Nom :	Prénom :	Groupe :

Durée : 90’ Documents, calculettes, téléphones et ordinateurs interdits

1 Calcul de π

Définir une fonction qui calcule π à l’ordre n selon la formule :

$$\pi = 2 \cdot \frac{4}{3} \cdot \frac{16}{15} \cdot \frac{36}{35} \cdot \frac{64}{63} \cdots = 2 \prod_{k=1}^{n} \frac{4k^2}{4k^2 - 1}$$

2 Conversion base $b \rightarrow$ décimal

Définir une fonction qui calcule la valeur décimale n d'un entier positif t codé en base b.

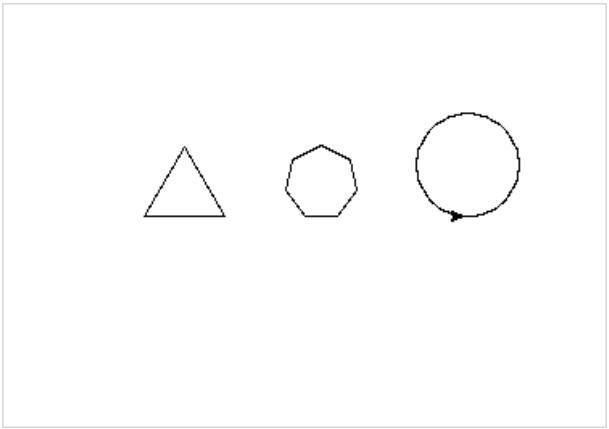
Exemples :
```
b = 2   t = [0, 0, 1, 0, 1, 1, 1] → n = 23
b = 5   t = [0, 0, 0, 4, 3]       → n = 23
b = 21  t = [1, 2]                → n = 23
b = 25  t = [0, 0, 0, 0, 0, 23]   → n = 23
```

3 Polygones réguliers

Définir une fonction `polygone(x0,y0,d,n)` qui trace un polygone régulier à `n` côtés de longueur `d` à partir du point de coordonnées (x_0, y_0). On utilisera les instructions de tracé *à la Logo* (voir annexe A page 9).

Exemples :

```
>>> from turtle import *
>>> polygone(-100,0,50,3)
>>> polygone(0,0,20,7)
>>> polygone(100,0,2,100)
```

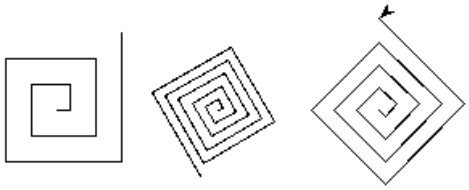


4 Spirales rectangulaires

Définir une fonction `spirale(n,x0,y0,a0,dr)` qui trace une spirale rectangulaire à `n` côtés à partir du point de coordonnées (x_0, y_0) et avec une orientation initiale a_0. `dr` représente l'incrément de longueur d'un côté de la spirale à son suivant immédiat (le premier côté ayant pour longueur `dr`). On utilisera les instructions de tracé *à la Logo* (voir annexe A page 9).

Exemples :

```
>>> from turtle import *
>>> spirale(10,-100,0,0,8)
>>> spirale(20,0,0,30,3)
>>> spirale(15,100,0,-45,5)
```



5 Portée des variables

On considère les fonctions `f`, `g` et `h` suivantes :

```
def f(x):
    x = 3*x
    print('f', x)
    return x
```

```
def g(x):
    x = 2*f(x)
    print('g', x)
    return x
```

```
def h(x):
    x = g(f(x))
    print('h', x)
    return x
```

Qu'affichent les appels suivants ?

1.
```
>>> x = 2
>>> print(x)

>>> y = f(x)
>>> print(x)

>>> z = g(x)
>>> print(x)

>>> t = h(x)
>>> print(x)
```

1.
```
>>> x = 2
>>> print(x)

>>> x = f(x)
>>> print(x)

>>> x = g(x)
>>> print(x)

>>> x = h(x)
>>> print(x)
```

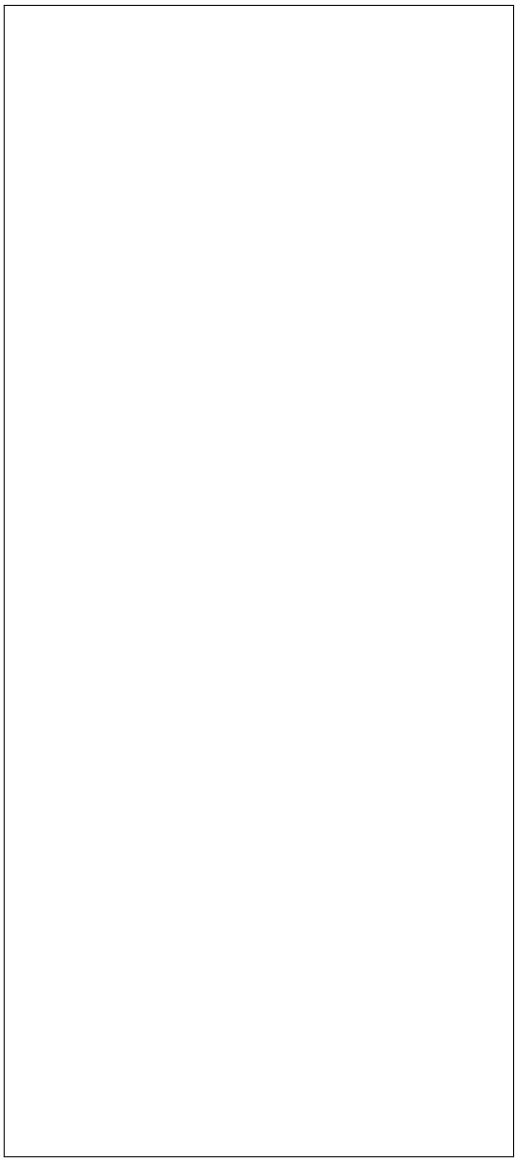
6 Recherche d'un élément dans un tableau

Définir une fonction itérative `recherche(t,x,k)` pour la recherche de la $k^{ème}$ occurrence d'un élément `x` dans un tableau `t` à partir de la fin du tableau. La fonction retourne un doublet `(ok,i)` où `ok` est une variable booléenne égale à `True` si la $k^{ème}$ occurrence de `x` existe dans `t` (`i` est alors le rang de cette occurrence dans `t`) ; `ok` vaut `False` sinon (la valeur de `i` n'a alors pas de signification : elle est sans importance).

Exemples :
```
>>> recherche([1,2,1,3,4,1,5],1,2)
(True, 2)
>>> recherche([1,2,1,3,4,1,5],1,4)
(False, -1)
```

7 Exécution d'une fonction de tri

Qu'affiche l'appel `f([5,1,4,7,10])` où `f` est la fonction itérative définie ci-contre ?



```
#---------------------------------------
def f(t):
#---------------------------------------
    assert type(t) is list

    for i in range(len(t)):
        m = i
        for j in range(i+1,len(t)):
            if t[j] > t[m]: m = j
        t[i],t[m] = t[m],t[i]
        print(i, t) #---------- affichage

    return t
#---------------------------------------
```

8 Exécution d'une fonction de sélection

Qu'affiche l'appel `g(h,[5,1,4,7,10])` où `g` et `h` sont les fonctions définies ci-contre ?

```
#---------------------------------------
def h(x):
#---------------------------------------
    assert type(x) is int

    return x%2 == 0
#---------------------------------------


#---------------------------------------
def g(p,t):
#---------------------------------------
    assert type(t) is list

    i = 0
    while i < len(t):
        if p(t[i]) == True:
            del t[i]
        else: i = i + 1
        print(i, t)  #--------- affichage
    return t
#---------------------------------------
```

A Annexe : instructions Logo

> Logo is the name for a philosophy of education and a continually evolving family of programming languages that aid in its realization (Harold Abelson, Apple Logo, 1982). *This statement sums up two fundamental aspects of Logo and puts them in the proper order. The Logo programming environments that have been developed over the past 28 years are rooted in constructivist educational philosophy, and are designed to support constructive learning. [...] Constructivism views knowledge as being created by learners in their own minds through interaction with other people and the world around them. This theory is most closely associated with Jean Piaget, the Swiss psychologist, who spent decades studying and documenting the learning processes of young children.*
>
> **Logo Foundation :** `http ://el.media.mit.edu/logo-foundation`

On suppose connues les procédures de tracés géométriques *à la Logo* :

`degrees()` fixe l'unité d'angle en degrés

`radians()` fixe l'unité d'angle en radians

`reset()` efface l'écran et réinitialise les variables

`clear()` efface l'écran

`forward(d)` avance d'une distance d

`backward(d)` recule d'une distance d

`left(a)` tourne sur la gauche d'un angle a

`right(a)` tourne sur la droite d'un angle a

`up()` lève le crayon

`down()` abaisse le crayon

`goto(x,y)` déplace le crayon à la position (x, y)

`setheading(a)` oriente la tortue selon un angle a avec l'axe horizontal

`towards(x,y)` donne l'angle entre la direction courante et la direction pointant vers le point de coordonnées (x, y)

`circle(r)` trace un cercle de rayon r

`circle(r,a)` trace un arc de cercle de rayon r et d'angle au sommet a