1 Calcul de π

```python
# -*- coding: utf-8 -*-

def calculPi(n):
    """
    y = calculPi(n)
    calcul de pi à l'ordre n

    >>> from math import fabs, pi
    >>> fabs(pi - calculPi(1)) < 1.
    True
    >>> fabs(pi - calculPi(1000000)) < 1.e-6
    True
    """
    assert type(n) is int and n >= 0

    y = 2.
    for k in range(1,n+1):
        u = 4*k*k
        y = y*u/(u-1)
    return y

#--------------------------------------------------------------------------
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

2 Conversion base $b \rightarrow$ décimal

```python
# -*- coding: utf-8 -*-

def conversion(code,b=2):
    """
    n = conversion(code,b)
    entier décimal qui représente le code en base b

    >>> conversion([0,0,1,0,1,1,1],2)
    23
    >>> conversion([0, 0, 0, 4, 3],5)
    23
    >>> conversion([1,2],21)
    23
    >>> conversion([0,0,0,0,0,23],25)
    23
    """
    assert type(b) is int and b > 1
    assert type(code) is list

    n = 0
    for i in range(len(code)):
        n = n + (b**i)*code[len(code)-1-i]

    return n

#--------------------------------------------------------------------------
```

```
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

3 Polygones réguliers

```
# -*- coding: utf-8 -*-

from turtle import *

#-------------------------------------------------------------------------
def polygone(n,d,x=0,y=0):
    """
    trace un polygone régulier à n côtés de longueur d
    à partir du point de coordonnées (x,y)

    >>> for i in range(3,10): polygone(i,100,-150,0)
    """
    up()
    goto(x,y)
    down()
    for i in range(n):
        forward(d)
        left(360./n)
    return

#-------------------------------------------------------------------------
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

4 Spirales rectangulaires

```
# -*- coding: utf-8 -*-

from turtle import *

#-------------------------------------------------------------------------
def spirale(n,x0,y0,a0,dr):
    """
    spirale(n,x0,y0,a0,dr)
    trace une spirale rectangulaire à n côtés à partir du point de
    coordonnées (x0,y0) et avec une orientation initiale a0.
    dr représente l'incrément de longueur d'un côté de la spirale
    à son suivant immédiat (le premier côté ayant pour longueur dr).

    >>> spirale(10,-100,0,0,8)
    >>> spirale(20,0,0,30,3)
    >>> spirale(15,100,0,-45,5)
    """
    assert type(n) is int and n >= 0
    assert type(x0) is int and type(y0) is int
    assert type(a0) is int
    assert type(dr) is int and dr >= 0
```

```
22
23      up()
24      goto(x0,y0)
25      setheading(a0)
26      down()
27
28      d = 0
29      for i in range(n):
30          d = d + dr
31          forward(d)
32          left(90)
33      return
34
35  #-----------------------------------------------------------------------
36  if __name__ == "__main__":
37      import doctest
38      doctest.testmod()
```

5 Portée des variables

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

>>> y = f(x)
>>> print(x)
f 6
2

>>> z = g(x)
>>> print(x)
f 6
g 12
2

>>> t = h(x)
>>> print(x)
f 6
f 18
g 36
h 36
2
```

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

>>> x = f(x)
>>> print(x)
f 6
6

>>> x = g(x)
>>> print(x)
f 18
g 36
36

>>> x = h(x)
>>> print(x)
f 108
f 324
g 648
h 648
648
```

6 Recherche d'un élément dans un tableau

```
1   # -*- coding: utf-8 -*-
2
3   def recherchekieme(t,x,k):
4       """
5       ok,i = recherchekieme(t,x,k)
6       recherche la kième occurence de x dans la liste t en commençant
7       par la fin de la liste.
8       ok == True si x a été trouvé à l'indice i, False sinon
9
10      >>> recherchekieme([1,2,1,3,4,1,5],1,2)
11      (True, 2)
```

```
12      >>> recherchekieme([1,2,1,3,4,1,5],1,4)
13      (False, -1)
14      """
15      assert type(t) is list
16      assert type(k) is int and k > 0
17
18      ok, i = False, len(t) - 1
19      occur = 0
20      while i >= 0 and not ok:
21              if t[i] == x:
22                  occur = occur + 1
23                  if occur == k: ok = True
24                  else: i = i - 1
25              else: i = i - 1
26
27      return ok, i
28
29  #-------------------------------------------------------------------------
30  if __name__ == "__main__":
31      import doctest
32      doctest.testmod()
```

7 Exécution d'une fonction de tri

```
>>> f([5,1,4,7,10])
0 [10, 1, 4, 7, 5]
1 [10, 7, 4, 1, 5]
2 [10, 7, 5, 1, 4]
3 [10, 7, 5, 4, 1]
4 [10, 7, 5, 4, 1]
```

8 Exécution d'une fonction de sélection

```
>>> g(h,[5,1,4,7,10])
1 [5, 1, 4, 7, 10]
2 [5, 1, 4, 7, 10]
2 [5, 1, 7, 10]
3 [5, 1, 7, 10]
3 [5, 1, 7]
```