01292021 - Functions

February 27, 2021

[1]:
```
print ('this is a print function')
print ([1,2])
print (('apple, cherry'))
```

```
this is a print function
[1, 2]
apple, cherry
```

[2]:
```
x = [1,2,3,4]
len(x)
```

[2]: 4

[3]:
```
print ()
```

[4]:
```
len() # gives an error saying, an argument must be passed
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-4-adf3103c7c3e> in <module>
----> 1 len()


TypeError: len() takes exactly one argument (0 given)
```

[5]:
```
print (x.sum()) # this functionality is not available in Basic Python
```

```
␣
↪---------------------------------------------------------------------------
```

```
AttributeError                            Traceback (most recent call␣
↪last)

<ipython-input-5-45d42b366c38> in <module>
----> 1 print (x.sum())


AttributeError: 'list' object has no attribute 'sum'
```

0.0.1 User defined function

[6]:
```
def welcome():
    print ("hi")
    print ("Python is insteresting")

# def - keyword to define the functions
# welcome - name of the user defined fuction

# syntax - def function_name() ==> to accept arguments
# there is no output
```

[7]:
```
welcome()
```

```
hi
Python is insteresting
```

0.0.2 User defined function with arguments

[8]:
```
def func_with_arg(var1):
    var1 = var1 * 2
    print ('funtion with argument : ', var1)
```

[9]:
```
func_with_arg(500)
```

```
funtion with argument :  1000
```

[10]:
```
func_with_arg('apple')
```

```
funtion with argument :  appleapple
```

[11]:
```
func_with_arg([1,2,3,4])
```

```
funtion with argument :  [1, 2, 3, 4, 1, 2, 3, 4]
```

0.0.3 User defined function with multiple arguments

[12]:
```
def function_mul_argu(arg1, arg2):
    print ('first arg : ', arg1)
    print ('sec arg : ', arg2)

    sumofarg = arg1 + arg2
    print ('sum of arg1 & arg2 : ', sumofarg)
```

[13]:
```
function_mul_argu(100,200)
```

```
first arg :  100
sec arg :  200
sum of arg1 & arg2 :  300
```

[14]:
```
function_mul_argu('apple', "banana")
```

```
first arg :  apple
sec arg :  banana
sum of arg1 & arg2 :  applebanana
```

[15]:
```
function_mul_argu([1,2,3,4], [100,200,300,400])
```

```
first arg :  [1, 2, 3, 4]
sec arg :  [100, 200, 300, 400]
sum of arg1 & arg2 :  [1, 2, 3, 4, 100, 200, 300, 400]
```

0.0.4 User defined function with default arguments

[16]:
```
def printinfo(name = "Adam", age = 35):
   print ("Name : ", name)
   print ("Age : ", age)
```

[17]:
```
printinfo()
```

```
Name :  Adam
Age :  35
```

[18]:
```
printinfo(name = 'Shweta', age = 20)
```

```
Name :  Shweta
Age :  20
```

[ ]:
```

```

[21]:
```
def function_mul_default_argu(arg1=10, arg2=20):
    sumofarg = arg1 + arg2
    print (sumofarg)
```

[22]:
```
function_mul_default_argu()
```

30

[25]:
```
outoffunc = function_mul_default_argu() # function is executed
print (outoffunc)

# Won't be able to save the output of a function to a variable
# print (x) is showing as nonetype
```

30
None

return - The return keyword is to exit a function and return a value.

[26]:
```
def function_mul_default_argu(arg1=10, arg2=20):
    sumofarg = arg1 + arg2
    return (sumofarg)
```

[27]:
```
function_mul_default_argu()
```

[27]: 30

[30]:
```
outoffunc = function_mul_default_argu() # function is executed
print (outoffunc)
```

30

[31]:
```
outoffunc + 100
```

[31]: 130

[32]:
```
function_mul_default_argu('apple', 'pineapple')
```

[32]: 'applepineapple'

[33]:
```
function_mul_default_argu(10, 'pineapple')
```

```
      ␣
↪---------------------------------------------------------------------------

    TypeError                                 Traceback (most recent call␣
↪last)

    <ipython-input-33-0d6d0f19e8dd> in <module>
----> 1 function_mul_default_argu(10, 'pineapple')
```

```
<ipython-input-26-0eeede576103> in function_mul_default_argu(arg1, arg2)
      1 def function_mul_default_argu(arg1=10, arg2=20):
----> 2     sumofarg = arg1 + arg2
      3     return (sumofarg)


TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

[34]:
```
function_mul_default_argu(100)
```

[34]: 120

[35]:
```
function_mul_default_argu(arg2 = 100)
```

[35]: 110

0.1 Examples

Write a Python function to sum all the numbers in a list.

Sample List : [8, 2, 3, 0, 7]

Expected Output : 20

[39]:
```
def sum(numbers):
    total = 0
    for x in numbers:
        total += x       # total = total + x
    return total
```

[40]:
```
sum([8,2,3,0,7])
```

[40]: 20

Write a Python function to multiply all the numbers in a list.

Sample List : [8, 2, 3, -1, 7]

Expected Output : -336

[41]:
```
def multiply(numbers):
    total = 1
    for x in numbers:
        total *= x       # total = total * x
    return total
```

[43]:
```
multiply([8,2,3,-1,7])
```

[43]: -336

Writing a function to check if the word "python" is in my string, return true else return false

python is an interesting language

True

[46]:
```
def search(str1):
    if str1.find('python')==-1:
        return False
    else:
        return True
search('python is an interesting language')
```

[46]: True

[47]:
```
def word_check(myword):
    if 'python' in myword:
        return True
    else:
        return False
word_check('python is an interesting language')
```

[47]: True

[48]:
```
word_check('Python is an interesting language')
```

[48]: False

[50]:
```
def word_check(myword):
    if 'python' in myword.lower():
        return True
    else:
        return False
print (word_check('python is an interesting language'))
print (word_check('Python is an interesting language'))
```

```
True
True
```

[51]:
```
word_check('Pythons are snakes')
```

[51]: True

[52]:
```
"Pythons" == 'Python'
```

[52]: False

[53]:
```
def check(st):
    if 'python' in st.split():
        return True
    return False
```

[55]:
```
check('pythons are snakes')
```

[55]: False

[56]:
```
"python" in "pythons"
```

[56]: True

1 Built-in Sequence Functions

1.0.1 enumerate

[57]:
```
store_list = ['MacDonalds', 'Taco Bell', 'Dunkin Donuts', 'Wendys', 'Chiptole']␣
↪#List of food stores
```

[58]:
```
# Print data element and index using enumerate method
for position, name in enumerate (store_list):
    print (position, name)
```

```
0 MacDonalds
1 Taco Bell
2 Dunkin Donuts
3 Wendys
4 Chiptole
```

[59]:
```
for a, b in enumerate (store_list):
    print (a, b)
```

```
0 MacDonalds
1 Taco Bell
2 Dunkin Donuts
3 Wendys
4 Chiptole
```

1.0.2 Sorted

Returns the new sorted list for the given sequence

[60]:
```
# Sort numbers

sorted([81,29,54,37,8,27,65]) # default - ascending order
```

[60]: [8, 27, 29, 37, 54, 65, 81]

[61]:
```
sorted([81,29,54,37,8,27,65], reverse = True)
```

[61]: [81, 65, 54, 37, 29, 27, 8]

1.0.3 Reversed

Iterates the data in reverse order

[62]:
```
# Create a list of numbers for range 15

num_list = range(15)
print (num_list)
list(num_list)
```

range(0, 15)

[62]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

[63]:
```
print(list(reversed(num_list)))
```

[14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

[70]:
```
reversed(num_list)
```

[70]: <range_iterator at 0x7f591a4400c0>

[64]:
```
list1 = [2009, 'mani', 123, 'xyz', 'zara', 'abc', 2009, 'mani']
print(list(reversed(list1)))
```

['mani', 2009, 'abc', 'zara', 'xyz', 123, 'mani', 2009]

1.0.4 Zip

Creates lists of tuples by pairing up elements of lists, tuples, or other sequence

[65]:
```
# Define list of subjects and count

subjects = ['maths', 'statistics', 'algebra']
count = ['one', 'two', 'three']
```

[67]:
```
# Zip function to pair the data elements of lists
total_subjects = zip(subjects,count)
print (tuple(total_subjects))
```

(('maths', 'one'), ('statistics', 'two'), ('algebra', 'three'))

[68]:
```
total_subjects
```

[68]: <zip at 0x7f591a654730>

[69]:
```
type (total_subjects)
```

[69]: zip

[71]:
```
print (list(total_subjects))
```

```
[]
```

[72]:
```
print (dict(total_subjects))
```

```
{}
```

[ ]:
```

```