In [1]:
```
help (eval)
```

```
Help on built-in function eval in module builtins:

eval(source, globals=None, locals=None, /)
    Evaluate the given source in the context of globals and locals.

    The source may be a string representing a Python expression
    or a code object as returned by compile().
    The globals must be a dictionary and locals can be any mapping,
    defaulting to the current globals and locals.
    If only globals is given, locals defaults to it.
```

In [5]:
```
def media (*arg):
    return round (sum(arg) / len(arg))

print (media(1,2,3,4,5,6,7,8,9))
```

```
5
```

In [7]:
```
def junta_items (nombre_archivo, separador, *arg) :
    f = open(nombre_archivo, "a")
    f.write(separador.join(arg) +'\n')
    f.close()

junta_items("./test.txt", "; ", 'Carles', '50', '08034')
```

In [11]:
```
def doblar (num) :
    return (num*2)

x = doblar

print (x(4))
```

```
8
```

In [17]:
```
def doblar (num) :
    return (num*2)

x = doblar

x = lambda num :num*2

y = lambda num : num**2+5

print (x(3))
```

```
6
```

In [18]:
```
print (y(3))
```

```
14
```

In [20]:
```
# def mayor (a,b) :
#     return (a>b)

z = lambda a,b : a>b

print (z(3,4))
```

```
False
```

In [21]:
```
l =[c for c in range (1,10) ]
print(l)
l =[chr(c+96) for c in range (1,10) ]
print(l)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
```

In [25]:
```
cadena = "11.1 \n11.2 \n11.3 \n11.4 \n11.5 \n11.9 \n11.10 \n11.11 \n"
print(cadena)
d = { str(c):chr(ord(str(c))+48) for c in range (1,10) }
# d['a0'] = 'j'
# d['a1'] = 'k'
for item in d :
    cadena = cadena.replace('.'+item, '.'+d[item])
print(cadena)
```

```
11.1
11.2
11.3
11.4
11.5
11.9
11.10
11.11

11.a
11.b
11.c
11.d
11.e
11.i
11.a0
11.a1
```

In [24]:
```
d = { str(c):chr(ord(str(c))+48) for c in range (1,10) }
print(d)
```

```
{'1': 'a', '2': 'b', '3': 'c', '4': 'd', '5': 'e', '6': 'f', '7': 'g', '8': 'h', '9
': 'i'}
```

In [2]:
```
def cuenta_atras(num):
#--------------- ejemplo de recursiva
    num -= 1
    if num > 0:
        print(num)
        cuenta_atras(num)
    else:
        print("BOOOOOOOOM !!!!!!")
        print(">>Fin de la función ", num)
    print ("retorno de", num+1)
#---------------------proceso principal
#
cuenta_atras(5)
```

```
4
3
2
1
BOOOOOOOOM !!!!!!
>>Fin de la función  0
retorno de 1
retorno de 2
retorno de 3
retorno de 4
retorno de 5
```

In [1]:
```
import turtle

miTortuga = turtle.Turtle()
miVentana = turtle.Screen()

def dibujarEspiral(miTortuga, longitudLinea):
    if longitudLinea > 0:
        miTortuga.forward(longitudLinea)
        miTortuga.right(90)
        dibujarEspiral(miTortuga,longitudLinea-5)
#-------------------------------------------------Proceso principal

dibujarEspiral(miTortuga,100)
miVentana.exitonclick()
```

In [ ]: