Mastering Python

الدرس #4

Python’s Language Basics

اساسيات لغة بايثون

By:

Hussam Hourani

V1.0 - NOV 2019

Agenda

- Functions
- Introduction to *args
- Global/Local Variables
- Recursive Functions
- Lambda
- Map
- Filter
- Zipping
- Reduce

Functions

A function is a block of code which only runs when it is called.

```
def my_function():
  print("Hello from a function")

my_function()
my_function()
my_function()
```

Output

```
Hello from a function
Hello from a function
Hello from a function
```

```
def my_function(fname):
  print(fname + " is the file name")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Output

```
Emil is the file name
Tobias is the file name
Linus is the file name
```

```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

Output

```
I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

Functions

```
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

Output

```
apple
banana
cherry
```

```
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

Output

```
15
25
45
```

```
def my_function(child3, child2, child1):
  print("The youngest child is " +
child3)

my_function(child1 = "Sam", child2 =
"Tobias", child3 = "Khalid")
```

Output

```
The youngest child is Khalid
```

Introduction to *args

```
1 def adder(x,y,z):
2     print("sum:",x+y+z)
3
4 adder(10,12,13)
```

Output

```
In [13]:
sum: 35

In [14]:
```

```
1  def adder(*num):
2      sum = 0
3
4      for n in num:
5          sum = sum + n
6      print("Sum:",sum)
7
8  adder(3,5)
9  adder(4,5,6,7)
10 adder(1,2,3,5,6)
11 adder(1,2,3,5,6,7,8,9,10,11,12,13,14)
```

Output

```
In [15]:
Sum: 8
Sum: 22
Sum: 17
Sum: 101

In [16]:
```

In the function, we should use an asterisk * before the parameter name to pass variable length arguments.

https://www.programiz.com/python-programming/args-and-kwargs

*args

```python
def myFun(arg1, *argv):
    print ("First argument :", arg1)
    for arg in argv:
        print("Next argument through *argv :", arg)

myFun('Hello', 'Welcome', 'to', 'GeeksforGeeks')
```

Output

```
In [17]: runfile('C:/Python/Mystuff/args2.py
First argument : Hello
Next argument through *argv : Welcome
Next argument through *argv : to
Next argument through *argv : GeeksforGeeks
```

```python
def area(*args):
    return args[0]*args[1]/2

print (area(10,30))
```

Output

```
In [26]:
150.0

In [27]:
```

Note

def area(base, height):
return base*height/2

https://www.geeksforgeeks.org/args-kwargs-python/

https://www.pythonforthelab.com/blog/what-are-args-and-kwargs-and-when-to-use-them/

*args

```python
def some_args(arg_1, arg_2, arg_3):
    print("arg_1:", arg_1)
    print("arg_2:", arg_2)
    print("arg_3:", arg_3)

my_list = [2, 3]
some_args(1, *my_list)
```

Output

```
In [22]:
arg_1: 1
arg_2: 2
arg_3: 3

In [23]:
```

```python
def test_var_args_call(arg1, arg2, arg3):
    print ("arg1:", arg1)
    print ("arg2:", arg2)
    print ("arg3:", arg3)

args = ("two", 3)
test_var_args_call(1, *args)
```

Output

```
In [24]: r
arg1: 1
arg2: two
arg3: 3

In [25]:
```


https://www.digitalocean.com/community/tutorials/how-to-use-args-and-kwargs-in-python-3

https://www.saltycrane.com/blog/2008/01/how-to-use-args-and-kwargs-in-python/

Introduction to **kwargs

```
1 def myFun(**kwargs):
2     for key, value in kwargs.items():
3         print ("%s == %s" %(key, value))
4
5 myFun(first ='Geeks', mid ='for', last='Geeks')
```

Output

```
In [21]: runfi
first == Geeks
mid == for
last == Geeks
```

https://www.programiz.com/python-programming/args-and-kwargs
https://www.geeksforgeeks.org/args-kwargs-python/

Global/Local Variables

```
x = "awesome"

def myfunc():
  x = "fantastic"
  print("Python is " + x)

myfunc()

print("Python is " + x)
```

Output

```
Python is fantastic
Python is awesome
```

```
x = "awesome"

def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

Output

```
Python is fantastic
```

Recursive Functions

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

factorial(10)
```

Output

```
factorial(10)
Out[]: 3628800
```

```
factorial(100)
```

Output

```
933262154439441526816992388562667004
907159682643816214685929638952175999
932299156089414639761565182862536979
208272237582511852109168640000000000
00000000000000
```

Lambda

- Lambda : Inline anonymous function (Not bounded to a name)

lambda *arguments* : *expression*

```
sum = lambda x, y : x + y
print ( sum(56,7) )
```

Output → 63

```
x= (lambda x, y: x + y)(2, 3)
print(x)
```

Output → 5

```
print((lambda x, y: x + y)(2, 3))
```

Output → 5

```
print ((lambda x, y, z=3: x + y + z)(1, 2))
```

Output → 6

Map

- Map : Applies a function to all the items in a sequence.

```
my_pets = ['alfred', 'tabitha', 'william', 'arla']
uppered_pets = []
for pet in my_pets:
     pet_ = pet.upper()
     uppered_pets.append(pet_)
print(uppered_pets)
```

Output

```
['ALFRED', 'TABITHA', 'WILLIAM', 'ARLA']
```

```
my_pets = ['alfred', 'tabitha', 'william', 'arla']
uppered_pets = list(map(str.upper, my_pets))
print(uppered_pets)
```

Output

```
['ALFRED', 'TABITHA', 'WILLIAM', 'ARLA']
```

```
list(map(lambda x: x.upper(), ['cat', 'dog', 'cow']))
```

Output

```
['CAT', 'DOG', 'COW']
```

```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
final_list = list(map(lambda x: x*2 , li))
print(final_list)
```

Output

```
[10, 14, 44, 194, 108, 124, 154, 46,
146, 122]
```

```
MyList = [0,1,2,3,4,10,13,22,25,100,120]
print("squared List:", list(map(lambda x: x**2, MyList)) )
```

Output

```
squared List:[0, 1, 4, 9, 16, 100,
169, 484, 625, 10000, 14400]
```

Map

```
circle_areas = [3.56773, 5.57668, 4.00914, 56.24241,
9.01344, 32.00013]
result = list(map(round, circle_areas, range(1,7)))
print(result)
```

Output

```
[3.6, 5.58, 4.009, 56.2424, 9.01344,
32.00013]
```

```
sentence = 'AlZytonah University of Jordan '
print ( list(map(lambda word: len(word), sentence.split())
) )
```

Output

```
[9, 10, 2, 6]
```

Filter

- Filter : Filter out all the elements of a sequence (filters the given sequence with the help of a function that tests each element in the sequence to be true or not.).

```
filtered = list( filter(lambda x: x % 2 == 0, range(0,11)))
print(filtered)
```

Output

```
[0, 2, 4, 6, 8, 10]
```

```
MyList = [0,1,2,3,4,10,13,22,25,100,120]
odd_numbers = list(filter(lambda x: x % 2, MyList))
print(odd_numbers)

even_numbers = list(filter(lambda x: x % 2 == 0, MyList))
print(even_numbers)
```

Output

```
[1, 3, 13, 25]
[0, 2, 4, 10, 22, 100, 120]
```

```
scores = [66, 90, 68, 59, 76, 60, 88, 74, 81, 65]
def is_A_student(score):
    return score > 75
over_75 = list(filter(is_A_student, scores))
print(over_75)
```

Output

```
[90, 76, 88, 81]
```

Zipping

- Python’s zip() function creates an iterator that will aggregate elements from two or more iterables

```
my_strings = ['a', 'b', 'c', 'd', 'e']
my_numbers = [1,2,3,4,5]
results = list(map(lambda x, y: (x, y), my_strings, my_numbers))
print(results)
```

Output

```
[('a', 1), ('b', 2), ('c', 3), ('d', 4),
('e', 5)]
```

```
my_strings = ['a', 'b', 'c', 'd', 'e']
my_numbers = [1,2,3,4,5]
results = list(zip(my_strings, my_numbers))
print(results)
```

Output

```
[('a', 1), ('b', 2), ('c', 3), ('d', 4),
('e', 5)]
```

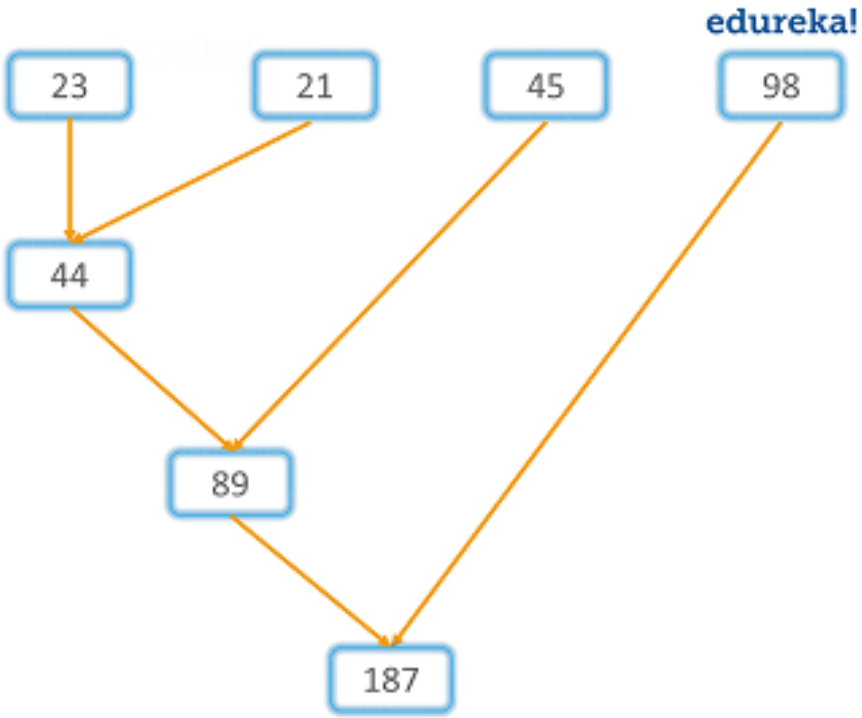
Reduce

- The reduce() function applies a given function to the iterables and returns a single value.

```
from functools import reduce
x= reduce(lambda a,b: a+b,[23,21,45,98])
print(x)
```

Output

```
187
```



Reduce

```
import functools

# initializing list
lis = [ 1 , 3, 5, 6, 2, ]

# using reduce to compute sum of list
print ("The sum of the list elements is : ",end="")
print (functools.reduce(lambda a,b : a+b,lis))

# using reduce to compute maximum element from list
print ("The maximum element of the list is : ",end="")
print (functools.reduce(lambda a,b : a if a > b else b,lis))
```

Output

```
The sum of the list elements is : 17
The maximum element of the list is : 6
```

Master in Software Engineering

Hussam Hourani has over 25 years of Organizations Transformation, VROs, PMO, Large Scale and Enterprise Programs Global Delivery, Leadership, Business Development and Management Consulting. His client experience is wide ranging across many sectors but focuses on Performance Enhancement, Transformation, Enterprise Program Management, Artificial Intelligence and Data Science.