Sample solutions for Exercise 2

Course: Scientific Programming (Python) by Bálint Aradi, University of Bremen

Fibonacci numbers (#3)

In [1]:
```
def fibonacci(nterms):
    """Generates a list with Fibonacci numbers.

    Args:
        nterms: Nr. of Fibonacci terms to generate.

    Returns:
        List containing the first nterms Fibonacci numbers.
    """
    if nterms < 1:
        fibo = []
    elif nterms == 1:
        fibo = [1,]
    else:
        fibo = [1, 1]
        for _ in range(2, nterms):
            fibo.append(fibo[-2] + fibo[-1])
    return fibo
```

In [2]:
```
fibo15 = fibonacci(15)
```

In [3]:
```
fibo15
```

Out[3]:
```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]
```

Aligned printing of ordered numbers

In [4]:
```
def print_ordered_ints(values):
    """Prints enumerated integer values with alignment.

    Args:
        values: List containing the integers to print.
    """
    indwidth = len(f"{len(values)}")
    valwidth = len(f"{values[-1]}")
    for ival, val in enumerate(values):
        print(f"{ival + 1:{indwidth}d}: {val:{valwidth}d}")
```

In [5]:
```
print_ordered_ints(fibo15)
```

```
 1:   1
 2:   1
 3:   2
 4:   3
 5:   5
 6:   8
 7:  13
 8:  21
 9:  34
10:  55
11:  89
12: 144
13: 233
14: 377
15: 610
```

Path plotting

In [6]:
```
import turtle
```

In [7]:
```
def plot_path(tt, path):
    """Plots a path with turtle graphics.

    Args:
        tt: Turtle to use.
        path: Points marking the path. It must be a list of (x, y) tuples
            the start point, the intermediate points and the end point of
    """
    tt.up()
    tt.goto(path[0])
    tt.down()
    for point in path[1:]:
        tt.goto(*point)
```

In [8]:
```
def plot_paths(tt, paths):
    """Plots a collection of paths turtle graphics.

    Args:
        tt: Turtle to use.
        paths: List of paths, where each path is a list of (x,y) tuples c
            the start point, the intermediate points and end points of th
    """
    for path in paths:
        plot_path(tt, path)
```

In [9]:
```
tt = turtle.Turtle()
```

In [10]:
```
tt.clear()
paths = [[(-50, -50), (50, -50), (50, 50), (-50, 50), (-50, -50)],
         [(-100, -100), (100, -100), (100, 100), (-100, 100), (-100, -100
plot_paths(tt, paths)
```

In [11]:
```
def vasarelly_star_paths(ngrids, incr):
    """Returns the paths for a Vasarelly-star.

    Args:
        ngrids: Nr. of axis grid points in each quadrant.
        incr: Inrement between grid points.

    Returns:
        List of paths, where each path is a list of (x, y) tuples.
    """
    size = incr * ngrids
    paths = []
    paths.append([(-size, 0), (size, 0)])  # x-axis
    paths.append([(0, -size), (0, size)])  # y-axis
    for ii in range(ngrids):
        xx = (ii + 1) * incr
        yy = size - xx
        paths.append([(xx, 0), (0, yy), (-xx, 0), (0, -yy), (xx, 0)])
    return paths
```

In [12]:
```
tt.clear()
paths = vasarelly_star_paths(10, 20)
plot_paths(tt, paths)
```

In [ ]:
```
```