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

```
23
24      return n
25
26  #-------------------------------------------------------------------------
27  if __name__ == "__main__":
28      import doctest
29      doctest.testmod()
```

3 Spirales rectangulaires

```
1   # -*- coding: utf-8 -*-
2
3   from turtle import *
4
5   #-------------------------------------------------------------------------
6   def spirale(n,x0,y0,a0,dr):
7       """
8       spirale(n,x0,y0,a0,dr)
9       trace une spirale rectangulaire à n côtés à partir du point de
10      coordonnées (x0,y0) et avec une orientation initiale a0.
11      dr représente l'incrément de longueur d'un côté de la spirale
12      à son suivant immédiat (le premier côté ayant pour longueur dr).
13
14      >>> spirale(10,-100,0,0,8)
15      >>> spirale(20,0,0,30,3)
16      >>> spirale(15,100,0,-45,5)
17      """
18      assert type(n) is int and n >= 0
19      assert type(x0) is int and type(y0) is int
20      assert type(a0) is int
21      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()
```

4 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
```

5 Exécution d'une fonction itérative

Il s'agit de l'algorithme du crible d'Eratosthène qui trouve tous les nombres premiers inférieurs à un certain entier n ($n = 10$ dans l'exemple ci-contre).

```
>>> f(list(range(2,10)))
0 2 [2, 3, 4, 5, 6, 7, 8, 9]
0 2 [2, 3, 5, 6, 7, 8, 9]
0 3 [2, 3, 5, 6, 7, 8, 9]
0 3 [2, 3, 5, 7, 8, 9]
0 4 [2, 3, 5, 7, 8, 9]
0 4 [2, 3, 5, 7, 9]
0 5 [2, 3, 5, 7, 9]
1 3 [2, 3, 5, 7, 9]
1 4 [2, 3, 5, 7, 9]
1 4 [2, 3, 5, 7]
2 4 [2, 3, 5, 7]
[2, 3, 5, 7]
```

6 Exécution d'une fonction récursive

Il s'agit de l'algorithme d'Euclide qui détermine le pgcd de 2 entiers (12 et 18 dans l'exemple ci-contre).

```
>>> f(12,18)
6 0
12 6
18 12
12 18
6
```