Python Basic Exercises 5 - Lists
(home work: will be solved next class)

- **Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple with the square of every number.**

- **Suppose the following input is supplied to the program: 34,67,55,33,12,98**

- **Then, the output should be:**
  [1156, 4489, 3025, 1089, 144, 9604]
  (1156, 4489, 3025, 1089, 144, 9604)

Hint: In case of input data being supplied to the question, it should be assumed to be a console input. tuple() constructor method can convert list to tuple

Python Basic Exercises 5 - Lists
(home work: will be solved next class)

- **Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple with the square of every number.**

SOLUTION:

```
numbers_str = input("Please enter a comma-separated sequence of
numbers:")
numbers_list = numbers_str.split(",")

numbers_squared_list = []
for item in numbers_list:
    numbers_squared_list.append(float(item)**2)

numbers_squared_tuple = tuple(numbers_squared_list)
print(numbers_squared_list)
print(numbers_squared_tuple)
```

Python Basic Exercises 6 - Dictionary
(home work: will be solved next class)

- **With a given integer number n, write a program to generate a dictionary that contains (i, i*i) such that is an integral number between 1 and n (both included). Finally, the program should print the dictionary.**

- **Suppose the following input is supplied to the program: 8**
  Then, the output should be:
  {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}

Hint: In case of input data being supplied to the question, it should be assumed to be a console input. Consider using the constructor method dict()

Python Basic Exercises 6 - Dictionary
(home work: will be solved next class)

- **With a given integer number n, write a program to generate a dictionary that contains (i, i*i) such that is an integral number between 1 and n (both included). Finally, the program should print the dictionary.**

SOLUTION:

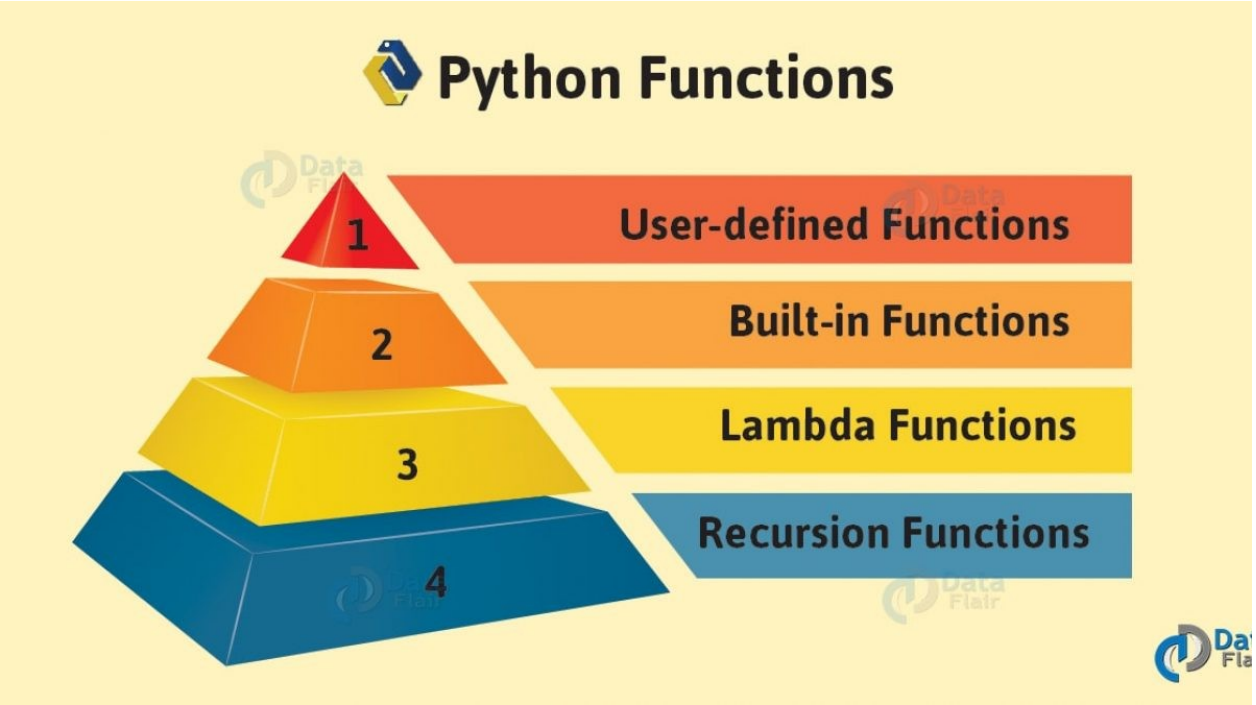
```
n = int(input("Please enter a positive integer number: "))

n_dict = {}
for i in range(n):
    n_dict[i] = i**2

print(n_dict)
```

Functions

a named section of a program
that performs a specific task




```
1   2              3    4
def function_name(args):
    #code in function
          5
```

```
(bottom)
1  def shake():
2      print('bugaloo')
3
4  shake()
5
```

Function & body

Function call

Debug I/O | Exceptions | Python Shell | Messages | OS Commands

Debug I/O (stdin, stdout, stderr) appears below

```
bugaloo
```

Function output

Line 5 Col 0 - [User]

```python
#### FUNCTIONS AND NOTHING - None is a type in python
#### FUNCIONES Y LA NADA - None es un tipo en python
def func1():
    """ this is my own doc string - it becomes available to anyone who calls help(func1). """
    pass

if func1()==None:
    print("It returned the 'nothing'! / Me ha devuelto la 'nada'!")
else:
    print("It returned something!")

help(func1)
```

```
#### FUNCTIONS AND NOTHING - None is a type in python
#### FUNCIONES Y LA NADA - None es un tipo en python
def func1():
          """ This is my own doc string - it becomes available to anyone who calls
help(func1)."""
    pass

if func1()==None:
    print("It returned the 'nothing'! / Me ha devuelto la 'nada'!")
else:
    print("It returned something!")

help(func1)
```

```python
#### FUNCTIONS - DEFAULT PARAMETERS & RECURSION- THE ZEN OF PYTHON.
#### FUNCIONES - PARAMETROS DEFAULT & RECURSIÓN - EL ZEN DE PYTHON.
def func2(y=True,z_lista=[], x=10, recursion_level=1):
    if (recursion_level<2):
        func2(x,y,z_lista.copy(),recursion_level+1)
    d_lista[2] = recursion_level
    return(x,y,z_lista)

a,b,c=func2()
print(a,b,c)

a,b,c=func2(True,d_lista.copy(),12,recursion_level=0)
print(a,b,c)
```


```python
#### FUNCTIONS - DEFAULT PARAMETERS & RECURSION- THE ZEN OF PYTHON.
#### FUNCIONES - PARAMETROS DEFAULT & RECURSIÓN - EL ZEN DE PYTHON.
def func2(y=True,z_lista=[], x=10, recursion_level=1):
    if (recursion_level<2):
        func2(x,y,z_lista.copy(),recursion_level+1)
    d_lista[2] = recursion_level
    return(x,y,z_lista)

a,b,c=func2()
print(a,b,c)

a,b,c=func2(True,d_lista.copy(),12,recursion_level=0)
print(a,b,c)
```

```
#### TUPLES & FUNCTIONS - PARAMETERS and RETURN VIA TUPLES
#### FUNCIONES - PARAMETROS y RETURN VIA TUPLAS.
d_list = [1,"a",True]

# On the right side you are packaging in a tuple.
# / En la parte derecha está empaquetando en una tupla.
# On the left side you are unpacking the tuple into variables.
# / En la parte izquierda está desempaquetando de la tupla en variables.
(d_list[2], d_list[1], d_list[0]) = (d_list[0], d_list[1], d_list[2])

d_tuple = (1,"a",True)
d_list[1] ="ready!"
print(d_list[1],d_tuple[1])

def func2(x,y,z):
    return(x,y,z)

a,b,c=func2(12,True,"Pepe")
print(a,b,c)
```


```
d_list = [1,"a",True]
(d_list[2], d_list[1], d_list[0]) = (d_list[0], d_list[1], d_list[2])

d_tuple = (1,"a",True)
d_list[1] ="ready!"
print(d_list[1],d_tuple[1])

def func2(x,y,z):
    return(x,y,z)

a,b,c=func2(12,True,"Pepe")
print(a,b,c)
```

```python
#### LAMBDA FUNCTIONS / FUNCIONES - LAMBDA.
def func1(x,y=10):
    a = 10
    return(x * 2+y+a)

print(func1(20))
lambda_func1 = lambda x,y=10,a=10: x * 2 + y +a
print(lambda_func1(20))
```


```
#### LAMBDA FUNCTIONS / FUNCIONES - LAMBDA.
def func1(x,y=10):
    a = 10
    return(x * 2+y+a)

print(func1(20))
lambda_func1 = lambda x,y=10,a=10: x * 2 + y +a
print(lambda_func1(20))
```

```python
#### LIST COMPREHENSION
#### LISTA COMPRENDIDA
collection = [0,1,2,3]
b = list() # b = []
for val in collection:
    if val % 2 == 0:
        b.append(val**2)
    else:
        b.append(val)

#[expr for val in collection if condition]

c_list_comprendida = [val**2 for val in collection if val % 2 == 0]
d_list_comprendida = [val**2  if val % 2 == 0 else val for val in collection]
#[expr for val in collection if condition]
```


```python
#### LIST COMPREHENSION
#### LISTA COMPRENDIDA
collection = [0,1,2,3]
b = list() # b = []
for val in collection:
    if val % 2 == 0:
        b.append(val**2)
    else:
        b.append(val)

#[expr for val in collection if condition]

c_list_comprendida = [val**2 for val in collection if val % 2 == 0]
d_list_comprendida = [val**2  if val % 2 == 0 else val for val in collection]
#[expr for val in collection if condition]
```

Python Basic Exercises 3 - function

Write a function which can compute the factorial of a given numbers. Suppose the following input is supplied to the program: 8
Then, the output should be: 40320

Hint: x=int(input("Enter the n to obtain the factorial(n):"))

Python Basic Exercises 4 - function

- **Write a program that invokes a function that calculates and returns the value according to the given formula:**

  Q = [(2 * C * D)/H]

Call your function several times, using integer values of your choice to C, D and H. The output of your function should always be integer.

Hint: If the output received is in decimal form, it should be rounded off to its nearest value (for example, if the output received is 26.1, it should be printed as 26)

Exercise for next Face to Face Class:

For next class, listen to the song, find the pattern, then write a function: soco.bate.vira(n)

That receives:

```
> soco.bate.vira(2)
[1] Soco Soco  Bate  Bate  Soco  Soco  Vira  Vira

> soco.bate.vira(1)
[1] Soco  Bate  Soco  Vira

> soco.bate.vira(0)
[1] Soco  Bate Vira
```

What is the result of soco.bate.vira(3)?

https://www.youtube.com/watch?v=VZGcm_JYOKM

Chocolate prize next F2F class for anyone that can do this version in class: https://www.youtube.com/watch?v=zk_3onMpwwU

SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

ADVANCED-OPTIONAL:

https://github.com/yhilpisch/py4fi/blob/master/jupyter36/04_Data_Structures.ipynb