A. Turtle

E.417

`forward(d)`	avance `d` pixels
`backward(d)`	recule `d` pixels
`left(a)`	tourne à gauche de `a`^{o}
`right(a)`	tourne à droite de `a`^{o}
`setposition(x,y)`	se déplace au point `(x,y)`
`home()`	se déplace au point `(0,0)`
`dot(t)`	trace un point de taille `t`
`penup()`	lève le stylo
`pendown()`	baisse le stylo
`pencolor(coul)`	change la couleur du stylo
`pensize(t)`	fixe la taille du stylo à `t`
`clear()`	efface la fenêtre
`speed(n)`	fixe la vitesse à `n` *(de 0 à 10)*
`screensize(w,h)`	fixe les dimensions de la fenêtre
`exitonclick()`	libère la fenêtre

E.414

Saisissez et exécutez le code ci-dessous dans votre IDE/Python :

```
1 from turtle import *
2 forward(120)
3 left(90)
4 color("red")
5 forward(80)
6 done()
```

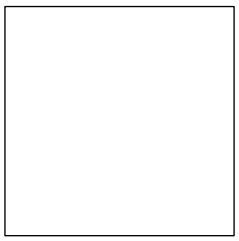
414-code.py

Compléter le code pour obtenir un carré.

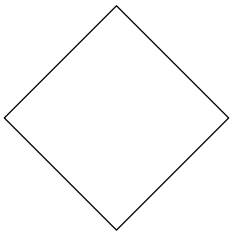
E.415

Réaliser les figures suivantes à l'aide du module "`turtle`" :

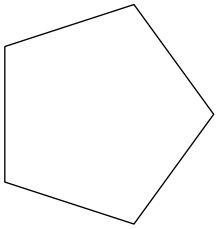
a



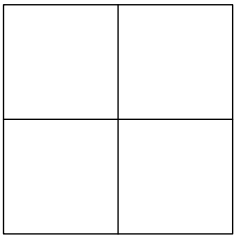
b



c



d



B. Turtle - fonction

E.891

On considère l'algorithme ci-dessous :

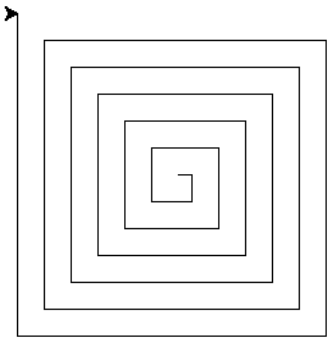
```
from turtle import *
fd(10); rt(90); fd(20); rt(90); fd(30)
rt(90); fd(40); rt(90); fd(50); rt(90)
fd(60); rt(90); fd(70); rt(90); fd(80)
rt(90); fd(90); rt(90); fd(100); rt(90)
fd(110); rt(90); fd(120); rt(90); fd(130)
```

891-code-a.py

1. Télécharger ce code, puis exécutez-le pour observer la figure tracée.
2. Voici un second algorithme :

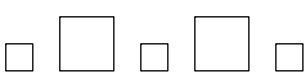
```
from turtle import *
def trace(a):
    fd(a+10); rt(90); fd(a+20); rt(90);
    fd(a+30); rt(90); fd(a+40); rt(90)
trace(0,5);
trace(...,);
trace(...);
trace(...);
trace(...);
trace(...);
```

Compléter ce code pour obtenir l'image :



E.501

Reproduire la figure ci-dessous qui est composée de 5 carrés dont les tailles sont successivement de 10 et de 20 et sont espacés de 15.



On utilisera le code suivant :

```
1 from turtle import *
2 def carre(a):
3   pendown()
4   forward(a)
5   left(90)
```

```
  forward(a)
  left(90)
  forward(a)
  left(90)
  forward(a)
  left(90)
  penup()
  forward(a+3)
for i in range(0,9):
    carre(10+i*2)
exitonclick()
```

501-code.py

E.416

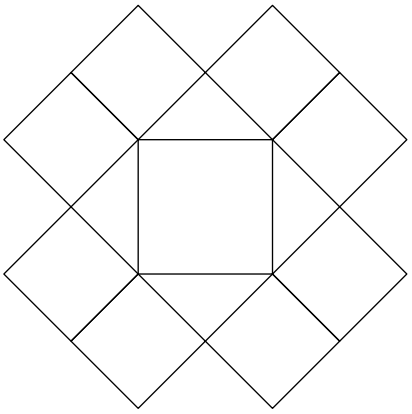
1 Saisissez le code ci-dessous :

```
from turtle import *
from math import sqrt
def carre(a,x,y):
  penup()
  goto(x,y)
  pendown()
  forward(a)
  left(90)
  forward(a)
  left(90)
  forward(a)
  left(90)
  forward(a)
  left(90)
```

```
  forward(a)
  left(90)
a=100
b=a*sqrt(2)/2
carre(a,0,0)
left(45)
carre(a,0,a)
```

416-code.py

2 À l'aide de ce code et de la fonction `carre()`, réaliser la figure ci-dessous :



Indication : dans la figure ci-dessous, le grand carré a pour côté 100 et les petits carrés ont pour côté $100 \times \frac{\sqrt{2}}{2}$.

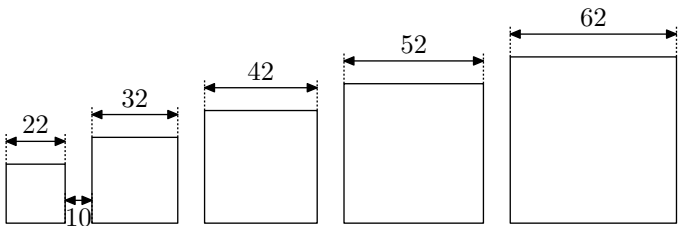
E.982

1 Saisissez le code ci-dessous :

```
from turtle import *
def carre(a):
    forward(a)
    left(90)
    forward(a)
    left(90)
    forward(a)
    left(90)
    forward(a)
    left(90)
carre(40)
carre(50)
carre(60)
```

982-code.py

2 Modifiez la fonction `carre()` pour obtenir l'affichage ci-dessous :



C. *Turtle - boucle*

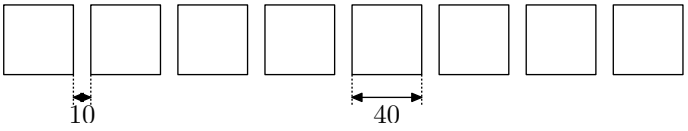
E.657

On considère le code suivant :

```
1 from turtle import *
2 def carre():
3     for i in range(4):
4         forward(10)
5         left(90)
6     forward(10)
7 for j in range(5):
8     print(j)
9     carre()
```

657-code.py

Modifier ce code pour obtenir le dessin suivant :



D. *Turtle - fonction et parametre*

E.442

1) Recopier le code ci-dessous :

```
1 from turtle import *
2 def carre(a):
3     for i in range(4):
4         forward(a)
5         left(90)
6 carre(10)
```

442-code.py

2) Modifier votre script pour obtenir l'affichage :

Indication : la figure est composée de 20 carrées.

E.443

1) Recopier le code ci-dessous :

```
1 from turtle import *
2 def triangle(a):
3     for i in range(3):
4         forward(a)
5         left(120)
6 triangle(10)
```

2) Modifier votre script pour obtenir l'affichage :

Indication : la figure est composée de 20 triangles.

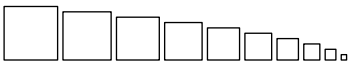
E.419

1) Modifiez votre code pour qu'il devienne :

```
1  from turtle import *
2  def carre(a):
3    pendown()
4    forward(a)
5    left(90)
6    forward(a)
7    left(90)
8    forward(a)
9    left(90)
10   forward(a)
11   left(90)
12   penup()
13   forward(a+3)
14 for i in range(0,9):
15     carre(10+i*2)
16 exitonclick()
```

419-code.py

2) Modifiez votre code pour qu'il affiche la figure suivante :



E. *Turtle - double boucle*

E.418

Utilisons les boucles pour créer un motif répétitif :

1) Saisissez et exécutez le code ci-dessous dans votre IDE/Python :

```
1  from turtle import *
2  def carre(a):
3    pendown()
4    forward(a)
5    left(90)
6    forward(a)
7    left(90)
8    forward(a)
9    left(90)
10   forward(a)
11   left(90)
12   penup()
13 speed(100)
14 carre(10)
15 forward(20)
16 carre(10)
17 forward(20)
18 carre(10)
19 exitonclick()
```

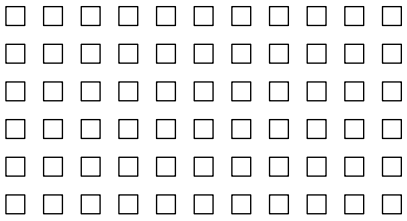
418-code.py

2) Remplacez les lignes 16 à 20 en :

```
for i in range(0,4):
  carre(10)
  forward(20)
```

puis exécutez le code.

3) Inspirez-vous de la ligne précédente pour réaliser le motif :



E.420

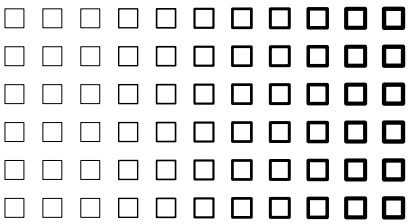
Nous allons ajouter un paramètre à la fonction `carre()` afin de modifier l'épaisseur du trait traçant le carré

1) Saisissez le code ci-dessous :

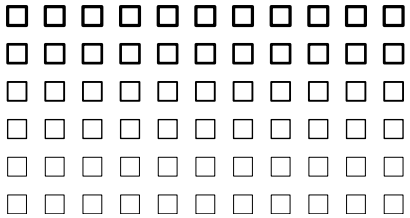
```
1  from turtle import *
2  def carre(a,w):
3    pensize(w)
4    pendown()
5    forward(a)
6    left(90)
7    forward(a)
8    left(90)
9    forward(a)
10   left(90)
11   forward(a)
12   left(90)
13   penup()
14 speed(100)
15 for i in range(0,10):
16     for j in range(0,4):
17         setposition(i*20,j*20)
18         carre(10,1)
19 carre(10,3)
20 exitonclick()
```

420-code.py

2) Modifier votre code pour obtenir :



3) Modifier votre code pour obtenir :



F. *Turtle - fonction et parametre*

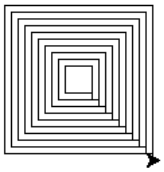
E.446

1) Recopier le code ci-dessous :

```
1  from turtle import *
2  def carre(a):
3      for i in range(4):
4          left(90)
5          forward(a)
6  carre(20)
7  carre(30)
8  carre(40)
```

446-code.py

2) Modifier votre script pour obtenir l'affichage :



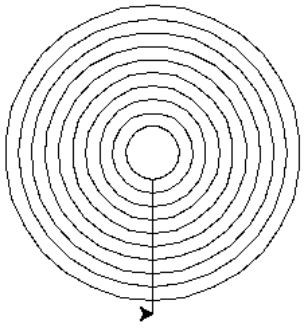
Indication : la figure est composée de 10 carrés dont le premier a pour côté 20 et les autres augmentent de 10 en 10.

E.445

1) Recopier le code ci-dessous :

```
1  from turtle import *
2  circle(20)
3  circle(30)
4  circle(40)
```

2) Modifier votre script pour obtenir l'affichage :



Indication : la figure est composée de 10 cercles dont le premier a pour rayon 20 et les autres augmentent de 10 en 10.

G. *Turtle - boucle et conditionnelle*

E.421

1 a Saisissez, puis exécutez le code ci-dessous :

```
from turtle import *
from random import *
for i in range(0,100):
  forward(10)
  if random()<0.5:
      left(90)
  else:
      right(90)
  print("x:",xcor(),"y:",ycor())
exitonclick()
```

421-code.py

b Comment peut-on expliquer la marche "*aléatoire*" de notre tortue.

2 Rajoutez le code nécessaire à votre programme pour que la tortue revienne au milieu lorsque celle-ci dépasse le cadre de la fenêtre.

E.502

Saisissez le code suivant :

```
from turtle import *
a=0
while True:
    if a==100:
        a=0
    dot(a*(100-a)*0.1, "white")
    a+=1
    dot(a*(100-a)*0.1, "orange")
```

502-code.py

H. *Turtle - interface utilisateur*

E.422

1 Recopier et exécuter le code ci-dessous :

```
import turtle
import random
a=50
posStylo=1
def up():
    turtle.setheading(90)
    turtle.forward(a)
def left():
    turtle.setheading(180)
    turtle.forward(a)
def space():
    global posStylo;
    if posStylo==1:
        turtle.penup()
        posStylo=0
    else:
        turtle.pendown()
        posStylo=1
turtle.listen()
turtle.onkey(up, "Up")
turtle.onkey(left, "Left")
turtle.onkey(space, "space")
turtle.mainloop()
```

422-code.py

2 Compléter le code ci-dessus pour que la "tortue" puisse bouger dans les 4 directions :

E.423

1 Recopier et testez le code ci-dessous :

```
from turtle import *
screen = Screen()
screen.setup(300,300)
def turnLeft():
    left(90);
    forward(10);
def turnRight():
    right(90)
    forward(10);
speed(0.5)
listen()
onkey(turnLeft, "Left")
onkey(turnRight, "Right")
while True:
    forward(10);
    if xcor()>150:
        reset()
        heading()
        speed(0.5)
exitonclick()
```

423-code.py

2 Faite en sorte que le jeu redémarre lorsque l'un des quatre côtés est atteint

I. *Autre*

E.444

Créez un code :

- demandant à l'utilisateur l'aire d'un carré en demandant "*Quelle est l'aire de votre carré*"
- et renvoyant dans la console la longueur approchée du côté du carré.

Rappel :

- On importe la bibliothèque `math` à l'aide de l'instruction :
  `import math`
- On détermine la racine carrée d'un nombre `x` par l'instruction :
  `math.sqrt(x)`

E.447

Créez un code :

- demandant à l'utilisateur le prix d'un objet
- et renvoyant dans la console le prix de l'objet augmenté de 12 %

Rappel :
Une valeur a augmentant de b % a pour nouvelle valeur :
$a \times \left(1 + \dfrac{b}{100}\right)$

E.656

On considère le pseudo-code ci-dessous :

```
a ← 3
a ← a+6
c ← 2
a ← c+a
Afficher a
Afficher b
```

Donner les valeurs affichées par l'exécution de ce code.