Python training - lab 11

Unpacking operators: *, **

```
def func(a, b, c):
   print(a, b, c)

t = (2, 3, 4)
func(t) # error
func(*t) # ok, like func(t[0], t[1], t[2])

d1 = {'b': 2, 'a': 3, 'c': 4}
d2 = {'b': 2, 'a': 3, 'd': 4}

func(*d1) # ok
func(**d1) # ok
func(*d2) # ok
func(**d2) # not ok
```

Unpacking operators: *, **

```
x, y, z = 3, 4, 5 # ok

x, y, z = 3, 4, 5, 6
# ValueError: too many values to unpack

x, y, z = 3, 4
# ValueError: not enough values to unpack

x, y, *z = 3, 4, 5, 6 # ok, z = [5, 6]
x, y, *z = 3, 4 # ok, z = []

x, *y, *z = 3, 4, 5, 6, 7, 8
# SyntaxError: multiple starred expressions in
assignment

(x, *y), *z = (3, 4, 5), 6, 7, 8 # ok
```

Working with the filesystem

```
import os, shutil

os.getcwd() # gets the current dir
os.chdir('path/to/dir') # changes the directory
os.mkdir('dirname') # creates a dir
os.rmdir('dirname') # removes an empty dir
shutil.rmtree('dirname') # removes a tree
os.listdir() # list the content of the dir
os.path.exists('name') # True if the file exists
os.path.isdir('name') # True if 'name' is dir
os.path.isfile('name') # True if 'name' is file
shutil.copy('file1', 'some/path/file2') # copy file
os.remove('file') # remove the file
os.system('os command') # call os command
```

Working with the filesystem

```
import os

for root, dirs, files in os.walk('path/to/start'):
   print(root)   # the current dir
   print(dirs)   # the subdirs of current dir
   print(files)  # the files of current dir
```

Command line parameters

```
# test.py
import sys

# print all arguments (parameters)
for p in sys.argv:
    print(p)
```

```
# command line

> python test.py param1 param2

# Linux
> "full/path/to/python" "full/path/to/test.py" 'Dana Popescu'

#Windows
> "C:\Program Files\Python3.9\bin\python" "full/path/to/test.py" "Dana
Popescu"
```

Doctests

```
def factorial(n):
    """
    Calculates factorial of n

    Parameters:
        n (int): the input number

    Returns:
         int: the computed factorial

    Example usage:
    >>> factorial(5)
    120
    >>> factorial(7)
    50401
    """

    if n <= 1:
        return 1
    return n * factorial(n - 1)
```

Doctests

```
if __name__ == '__main__':
    from doctest import testmod
    testmod()
    testmod(verbose=True)
```