# Python Programming (Basic-Intermediate)

## Module 3 - Functions

## Defining a function

```
dir()
```

```
['In',
 'Out',
 '_',
 '_1',
 '_2',
 '_4',
 '__',
 '___',
 '__builtin__',
 '__builtins__',
 '__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_dh',
 '_i',
 '_i1',
 '_i10',
 '_i11',
```

```
def greet_user():
    """Display a simple greeting."""
    print("Hello!")
```

```
dir()
```

```
['In',
 'Out',
 '_',
 '_1',
 '_11',
 '_2',
```

```
'_4',
'__',
'___',
'__builtin__',
'__builtins__',
'__doc__',
'__loader__',
'__name__',
'__package__',
'__spec__',
'_dh',
'_i',
'_i1',
'_i10'
```

```
greet_user()
```

```
Hello!
```

```
?greet_user
```

## Passing information to a function

```
def greet_user(username):
  """Display a simple greeting."""
  print(f"Hello, {username.title()}!")
```

```
greet_user('santitham')
```

```
Hello, Santitham!
```

```
# greet_user(123)
# greet_user("123") # Must be this
```

## Positional arguments

```
def difference(arg1, arg2):
  return arg1 - arg2
```

```
difference(5,3)
```

```
2
```

## Keyword arguments

```
def describe_pet(animal_type, pet_name):
  """Display information about a pet."""
  print(f"\nI have a {animal_type}.")
  print(f"My {animal_type}'s name is {pet_name.title()}.")
```

```
describe_pet(animal_type='cat', pet_name='tigris')
```

```
I have a cat.
My cat's name is Tigris.
```

```
describe_pet(pet_name='tigris', animal_type='cat')
```

```
I have a cat.
My cat's name is Tigris.
```

## Default values

```
def describe_pet(pet_name, animal_type='cat'):
  """Display information about a pet."""
  print(f"\nI have a {animal_type}.")
  print(f"My {animal_type}'s name is {pet_name.title()}.")
```

```
describe_pet('Cheesy')
```

```
I have a cat.
My cat's name is Cheesy.
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# should mount drive before run this
import pandas as pd
df=pd.read_excel('/content/drive/MyDrive/AIS_DG/Superstore.xlsx')
```

## Return - simple value

```
def get_formatted_name(first_name, last_name):
  """Return a full name, neatly formatted."""
  full_name = f"{first_name} {last_name}"
  return full_name.title()
```

```
ajyai = get_formatted_name('santitham','prom-on')
print(ajyai)
```

```
Santitham Prom-On
```

## Return - a dictionary

```
def build_person(first_name, last_name):
  """Return a dictionary of information about a person."""
  person = {'first':first_name, 'last':last_name}
  return person
```

```
build_person('santitham','prom-on')
```

```
{'first': 'santitham', 'last': 'prom-on'}
```

```
def multiple_return():
    return (1,2)
```

```
x,y = multiple_return()
x
```

```
1
```

```
len([1,2,3])
```

```
3
```

```
sum([1,2,3])
```

```
6
```

```
def function_choice(name):
    if name == 'sum':
        return sum
    elif name == 'len':
        return len
```

```
ret = function_choice('len')
ret([1,2,3])
```

```
3
```

## Multiple arguments

```
def sum_many_args(*args):
  print(type(args))
  return sum(args)
```

```
sum_many_args(1,2,3,4,5)
```

```
<class 'tuple'>
```

```
15
```

## Lambda functions

```
pow = lambda x,y : x**y
pow(2,3)
```

```
8
```

```
import pandas as pd
df=pd.read_excel('/content/drive/MyDrive/AIS_DG/Superstore.xlsx')
```

```
df
```

```
      Row ID        Order ID Order Date  Ship Date       Ship Mode  \
0          1  CA-2013-152156 2014-11-09 2014-11-12    Second Class
1          2  CA-2013-152156 2014-11-09 2014-11-12    Second Class
2          3  CA-2013-138688 2014-06-13 2014-06-17    Second Class
3          4  US-2012-108966 2013-10-11 2013-10-18  Standard Class
4          5  US-2012-108966 2013-10-11 2013-10-18  Standard Class
...      ...             ...        ...        ...             ...
9989    9990  CA-2011-110422 2012-01-22 2012-01-24    Second Class
9990    9991  CA-2014-121258 2015-02-27 2015-03-04  Standard Class
9991    9992  CA-2014-121258 2015-02-27 2015-03-04  Standard Class
9992    9993  CA-2014-121258 2015-02-27 2015-03-04  Standard Class
9993    9994  CA-2014-119914 2015-05-05 2015-05-10    Second Class

     Customer ID     Customer Name    Segment        Country
0       CG-12520       Claire Gute   Consumer  United States
1       CG-12520       Claire Gute   Consumer  United States
2       DV-13045   Darrin Van Huff  Corporate  United States
3       SO-20335    Sean O'Donnell   Consumer  United States
4       SO-20335    Sean O'Donnell   Consumer  United States
```

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2013-152156 | 2014-11-09 | 2014-11-12 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... |
| 1 | 2 | CA-2013-152156 | 2014-11-09 | 2014-11-12 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... |
| 2 | 3 | CA-2013-138688 | 2014-06-13 | 2014-06-17 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | ... |
| 3 | 4 | US-2012-108966 | 2013-10-11 | 2013-10-18 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... |
| 4 | 5 | US-2012-108966 | 2013-10-11 | 2013-10-18 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9989 | 9990 | CA-2011-110422 | 2012-01-22 | 2012-01-24 | Second Class | TB-21400 | Tom Boeckenhauer | Consumer | United States | Miami | ... |
| 9990 | 9991 | CA-2014-121258 | 2015-02-27 | 2015-03-04 | Standard Class | DB-13060 | Dave Brooks | Consumer | United States | Costa Mesa | ... |
| 9991 | 9992 | CA-2014-121258 | 2015-02-27 | 2015-03-04 | Standard Class | DB-13060 | Dave Brooks | Consumer | United States | Costa Mesa | ... |
| 9992 | 9993 | CA-2014-121258 | 2015-02-27 | 2015-03-04 | Standard Class | DB-13060 | Dave Brooks | Consumer | United States | Costa Mesa | ... |

```
df[['Sales','Profit','Quantity']].apply(lambda x: x/x.max())
```

```
         Sales    Profit  Quantity
0     0.011571  0.004990  0.142857
1     0.032332  0.026141  0.214286
2     0.000646  0.000818  0.142857
3     0.042299 -0.045599  0.357143
4     0.000988  0.000300  0.142857
```

```
...        ...       ...       ...
9989  0.001115  0.000488  0.214286
9990  0.004062  0.001861  0.142857
9991  0.011422  0.002309  0.142857
9992  0.001308  0.001586  0.285714
9993  0.010741  0.008684  0.142857

[9994 rows x 3 columns]
```

| | Sales | Profit | Quantity |
|---|---|---|---|
| 0 | 0.011571 | 0.004990 | 0.142857 |
| 1 | 0.032332 | 0.026141 | 0.214286 |
| 2 | 0.000646 | 0.000818 | 0.142857 |
| 3 | 0.042299 | -0.045599 | 0.357143 |
| 4 | 0.000988 | 0.000300 | 0.142857 |
| ... | ... | ... | ... |
| 9989 | 0.001115 | 0.000488 | 0.214286 |
| 9990 | 0.004062 | 0.001861 | 0.142857 |
| 9991 | 0.011422 | 0.002309 | 0.142857 |
| 9992 | 0.001308 | 0.001586 | 0.285714 |
| 9993 | 0.010741 | 0.008684 | 0.142857 |

9994 rows × 3 columns

## Variable scope - global

```
x = "global"

def foo():
  print("x inside:", x)

foo()
print("x outside:", x)
```

```
x inside: global
x outside: global
```

## Error in attempting to update global

```
x = "global"

def foo():
  x = 'x * 2'
  print(x)

foo()
print(x)
```

```
x * 2
global
```

## Variable scope - local

```
def sum(x,y):
  s = x + y
  return s

print(sum(5,10))
```

```
15
```

```
def foo():
  y1 = "local"

foo()
print(y1)
```

```
name 'y1' is not defined
```

## Variable scope - nonlocal

```
def outer():
  x = 'local'

  def inner():
    nonlocal x
    x = 'nonlocal'
    print('inner: ', x)

  inner()
  print('outer: ', x)

outer()
```

```
inner:  nonlocal
outer:  nonlocal
```

## Import your own module

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import sys
sys.path.append('/content/drive/MyDrive/AIS_DG/lib')
```

```
import mymodule1 as mm
```

```
mm.STATIC_VALUE
```

```
10
```

```
import mymodule1 as mm
mm.build_person('Santitham','Prom-on')
```

```
{'first': 'Santitham', 'last': 'Prom-on'}
```

```
from mymodule1 import STATIC_VALUE
from mymodule1 import build_person, build_person_with_title
```

```
dir()
```

```
['In',
 'Out',
 'STATIC_VALUE',
 '_',
 '_1',
 '_11',
 '_13',
 '_2',
 '_20',
 '_31',
 '_33',
 '_34',
 '_35',
 '_37',
 '_39',
 '_4',
 '_40',
 '_42',
 '_43',
 '_51',
```

```
build_person_with_title('Dr.','Santitham','Prom-on')
```

```
{'title': 'Dr.', 'first': 'Santitham', 'last': 'Prom-on'}
```

```
STATIC_VALUE
```

```
10
```

```
from mymodule1 import *

print(build_person('Santitham','Prom-on'))
print(build_person_with_title('Dr.','Santitham','Prom-on'))
```

```
{'first': 'Santitham', 'last': 'Prom-on'}
{'title': 'Dr.', 'first': 'Santitham', 'last': 'Prom-on'}
```

```
from numpy import *
```

```
dir()
```

```
['ALLOW_THREADS',
 'AxisError',
 'BUFSIZE',
 'CLIP',
 'ComplexWarning',
 'DataSource',
 'ERR_CALL',
 'ERR_DEFAULT',
 'ERR_IGNORE',
 'ERR_LOG',
 'ERR_PRINT',
 'ERR_RAISE',
 'ERR_WARN',
 'FLOATING_POINT_SUPPORT',
 'FPE_DIVIDEBYZERO',
 'FPE_INVALID',
 'FPE_OVERFLOW',
 'FPE_UNDERFLOW',
 'False_',
 'In',
```

```
import mymodule
```

```
from importlib import reload
reload(mymodule)
```

```
<module 'mymodule' from '/content/drive/MyDrive/AIS_DG/lib/mymodule.py
```

```
!pip show pandas
```

```
Name: pandas
Version: 1.5.3
Summary: Powerful data structures for data analysis, time series, and
Home-page: https://pandas.pydata.org
Author: The Pandas Development Team
Author-email: pandas-dev@python.org
License: BSD-3-Clause
Location: /usr/local/lib/python3.10/dist-packages
Requires: numpy, python-dateutil, pytz
Required-by: altair, arviz, bigframes, bokeh, bqplot, cmdstanpy, cuffl
```

## Activity

Write a function in a file 'myutils.py' that perform:

- Receive list as argument
- Find maximum value/location
- Return value, location as a tuple

Import and test it.

```
x = [1,2,3,10,0,3,4]
```

```
max_x = max(x)
[(k,v) for k,v in enumerate(x) if v == max_x]
```

```
[(3, 10)]
```

```
def find_max(x):
    """
    Find index and value of maximum values in the list

    Parameter
    ------------
    x: list
       Input argument to be used

    Return
    ------------
    List of tuples that have maximum values

    Example
    ------------
    >>> x = [1,2,3,10,0,3,4]
    >>> find_max(x)
        [(3, 10)]
    """
    max_x = max(x)
    return [(k,v) for k,v in enumerate(x) if v == max_x]
```

```
find_max(x)
```

```
[(3, 10)]
```

```
%%writefile /content/drive/MyDrive/AIS_DG/lib/oamUtil.py
def find_max(x):
    """
    Find index and value of maximum values in the list

    Parameter
    ------------
    x: list
       Input argument to be used

    Return
    ------------
    List of tuples that have maximum values

    Example
    ------------
    >>> x = [1,2,3,10,0,3,4]
    >>> find_max(x)
        [(3, 10)]
    """
    max_x = max(x)
    return [(k,v) for k,v in enumerate(x) if v == max_x]
```

```
Writing /content/drive/MyDrive/AIS_DG/lib/oamUtil.py
```

```
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
```

```
Mounted at /content/drive
```

```
import sys
sys.path.append('/content/drive/MyDrive/AIS_DG/lib')
```

```
# work here
import oamUtil
```

```
x = [1,2,3,10,0,3,4]
oamUtil.find_max(x)
```

```
[(3, 10)]
```