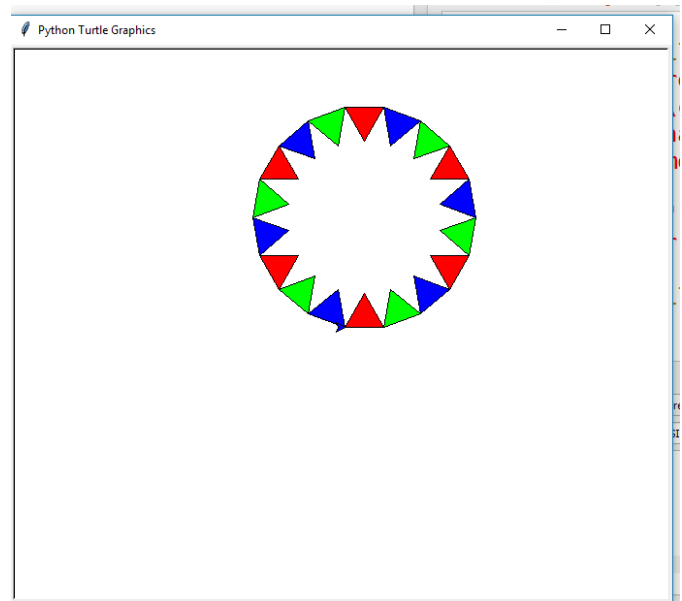
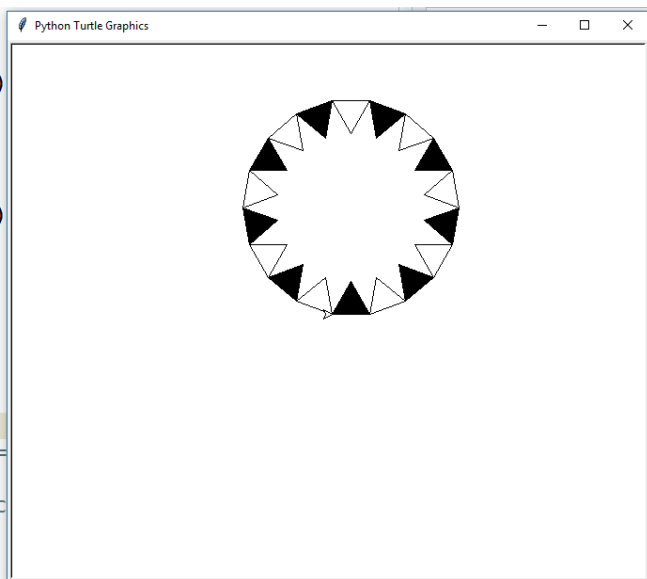
1 from turtle import *
2
3 # Fonctions
4 def triangle(a,r,g,b) :
5     up()
6     fillcolor(r,g,b)
7     down()
8     begin_fill()
9     for i in range(3) :
10         forward(a)
11         left(120)
12     end_fill()
13     forward(a)
14
15 # Main
16 for i in range(10) :
17     c = i / 10
18     triangle(20,c,c,c)
19
20 exitonclick()
    
```



⇒ Modifier ce code afin qu'il puisse tracer le motif donné ci-dessous :

a)

b)



Aide : Pour ces tracés, on pourra utiliser les fonctions :

`left(20)` ou `if i%3 == 0 :` ou

```
1 from turtle import *
2
3 # Fonctions
4 def triangle(a,r,g,b) :
5     up()
6     fillcolor(r,g,b)
7     down()
8     begin_fill()
9     for i in range(3) :
10         forward(a)
11         left(120)
12     end_fill()
13     forward(a)
14
15 # Main
16 speed(10)
17 for i in range(18) :
18     if i%2 == 0 :
19         c = 0
20     else : c = 1
21     triangle(40,c,c,c)
22     left(20)
23
24 exitonclick()
```

```
1 from turtle import *
2
3 # Fonctions
4 def triangle(a,r,g,b) :
5     up()
6     fillcolor(r,g,b)
7     down()
8     begin_fill()
9     for i in range(3) :
10         forward(a)
11         left(120)
12     end_fill()
13     forward(a)
14
15 # Main
16 speed(10)
17 for i in range(18) :
18     if i%3 == 0 :
19         r,g,b = 1,0,0
20     elif i%3 == 1 :
21         r,g,b = 0,1,0
22     else : r,g,b = 0,0,
23     triangle(40,r,g,b)
24     left(20)
25
26 exitonclick()
```