Voici des extraits du DNB 2018 : la version Scratch donnée au DNB, une proposition avec la tortue de Python *D'après les sujets de l'APMEP*

1. Amérique du Nord, juin 2018, exercice 4

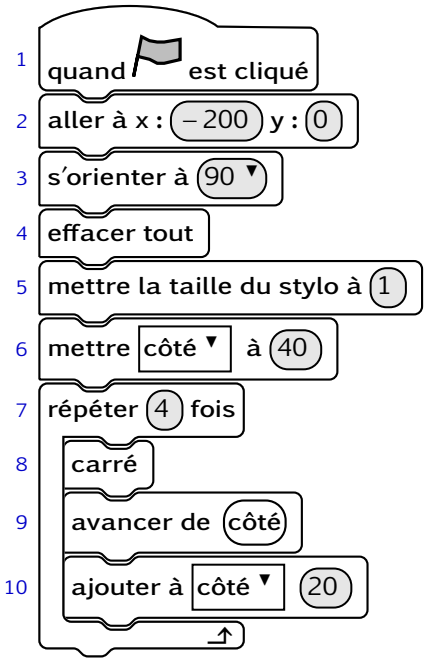
- module turtle
- import ... as ...
- variable globale
- définition d'une fonction
- mainloop()

14 points (sur 100)

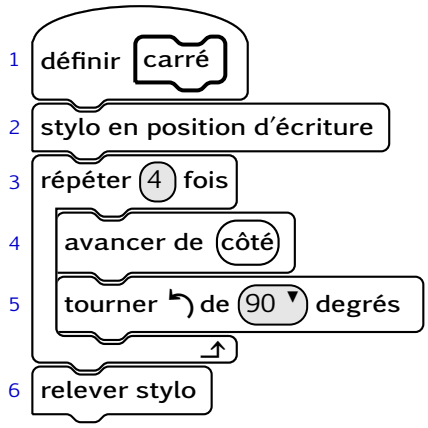
Dans cet exercice, aucune justification n'est attendue.

Simon travaille sur un programme. Voici des copies de son écran :

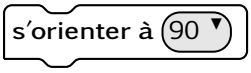
Script principal



Bloc carré



Information

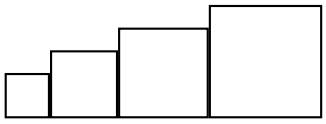
L'instruction 

signifie qu'on se dirige vers la droite.

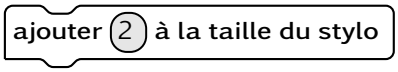
1. Il obtient le dessin ci-contre.

a) D'après le script principal, quelle est la longueur du côté du plus petit carré dessiné ?

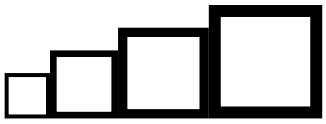
b) D'après le script principal, quelle est la longueur du côté du plus grand carré dessiné ?



dessin de la question 1

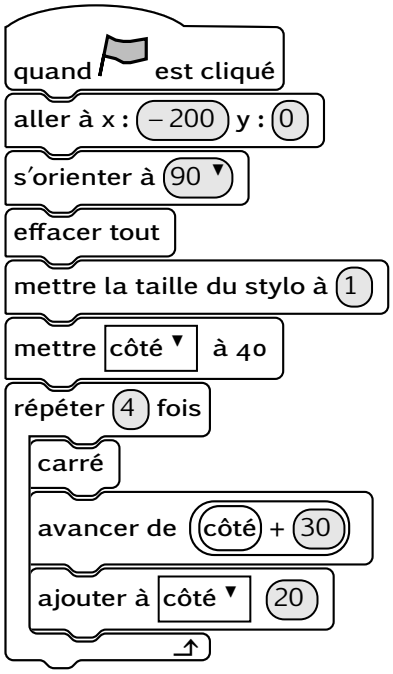
2. Dans le script principal, où peut-on insérer l'instruction 

de façon à obtenir le dessin ci-contre ?



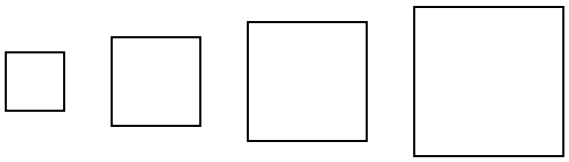
dessin de la question 2

3. On modifie maintenant le script principal pour obtenir celui qui est présenté ci-dessous :

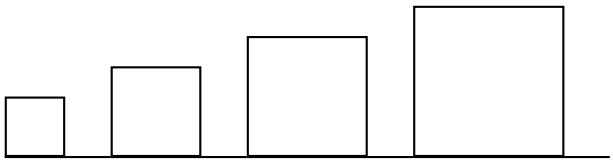


Parmi les dessins ci-dessous, lequel obtient-on ?

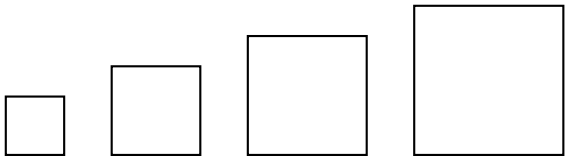
Dessin 1



Dessin 2



Dessin 3



```
# -*- coding:utf8 -*-
# python 3

import turtle as tl

def carre():
    tl.pendown()
    for _ in range(4):
        tl.forward(COTE)
        tl.left(90)
    tl.penup()

global COTE
tl.setposition(-200, 0)
tl.clear()
tl.pensize(1)
COTE = 40
for i in range(4):
    carre()
    tl.forward(COTE)
    COTE = COTE + 20
tl.mainloop()
```

Adapter le programme pour obtenir les réponses aux questions 2 et 3.

2. Métropole, juin 2018, exercice 6

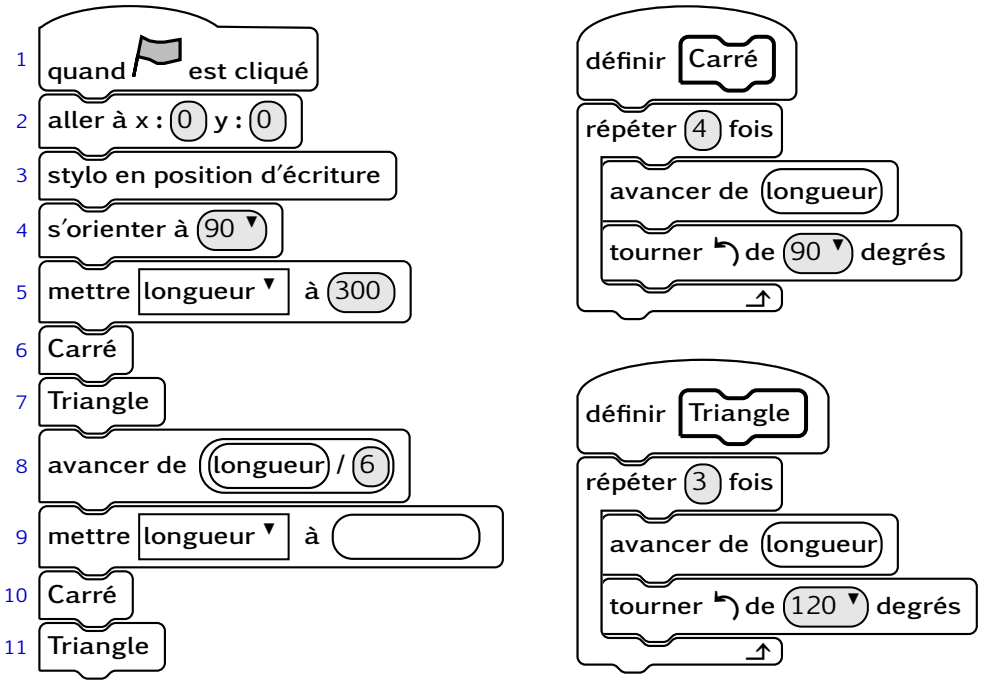
16 points (sur 100)

- notion de fonction avec une variable

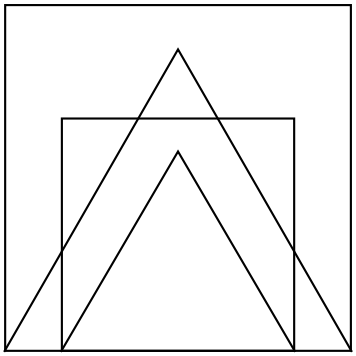
Les longueurs sont en pixels.

L'expression « s'orienter à 90 » signifie que l'on s'oriente vers la droite.

On donne le programme suivant :



1. On prend comme échelle 1 cm pour 50 pixels.
   a) Représenter sur votre copie la figure obtenue si le programme est exécuté jusqu'à la ligne 7 comprise.
   b) Quelles sont les coordonnées du stylo après l'exécution de la ligne 8 ?
2. On exécute le programme complet et on obtient la figure ci-contre qui possède un axe de symétrie vertical. Recopier et compléter la ligne 9 du programme pour obtenir cette figure.
3. a) Parmi les transformations suivantes, translation, homothétie, rotation, symétrie axiale, quelle est la transformation géométrique qui permet d'obtenir le petit carré à partir du grand carré ? Préciser le rapport de réduction.
   b) Quel est le rapport des aires entre les deux carrés dessinés ?



- module random
- module math

3. Pondichéry, mai 2018, exercice 5

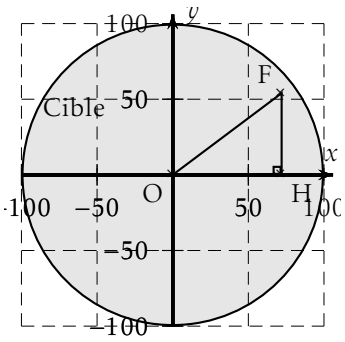
20 points (sur 100)

Dans tout l'exercice l'unité de longueur est le mm.

On lance une fléchette sur une plaque carrée sur laquelle figure une cible circulaire (en gris sur la figure), Si la pointe de la fléchette est sur le bord de la cible, on considère que la cible n'est pas atteinte.

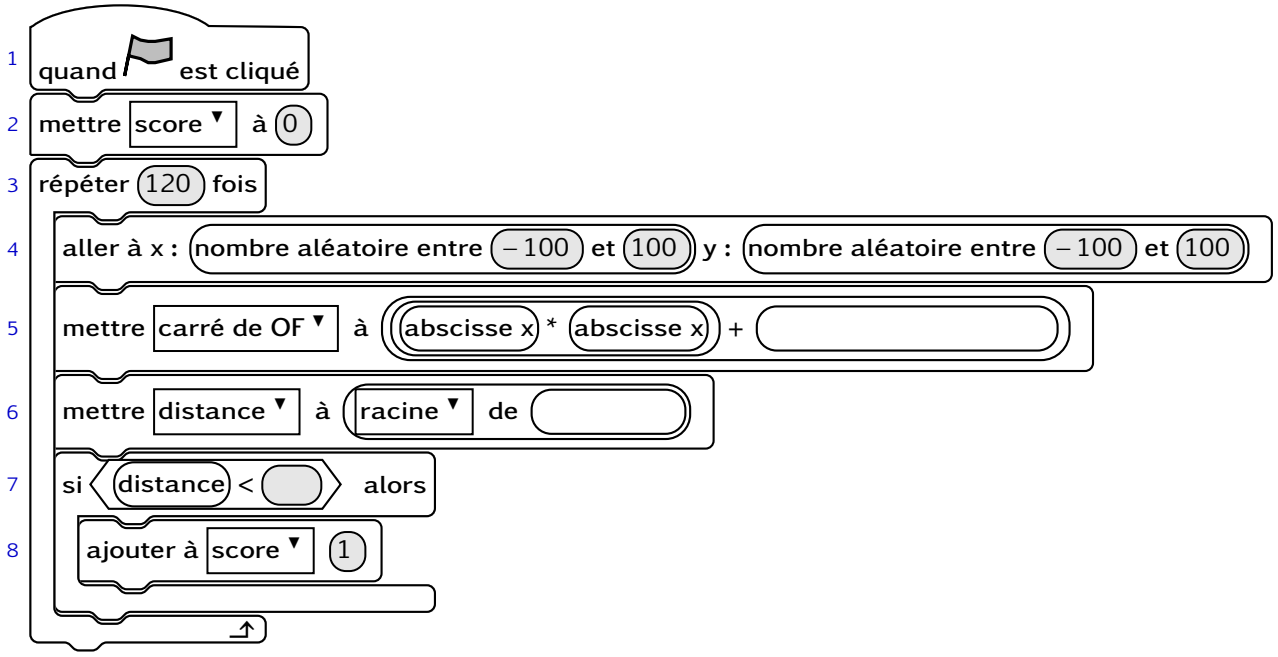
On considère que cette expérience est aléatoire et l'on s'intéresse à la probabilité que la fléchette atteigne la cible.

- La longueur du côté de la plaque carrée est 200.
- Le rayon de la cible est 100.
- La fléchette est représentée par le point F de coordonnées $(x\,;y)$ où x et y sont des nombres aléatoires compris entre -100 et 100.



1. Dans l'exemple ci-dessus, la fléchette F est située au point de coordonnées (72 ; 54). Montrer que la distance OF, entre la fléchette et l'origine du repère est 90.
2. D'une façon générale, quel nombre ne doit pas dépasser la distance OF pour que la fléchette atteigne la cible ?

3. On réalise un programme qui simule plusieurs fois le lancer de cette fléchette sur la plaque carrée et qui compte le nombre de lancers atteignant la cible. Le programmeur a créé trois variables nommées :
   carré de OF, distance et score.
   a) Lorsqu'on exécute ce programme, combien de lancers sont simulés ?
   b) Quel est le rôle de la variable **score** ?
   c) Compléter et recopier sur la copie uniquement les lignes 5, 6 et 7 du programme afin qu'il fonctionne correctement.
   d) Après une exécution du programme, la variable **score** est égale à 102.
   À quelle fréquence la cible a-t-elle été atteinte dans cette simulation ?
   Exprimer le résultat sous la forme d'une fraction irréductible.
4. On admet que la probabilité d'atteindre la cible est égale au quotient : aire de la cible divisée par aire de la plaque carrée.
   Donner une valeur approchée de cette probabilité au centième près.




```
1  # -*- coding:utf8 -*-
2  # python 3
3  from random import uniform
4  from math import sqrt
5
6  score = 0
7  for _ in range(120):
8      x, y = uniform(-100, 100), uniform(-100, 100)
9      carreOF = x*x + y*y
10     distance = sqrt(carreOF)
11     if distance < 100:
12         score = score + 1
13 print(score)
```

Pour compléter

- module mathplotlib
- module numpy
- listes par compréhension ; longueur d'une liste ; ajouter des éléments à une liste

Une version plus « lycée »

```
# -*- coding:utf8 -*-
# python 3

from random import uniform

n, s = 1000, 0
for _ in range(n):
    x, y = uniform(-100, 100), uniform(-100, 100)
    if x**2 + y**2 <= 10000:
        s = s + 1

print("fréquence = ", s/n)
```

Avec graphique

```
# -*- coding:utf8 -*-
# python 3

from random import uniform
from matplotlib import pyplot as plt
from math import sqrt

n, s = 4000, 0
for _ in range(n):
    x, y = uniform(-100, 100), uniform(-100, 100)
    if x**2 + y**2 <= 10000: #10000=100**2
        plt.plot(x, y, 'ko')
        s = s + 1
    else:
        plt.plot(x, y, 'wo') #si white ne donne rien : changer la
    ↪ couleur

# une méthode possible pour afficher le cercle
abscisses = [x for x in range(-100, 101, 2)] #x de [-100;101[ par
    ↪ pas de 2
ordonnees = [sqrt(10000 - x**2) for x in abscisses]
plt.plot(abscisses, ordonnees,'b-',linewidth=3)
ordonnees = [-y for y in ordonnees]
plt.plot(abscisses, ordonnees, 'b-', linewidth=3)

plt.title("n = " + str(n) + "; aire disque = " + str(s/n * 40000))
# aire du carré = 200**2 = 40000
plt.axis('equal')
plt.axis('on')
```

help(plt.plot) donne les informations suivantes :

```
================    ========================
character           description
================    ========================
    '-'             solid line style
    '--'            dashed line style
    '-.'            dash-dot line style
    ':'             dotted line style
    '.'             point marker
    ','             pixel marker
    'v'             triangle_down marker
    'o'             circle marker
 ...
================    ========================

==========  ========
character   color
==========  ========
    'b'         blue
    'g'         green
    'r'         red
    'c'         cyan
...
==========  ========
```

```
28 plt.show()
```

Accélération grâce aux listes

```
1  # -*- coding:utf8 -*-
2  # python 3
3
4  from random import uniform
5  from matplotlib import pyplot as plt
6  import numpy as np
7
8  # on accelère l'affichage en travaillant sur des listes de points
9  # le package <numpy> contient la fonction <linspace>
10 # en mode console : import numpy as np puis help(np.linspace)
11
12 n = 10000
13 x_in, x_out, y_in, y_out = [], [], [], [] # liste des coordonnées
14 for _ in range(n):
15     x, y = uniform(-100, 100), uniform(-100, 100)
16     if x**2 + y**2 < 10000:
17         x_in, y_in = x_in + [x], y_in + [y]
18     else:
19         x_out, y_out = x_out + [x], y_out + [y]
20
21 # Affichage des points dans le repère
22 plt.plot(x_in, y_in, marker='o', color='black', linestyle='none')
23 plt.plot(x_out, y_out, marker='o', color='white', linestyle='none
       ↪')
24
25 # Affichage du cercle
26 x = np.linspace(-100, 100, 200)
27 y = np.sqrt(10000 - x**2)
28 plt.plot(x, y, "r-", linewidth=3)
29 plt.plot(x, -y, "r-", linewidth=3)
30 plt.title("n = " + str(n) + " et aire disque = " + str(len(x_in)
       ↪/ n * 40000))
31 plt.axis('equal')
32 plt.axis('on')
33 plt.grid('on')
34 plt.show()
```

Méthode de Monte-Carlo

On cherche une représentation de l'inéquation [1] :

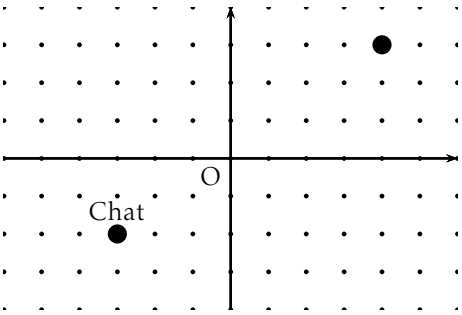
$$(x^2+y^2-1)^3 < 4x^2y^3$$

Adapter les programmes précédents pour trouver une représentation.

4. Amérique du Nord, juin 2017

- module turtle
- Notions de scripts se déroulant en parallèles !
- événements contrôlés au clavier

L'image représente la position obtenue au déclenchement du bloc départ d'un programme de jeu.

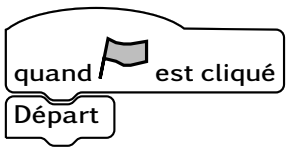


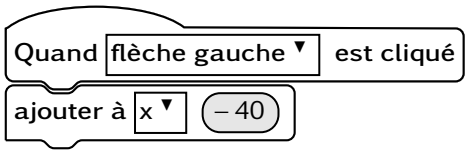
L'arrière-plan est constitué de points espacés de 40 unités.
Dans cette position, le chat a pour coordonnées $(-120 \quad -80)$.
Le but du jeu est de positionner le chat sur la balle.

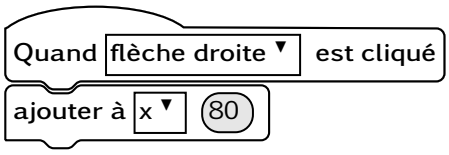
1. Quelles sont les coordonnées du centre de la balle représentée dans cette position ?
2. Dans cette question, le chat est dans la position obtenue au déclenchement du bloc départ.
   Le script du lutin « chat » qui se déplace est donné ci-après.
   a) Expliquez pourquoi le chat ne revient pas à sa position de départ si le joueur appuie sur la touche → puis sur la touche ←.
   b) Le joueur appuie sur la succession de touches suivante : → → ↑ ← ↓.
   Quelles sont les coordonnées x et y du chat après ce déplacement ?
   c) Parmi les propositions de succession de touches ci-dessous, laquelle permet au chat d'atteindre la balle ?

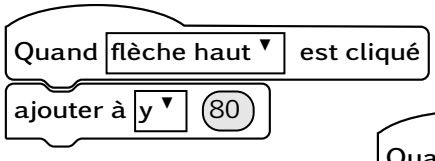
Déplacement 1	Déplacement 2	Déplacement 3
→→→→→→→↑↑↑↑↑	→→→↑↑↑→↓←	↑→↑→↑→→↓↓

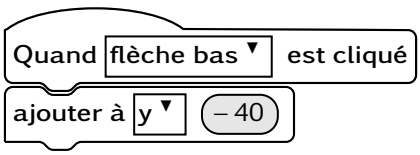
3. Que se passe-t-il quand le chat atteint la balle ?

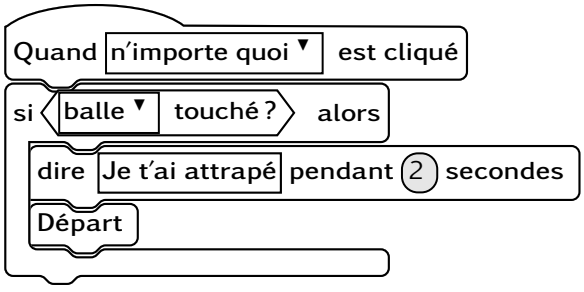












1 – GeoGebra ne sait représenter *que* l'égalité

```python
# -*- coding:utf8 -*-
# python3

from turtle import *

def fleche_g():
    setx(xcor() - 40)
    appui_nimporte_quoi()

def fleche_d():
    setx(xcor() + 80)
    appui_nimporte_quoi()

def fleche_h():
    sety(ycor() + 80)
    appui_nimporte_quoi()

def fleche_b():
    sety(ycor()-40)
    appui_nimporte_quoi()

def appui_nimporte_quoi():
    if position() == (160, 120):
        write("je t'ai attrapé",False)
        ontimer(depart, 2000)

def depart():
    clear()
    hideturtle()
    penup()
    setposition(160, 120)
    dot(10,"red")
    setposition(-120, -80)
    showturtle()
    pendown()

depart()
listen()  #Pour "écouter"
onkeypress(fleche_g, "Left")   #
    ↪Touche gauche
onkeypress(fleche_d, "Right")  #
    ↪Touche droite
onkeypress(fleche_h, "Up")     #
    ↪Touche haut
onkeypress(fleche_b, "Down")   #
    ↪Touche bas
onkeypress(appui_nimporte_quoi) #
    ↪Toutes les touches libres sont
                              # associé
    ↪es à appui_quelconque
mainloop()
```