NSi	Les modules en Python	TP3

1. Module math :

1.1. pi :

Dans l'interpréteur Python, taper **pi,** que constatez-vous ?

Toujours dans l'interpréteur, taper import math puis math.pi, que constatez-vous ?

- Calculer le périmètre d'un cercle de rayon = 2cm.
  Recopier la syntaxe :

- Calculer la surface de ce cercle. Nous écrirons X puissance 2 de la manière suivante : X**2
  Recopier la syntaxe :

1.2. ceil :

A l'aide de la fonction **ceil**, compléter le tableau suivant :

x	-1,5	-2	1,5	2
math.ceil(x)				

1.3. floor :

A l'aide de la fonction **floor**, compléter le tableau suivant :

x	-1.5	-2	1.5	2
math.floor(x)				

1.4. sqrt:

A l'aide de la fonction **sqrt**, compléter le tableau suivant :

x	4	9	16	25
math.sqrt(x)				

1.5. pow:

Taper: math.pow(2,3) => math.pow(3,2) =>

Conclusion:

Proposer une nouvelle syntaxe afin de calculer la surface du cercle en **1.1.**

1.6. hypot:

Déterminer la valeur de l'hypoténuse d'un triangle rectangle ayant pour dimensions 2 et 1 cm :

Taper ensuite dans l'interpréteur : math.hypot(2, 1) :

Que constatez-vous ?

1.7. gcd:

Quel est le plus grand diviseur commun entre 10 et 6 :

Taper ensuite dans l'interpréteur : math.gcd(10,6):

Que constatez-vous ?

1.8. Création d'un module :

On désire créer un module permettant de connaître les différentes caractéristiques d'une sphère en fonction de la valeur de son rayon (diamètre, surface de la section, volume).

Créer un programme qui sera en fait un module: **caracteristique_sphere.py** et saisissez le code suivant :

```
def perimetre(r) :
    resultat = 2*pi*r
    return resultat

def surface_section(r) :
    resultat = pi*r**2
    return resultat
```

Créer un nouveau programme qui appelera une partie du module caracteristique_sphere :

```
from caracteristique_sphere import surface_section
r=2
s=surface_section(r)
print("la surface de la section de la sphère vaut", s)
```

Que doit-on rajouter au programme du module pour que cela fonctionne ?

Rajouter 2 fonctions supplémentaires au programme du module **caracteristique_sphere** afin qu'il puisse déterminer, le volume ainsi que la masse.

Concernant le programme **masse**, il faudra tenir prendre en compte une seconde variable, la ***masse volumique*** (mv).

1.9. Création de fonction :

Ecrire une fonction produisant le même résultat que la fonction ceil (naturellement sans utiliser cette dernière.

Idem avec floor

2. Module random:

Ce module permet la génération de nombres aléatoires.

- ✓ Saisir import random dans l'interpréteur.

2.1. Completez le tableau suivant:

Saisir dans l'interpréteur et exécuter	1er essai	2ième essai	3ième essai
random.random()			
random.randint(0, 3)			
random.sample(['a', 'b', 'c', 'd', 'e'], 2)			
ma_liste=["alpha","bravo","charlie","delta","oscar","papa","romeo","tango","yankee","zulu"]			
random.choice(ma_liste)			

2.2. Définir une fonction jeu permettant d'afficher un nombre entier au hasard entre 0 et un autre nombre saisi par l'utilisateur.

2.3. Afin de rendre ludique ce jeu on va fixer le nombre de tentatives à 5.
Dépassé ce nombre, un message indiquera à l'utilisateur qu'il a dépassé le nombre de tentatives.

2.4. Améliorez votre programme afin que l'utilisateur saisisse lui-même le nombre de tentatives maximal

3. Module Turtle :

Le Module Turtle permet de dessiner en programmant. Découvrons quelques possibilités offertes par ce module.

3.1. Après avoir créé un nouveau fichier, saisir le code suivant :

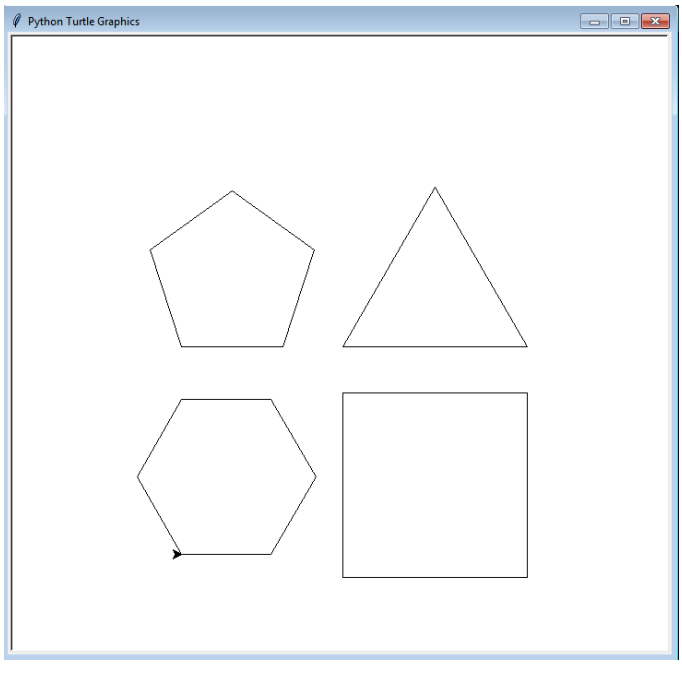
```
from turtle import *
def carré (pixel):
    for i in range(4):
        forward (pixel)
        left(90)
carré(200)
```

Lancer son execution

3.2. Compléter le programme précédent en insérant entre la première et la deuxième ligne, le code suivant :

```
up()
goto(0,-250)
down()
```

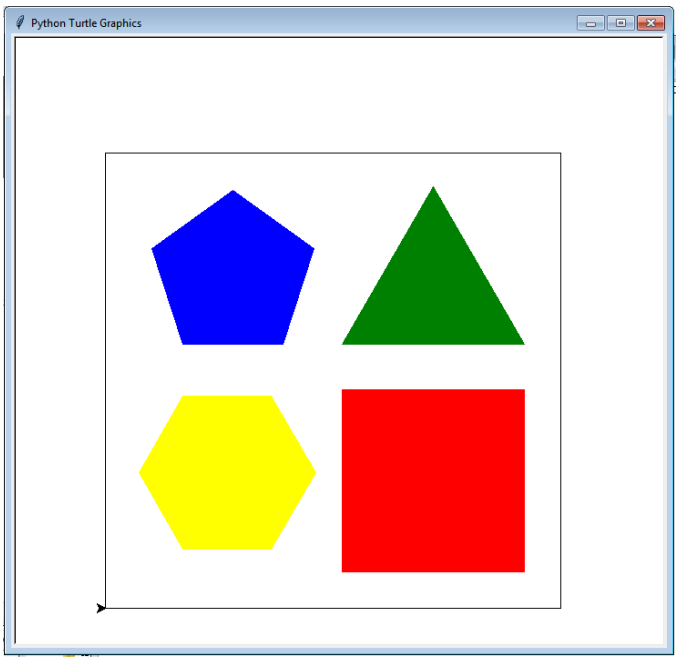
3.3. Compléter encore le code afin d'obtenir le motif suivant :



3.3. Il est possible de colorer les formes obtenues grâce à begin_fill() et end_fill(), par exemple le carré en rouge :

```
def carré (pixel):
    color("red")
    begin_fill()
    for i in range(4):
        forward (pixel)
        left(90)
    end_fill()
carré(200)
```

3.4. Compléter encore le code afin d'obtenir le triangle vert, le pentagone bleu et l'hexagone jaune le tout encadré d'un cadre noir comme sur la copie d'écran ci-dessous :



4. Module Matplotlib:

Ce module est une bibliothèque qui permet de tracer des graphes.

4.1. Après avoir créé un nouveau fichier, saisir le script suivant :

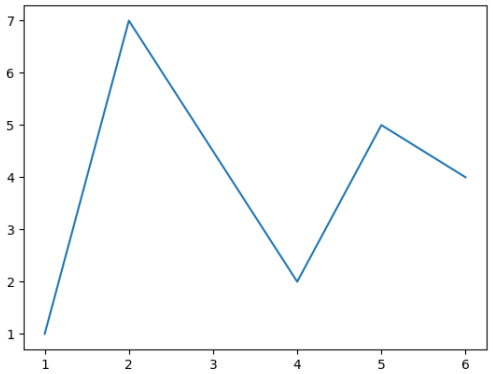
```
import matplotlib.pyplot as plt

X = [1, 3, 4, 6]
Y = [2, 3, 5, 1]

plt.figure() # début de la figure
plt.plot(X, Y) # on trace la figure

plt.show()
```

Lancer son execution

Remarque: on peut modifier la couleur et l'épaisseur grâce à :

plt.plot(X, Y, color="green", linewidth=3.0)

4.2. Modifier votre programme afin d'obtenir la courbe suivante :



4.3. Lors d'un TP de physique on a obtenu le relevé de température suivant.

Temp(°C)	21	25	30	45	55	65	70	80	85
Temps(s)	0	30	60	90	120	150	180	210	240

- Proposer un programme permettant de tracer la courbe caractéristique de la température en fonction du temps.
- Utiliser les instructions ***plt.ylabel("Temp(°C) ")*** et ***plt.xlabel("Temps(s)")*** afin d'ajouter des légendes aux axes.

4.4. Comment tracer la courbe représentative d'une fonction :

Soit la fonction f(x)= x^2- 2x+1, créer un nouveau fichier, saisir le script suivant :

```
import matplotlib.pyplot as plt
def f(x):
   return x**2-2*x+1
x=[i*0.01 for i in range (101)]
y=[f(u) for u in x]
plt.plot(x,y)
plt.grid()
plt.show()
```

- La fonction plot construit la courbe
- La fonction show l'affiche à l'écran
- La fonction grid affiche la grille

4.5. Tracer plusieurs courbes :

Il est également possible de tracer simultanément plusieurs courbes.

- Après avoir créé un nouveau fichier, saisir le script suivant :

```
import matplotlib.pyplot as plt
def f(x):
   return x**2-3*x+2
def g(x):
   return -x**2+x+1

x=[i*0.01 for i in range (101)]
y1=[f(u) for u in x]
y2=[g(u) for u in x]

plt.plot(x, y1, "red"))
plt.plot(x, y2, "green")
plt.grid()
plt.show()
```

- Modifier le script précédent afin de tracer les courbes représentatives des 2 fonctions suivantes
  f1(x)= x+2
  f2(x)= -x+3

4.6. Pour le fun : des courbes en 3 dimensions :

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from math import pi, sin, cos

fig = plt.figure()
ax = fig.gca(projection='3d')
theta = [-4 * pi + i * 8 * pi / 100 for i in range(101)]
z = [-2 + i * 4 / 100 for i in range(101)]
r = [zi ** 2 + 1 for zi in z]
x = [r[i] * sin(theta[i]) for i in range(101)]
y = [r[i] * cos(theta[i]) for i in range(101)]

ax.plot(x, y, z, label='parametric curve')
ax.legend()

plt.show()
```

N'hésitez pas à utiliser