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 numérique en utilisant un logiciel de traitement de texte (*Word ou Libre Office*). Le fichier constitué sera appelé *tp4.doc* ou *tp4.odt* et devra être transféré en fin d'activité **par l'intermédiaire** du site *nsibranly.fr* : se loguer et transférer 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 > 2022 :
        return "pas encore né"
    else :
        age = 2022 - annee
        return age

# Main
a = mon_age(2006)
print(a)
a = mon_age(2025)
print(a)
print(mon_age(2000))
```

En l'exécutant, cela donne :

```
>>> (executing file "ex_1.py")
16
pas encore né
22
```

⇨ 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 $\pi \approx 3.14$.

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
sortie = volume_cylindre(-1,2)
print(sortie)
message =  volume_cylindre(1,-2)
print(message)

v = volume_cylindre(1,2)
print(v)
assert volume_cylindre(1,1) == "3.14 m3"
```

Son exécution donne :

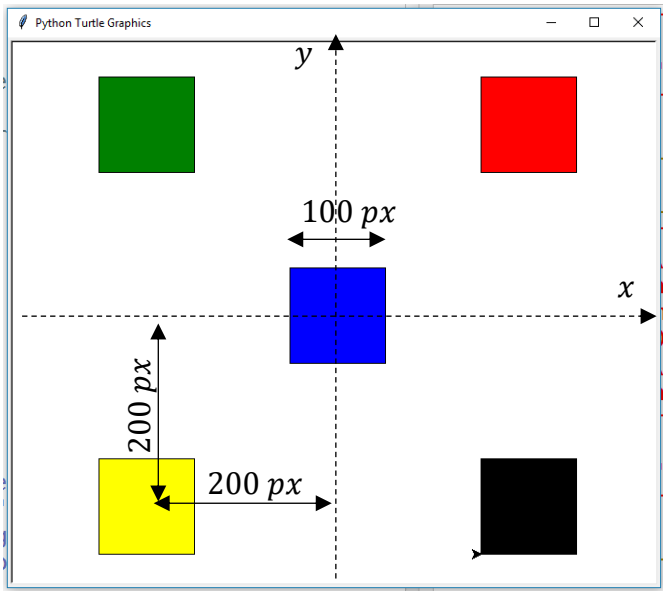
```
>>> (executing file "ex_2.py")
les valeurs doivent être positives
les valeurs doivent être positives
6.28 m3
```

⇨ Compléter ce code pour obtenir le même résultat à l'exécution.

INFO : avec le mot clé **assert**, on crée une assertion. Si la valeur renvoyée par l'exécution volume_cylindre(1,1) n'est pas égale à ''3.14 m3'' , le code s'arrête et indique une erreur.

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)** :




```
from turtle import *

# Fonctions
def carre(a,x,y,couleur) :
    up()
    fillcolor(couleur)
    x = x - a/2
    y = y - a/2
    goto(x,y)
    down()
    begin_fill()
    for i in range(4) :
        forward(a)
        left(90)
    end_fill()

# Main
carre(100,0,0,"blue")
carre(100,200,200,"red")
carre(100,-200,-200,"yellow")
carre(100,-200,200,"green")
carre(100,200,-200,"black")
exitonclick()
```

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

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* :

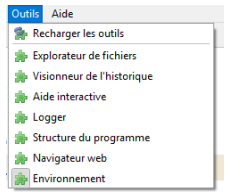
⇨ Que retourne la fonction *nb_caractere* après exécution ?

Cette fonction retourne le nombre de caractère qui composent la chaine de caractère mise en argument.

exercice_1.py

```
# Fonctions
def nb_caracteres(chaine) :
    nb = 0
    for l in chaine :
        nb = nb + 1
    return nb

# Main
n = nb_caracteres("Vacances")
print(n)
```

⇨ Afficher la fenêtre *Environnement* de pyzo :



⇨ Insérer un point d'arrêt ● en cliquant ici .

⇨ Exécuter alors le script (*F5 clavier*) et cliquer sur  pour l'exécuter ligne par ligne et **compléter ci-contre le tableau** (*à reproduire*), **donnant les valeurs des variables** étape par étape :

```
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)
11
12
13
14
15
16
```

```
Python
>>> (executing file "exerci
")

(nb_caracteres)>>>
```

Environnement

Nom	Type	Repr
chaine	str	'Vacances'
n	int	8
nb_caracteres	function	<function nb_caracteres at 0x000001C86

nb	***chaine***	***l***
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 chaine de caractère :

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

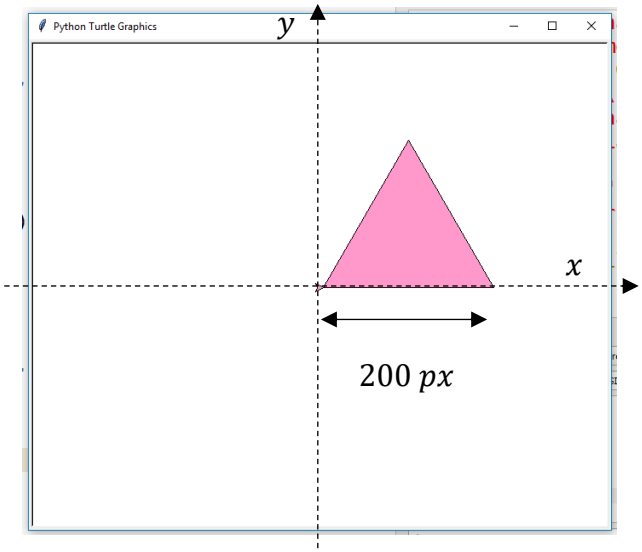
```
1  # Fonctions
2  def nb_caracteres(chaine) :
3      a_l_envers = ""
4      for l in chaine :
5          a_l_envers = l + a_l_envers
6      return a_l_envers
7
8  # Main
9  mot_1 = nb_caracteres("Nadal")
10 mot_2 = nb_caracteres(mot_1)
11 print(mot_1)
12 print(mot_2)
```

⇨ Réaliser une exécution normale (F5) ou étape par étape (avec point d'arrêt et ) et compléter le tableau ci-dessous, donnant la valeur des variables au cours de l'exécution :

	mot_1	***mot_2***	***chaine***	***a_l_envers***	***l***
Etape 1			Nadal		N
Etape 2			Nadal	N	a
Etape 3			Nadal	aN	d
Etape 4			Nadal	daN	a
Etape 5			Nadal	adaN	l
Etape 6			Nadal	ladaN	l
Etape 7	**ladaN**		**ladaN**		**l**
Etape 8	**ladaN**		**ladaN**	**l**	**a**
Etape 9	**ladaN**		**ladaN**	**al**	**d**
Etape 10	**ladaN**		**ladaN**	**dal**	**a**
Etape 11	**ladaN**		**ladaN**	**adal**	**N**
Etape 12	**ladaN**		**ladaN**	**Nadal**	**N**
Etape 13	**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)** :

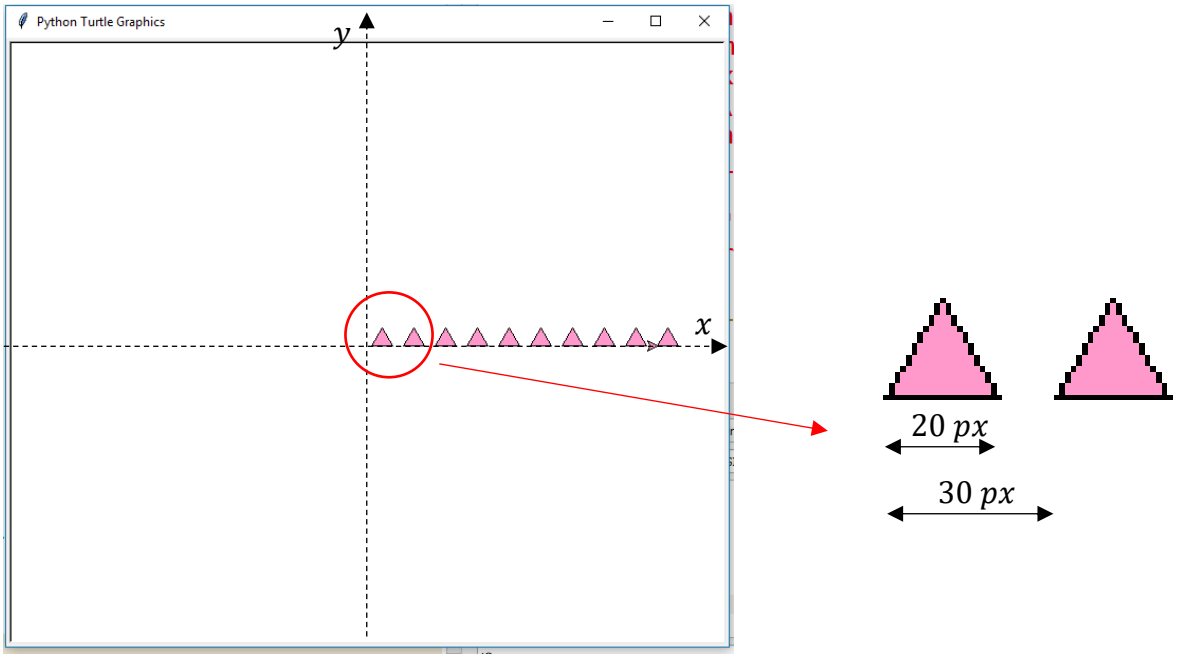


⇨ Compléter ce code afin d'obtenir le même résultat.

```
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()
```

7. Code qui trace plusieurs triangles roses

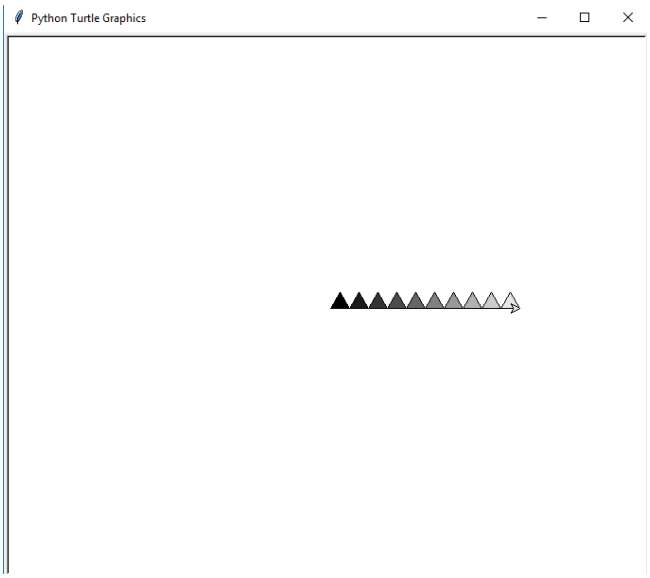
⇨ Faire un copier/coller de la fonction triangle dans un nouveau fichier *ex_7.py* . Compléter alors ce code en utilisant cette fonction, afin d'obtenir le dessin ci-dessous à son exécution :



```
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 :

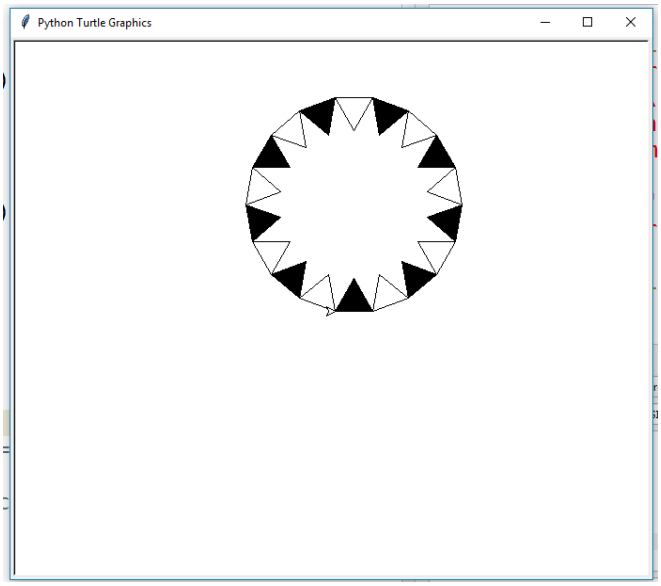
⇨ On donne une autre version de la fonction *triangle* . Ici, les coordonnées x,y ont été enlevées. Après tracé du triangle, la tortue se déplace sur le sommet de droite.



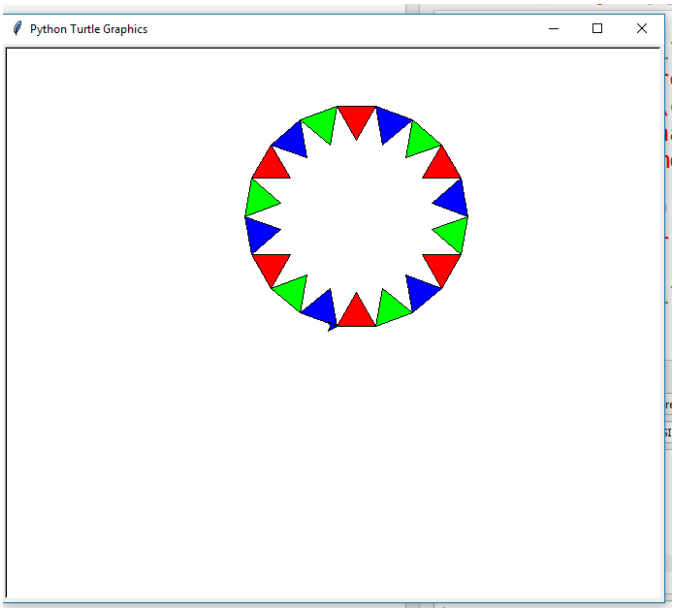

```
from turtle import *

# Fonctions
def triangle(a,r,g,b) :
    up()
    fillcolor(r,g,b)
    down()
    begin_fill()
    for i in range(3) :
        forward(a)
        left(120)
    end_fill()
    forward(a)

# Main
for i in range(10) :
    c = i / 10
    triangle(20,c,c,c)

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 ……

```python
from turtle import *

# Fonctions
def triangle(a,r,g,b) :
    up()
    fillcolor(r,g,b)
    down()
    begin_fill()
    for i in range(3) :
        forward(a)
        left(120)
    end_fill()
    forward(a)

# Main
speed(10)
for i in range(18) :
    if i%2 == 0 :
        c = 0
    else : c = 1
    triangle(40,c,c,c)
    left(20)

exitonclick()
```

```python
from turtle import *

# Fonctions
def triangle(a,r,g,b) :
    up()
    fillcolor(r,g,b)
    down()
    begin_fill()
    for i in range(3) :
        forward(a)
        left(120)
    end_fill()
    forward(a)

# Main
speed(10)
for i in range(18) :
    if i%3 == 0 :
        r,g,b = 1,0,0
    elif i%3 == 1 :
        r,g,b = 0,1,0
    else : r,g,b = 0,0,
    triangle(40,r,g,b)
    left(20)

exitonclick()
```

9. Code qui renvoie le nombre de voyelles d'un mot :

⇨ Ecrire le script d'une fonction *voyelle(mot)* qui renvoie le nombre de voyelles de la chaine de caractère mise en argument.

```python
def voyelle(mot) :
    nb = 0
    for l in mot :
        if l in "aeiouy" :
            nb = nb + 1
    return nb

# Main
n = voyelle("salut les amis")
print(n)
```

10. Code qui calcule une somme :

⇨ Ecrire le script d'une fonction *somme(n)* qui renvoie la somme suivante :

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \cdots \ldots . + \frac{1}{n}$$

⇨ Exécuter somme*(500000)*

```
def somme(n) :
    s = 0
    for i in range(1,n+1) :
        s = s + 1/i
    return s

# Main
s = somme(500000)
print(s)
```

11. Code qui inverse l'ordre des lettres :

⇨ Ecrire le script d'une fonction *verlan(mot)* qui renvoie la chaine de caractère mise en argument avec les lettres dans un ordre inversé.

```
def verlan(mot) :
    new = ""
    for l in mot :
        new = l + new
    return new

# Main
m = verlan("bonjour les amis")
print(m)
```

12. Code qui renvoie une chaine de caractère de 6 lettres qui sont définies aléatoirement :

⇨ Ecrire le script d'une fonction *mystere()* qui renvoie une chaine de caractère de 6 lettres définies aléatoirement.

```
from random import randint

def mystere() :
    mot = ""
    for i in range(6) :
        num = randint(1,26)
        n = 0
        for l in "abcdefghijklmnopqrstuvwxyz" :
            n = n + 1
            if n == num :
                mot = mot + l
    return mot

# Main
m = mystere()
print(m)
```