Lab 1: Basic Python Programming

CPE232 Data Models

---

[1] Variable

1.1 Number Variable

```
num = 100 #integer variable
num2 = 12.5 #float variable
print(num)
print(num2)

print(num + num2)   #addition
print(num - num2)   #subtraction
print(num * num2)   #multiplication
print( num / num2)  #division
```

1.2 String Variable

```
#string variable
string = "Data Models"
print(string) #print complete string

print("Hello " + string)    #print concatenated string
print(string[0])            #print first character of the string
print(string[:4])           #print first to 4th character of the string
print(string[5:])           #print 6th to last character of the string
print(string[1:4])          #print 2nd to 4th character of the string
print(string * 2)           #print string 2 time
```

1.3 Boolean Variable

```
#boolean variable
boolean = True
boolean2 = False

print(boolean)              #print boolean variable
print(not boolean)          #print opposite of boolean variable
print(boolean and boolean2) #print boolean and boolean2
print(boolean or boolean2)  #print boolean or boolean2
```

1.4 List Variable

```
#list variable
list = ["Data",20,123.23,40,50]
another_list = ["Models",60]

print(list)                 #print complete list
print(list[0])              #print first element of the list
print(list[1:3])            #print 2nd to 3rd element of the list
print(list[2:])             #print 3rd to last element of the list
print(another_list)         #print complete another_list
print(another_list * 2)     #print another_list two times
print(list + another_list)  #print concatenated list

list[0] = "CPE232"          #change first element of the list
print(list)                 #print complete list
```

1.5 Tuple Variable

```
#tuple variable
tuple = ("Data",20,123.23,40,50)
```

```
another_tuple = ("Models",60)

print(tuple)                    #print complete tuple
print(tuple[0])                 #print first element of the tuple
print(tuple[1:3])               #print 2nd to 3rd element of the tuple
print(tuple[2:])                #print 3rd to last element of the tuple
print(tuple * 2)                #print tuple two times
print(tuple + another_tuple)    #print concatenated tuple
```

```
tuple[0] = "CPE232"          #trying to change first element of the tuple but it cannot be changed so it gives error
```

1.6 Dictionary Variable

```
#dictionary variable
dictionary = {"name":"Alice","age":21}
another_dictionary = {}
another_dictionary["name"] = "Bob"
another_dictionary["age"] = 21

print(dictionary)                  #print complete dictionary
print(dictionary["name"])          #print value for specific key
print(dictionary.keys())           #print all the keys
print(dictionary.values())         #print all the values
print(dictionary.items())          #print all the items
print(another_dictionary)          #print complete another_dictionary
```

[2] Control Flow

2.1 IF ... ELIF ... ELSE

```
number = 123
number2 = 34

if number > number2:
    print("number is greater thanu number2")
elif number < number2:
    print("number is less than number2")
else:
    print("number is equal to number2")
```

[3] Loop

3.1 For Loop

```
#for loops
for num in range(0,10):
    print(num)
```

```
#for loop with list

list = ["Alice","Bob","Charlie","Daisy"]

for name in list:
    print(name)
```

```
#continue in for loop

list = [1,23,7,"hello",True,1123,43,23,12]

for element in list:
    if type(element) != int:
        continue
    print(element)
```

```
#break in for loop

list = [1,23,7,"hello",True,1123,43,23,12]
```

```
for element in list:
    if type(element) != int:
        break
    print(element)
```

3.2 While loop

```
#while loop

list = ["Alice","Bob","Charlie","Daisy"]
count = 0

while count < len(list):
    print(list[count])
    count += 1
```

```
#continue in while loop

list = [1,23,7,"hello",True,1123,43,23,12]
count = 0

while count < len(list):
    if type(list[count]) != int:
        count += 1
        continue
    print(list[count])
    count += 1
```

```
#break in while loop

list = [1,23,7,"hello",True,1123,43,23,12]
count = 0

while count < len(list):
    if type(list[count]) != int:
        break
    print(list[count])
    count += 1
```

[4] Function

```
#define function
def function_name (arg1, arg2):
    return arg1 + arg2

#calling function
function_name(1,2)
```

```
#define function with default argument
def function_with_default_arg(arg1, arg2 = 10, arg3 = 20 , arg4 = 30):
    return arg1 + arg2 + arg3 + arg4

result_1 = function_with_default_arg(1)
result_2 = function_with_default_arg(1,2,5)
result_3 = function_with_default_arg(1,2,5,10)

print(result_1)
print(result_2)
print(result_3)
```

```
#multiple agument
def function_with_multiple_arg(*args):
    print(args)
    print(type(args))
    sum = 0
    for num in args:
        sum += num

    return sum

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

```
#lambda function
lambda_function = lambda arg1, arg2: arg1 + arg2

print(lambda_function(1,2))
```

[5] File Handling

5.1 Text File

```
with open("test.txt","w") as file:
    file.write("Hello World")
```

```
with open("test.txt","r") as file:
    print(file.read())
```

5.2 CSV File

```
import csv

with open("test.csv","w",newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Name","Surname"])
        writer.writerow(["Alice","Johnson"])
        writer.writerow(["Bob","Smith"])
```

```
import csv

with open("test.csv","r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

[4] Libraries

4.1 Numpy

import numpy library

```
import numpy as np
```

ndarray initialization

Construct using python list

```
# 1d ndarray from 1d python list
list_a1=[1,2,3.5]
arr_a1=np.array(list_a1)
arr_a1
```

```
# 2d ndarray from 2d python list (list of list)
list_a2=[[1,2],[3,4],[5,6]]
arr_a2=np.array(list_a2)
arr_a2
```

```
list_a3=[[[1,2],[2,3]],[[3,4],[4,5]]]
arr_a3=np.array(list_a3)
arr_a3
```

or construct using some numpy classes and functions

```
np.zeros(5)
np.ones((3,4),dtype=float)
```

```
np.full((4,),999)
```

```
np.arange(3,10,2)
```

```
np.linspace(10,15,11)
```

```
np.random.choice(['a','b'],9)
```

```
np.random.randn(10)
```

ndarray properties

```
list_a=[[1,2,3,4],[5,6,7,8],[9,10,11,12]]
arr_a=np.array(list_a)
arr_a
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
arr_a.ndim
```

```
arr_a.shape
```

```
(3, 4)
```

```
arr_a.dtype
```

```
arr_a.size
```

Reshaping & Modification

from this original ndarray

```
arr_a
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

try to convert into 3D array

```
arr_a.reshape((2,2,3))
```

```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]]])
```

sometimes you may resize for same dimension where only known some dimension, insert -1 for unknown len

```
arr_a.reshape((-1,6))
```

```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12]])
```

Would you like to try this?

```
arr_a.reshape((-1,5))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-286d5aa6424c> in <cell line: 0>()
----> 1 arr_a.reshape((-1,5))

ValueError: cannot reshape array of size 12 into shape (5)
```

[Q1] From the above cell, explain in your own words why it worked or did not work.

Ans: It not work because 5 can not multiply to 12 in integer number

Next, try to append any value(s) into exist 2darray

```
np.append(arr_a,13)
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13])
```

```
np.append(arr_a,arr_a[0])
```

```
np.append(arr_a,arr_a[0].reshape((1,-1)),axis=0)
```

```
np.append(arr_a,arr_a[:,0].reshape((-1,1)),axis=1)
```

```
np.concatenate([arr_a,arr_a])
```

```
np.concatenate([arr_a,arr_a],axis=1)
```

indexing & slicing

from this original array again

```
arr_a
```

try to access all element at the first row

```
arr_a[1]
```

then you would like to access the second element from the first row

```
arr_a[1][2]
```

```
arr_a[1,2]
```

Next, try to access all element start from 1th in the first row

```
arr_a[1,1:]
```

```
arr_a[:2,1:]
```

sometimes you may specify some row number using list within indicing

```
arr_a[[1,2,1],1:]
```

Boolean slicing

based on this original array

```
arr_a
```

try to filter all elements which more than 5

```
arr_a>5
```

```
array([[False, False, False, False],
       [False,  True,  True,  True],
       [ True,  True,  True,  True]])
```

Next, try to filter all elements which more than 5 and less than 10

```
(arr_a>5)&(arr_a<10)
```

```
array([[False, False, False, False],
       [False,  True,  True,  True],
       [ True, False, False, False]])
```

Run the cell below and answer a question.

```
arr_a[(arr_a>5)&(arr_a<10)]
```

```
array([6, 7, 8, 9])
```

[Q2] From the above cell, explain in your own words how the output came about?

Ans: we print the filterd values in the arr_a those are greater than 5 and less than 10

Try running the cell below.

```
arr_a[(arr_a>5) and (arr_a<10)]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-78eb1746bbfd> in <cell line: 0>()
----> 1 arr_a[(arr_a>5) and (arr_a<10)]

ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()
```

[Q3] Explain in your own words why the above cell gives an error.

Ans: Because "and" can only use in the simple boolean for this problem can not use this is boolean array not simple boolean

[Q4] And what should be written instead so that the code is error-free?

Ans: replace "and" into "&"

Basic operations

```
list_b=[[1,2,3,4],[1,2,3,4],[1,2,3,4]]
arr_b=np.array(list_b)
arr_b
```

This is some operations for only 1 array

```
np.sqrt(arr_b)
```

This is some operations for 2 arrays with the same shape

```
arr_a-arr_b
```

```
np.add(arr_a,arr_b)
```

Next, try to operate with 1 array and one numeric variable

```
arr_a*3
```

```
1+arr_a**2
```

Try to play with 2 arrays with different shape

```
arr_c=np.array([1,2,3])
arr_d=np.array([[3],[5],[8]])
```

```
arr_c-arr_d
```

Basic aggregations

```
arr_a
```

```
arr_a.sum()
```

```
arr_a.mean()
```

```
arr_a.min()
```

```
arr_a.max()
```

```
arr_a.std()
```

ndarray axis

```
arr_a
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
arr_a.sum(axis=0)
```

```
array([15, 18, 21, 24])
```

```
arr_a.sum(axis=1)
```

```
array([10, 26, 42])
```

[Q5] Summarize the value of the argument *axis*, what is the value for row-wise summation and column-wise summation, respectively?

Ans: 0: Column-wise operation and 1: Row-wise operation

4.2 Pandas

Series

```
import pandas as pd
import numpy as np
```

```
pd.Series(np.random.randn(6))
```

```
pd.Series(np.random.randn(6), index=['a','b','c','d','e','f'])
```

Constructing Dataframe

Constructing DataFrame from a dictionary

```
d = {'col1':[1,2], 'col2': [3,4]}
```

```
df = pd.DataFrame(data=d)
df
```

	col1	col2
0	1	3
1	2	4

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
d2 = {'Name':['Joe','Nat','Harry','Sam','Monica'],
      'Age': [20,21,19,20,22]}
```

```
df2 = pd.DataFrame(data=d2)
df2
```

	Name	Age
0	Joe	20
1	Nat	21
2	Harry	19
3	Sam	20
4	Monica	22

Next steps: Generate code with df2 | View recommended plots | New interactive sheet

Constructing DataFrame from a List

```
marks_list = [85.10, 77.80, 91.54, 88.78, 60.55]
```

```
df3 = pd.DataFrame(marks_list, columns=['Marks'])
df3
```

Creating DataFrame from file

```
# Read csv file from path and store to df for create dataframe
df = pd.read_csv('nss15.csv')
```

```
df
```

Viewing DataFrame information

(.shape, .head, .tail, .info, select column, .unique, .describe, select low with .loc and .iloc)

Check simple information

```
# Check dimension by .shape
df.shape
```

```
(334839, 12)
```

```
# Display the first 5 rows by default
df.head()
```

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
0	150733174	7/11/2015	15.7762	V	5	Male	NaN	57	33	1	9	1267
1	150734723	7/6/2015	83.2157	S	36	Male	White	57	34	1	1	1439
2	150817487	8/2/2015	74.8813	L	20	Female	NaN	71	94	1	0	3274
3	150717776	6/26/2015	15.7762	V	61	Male	NaN	71	35	1	0	611
4	150721694	7/4/2015	74.8813	L	88	Female	Other	62	75	1	0	1893

```
# Display the first 3 rows
df.head(3)
```

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
0	150733174	7/11/2015	15.7762	V	5	Male	NaN	57	33	1	9	1267
1	150734723	7/6/2015	83.2157	S	36	Male	White	57	34	1	1	1439
2	150817487	8/2/2015	74.8813	L	20	Female	NaN	71	94	1	0	3274

```
# Display the last 5 rows by default
df.tail()
```

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
334834	150739278	5/31/2015	15.0591	V	7	Male	NaN	59	76	1	1	1864
334835	150733393	7/11/2015	5.6748	C	3	Female	Black	68	85	1	0	1931
334836	150819286	7/24/2015	15.7762	V	38	Male	NaN	71	79	1	0	3250
334837	150823002	8/8/2015	97.9239	M	38	Female	White	59	82	1	1	464
334838	150723074	6/20/2015	49.2646	M	5	Female	White	57	34	1	9	3273

```
# Overview information of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 334839 entries, 0 to 334838
Data columns (total 12 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   caseNumber     334839 non-null  int64  
 1   treatmentDate  334839 non-null  object 
 2   statWeight     334839 non-null  float64
 3   stratum        334839 non-null  object 
 4   age            334839 non-null  int64  
 5   sex            334837 non-null  object 
 6   race           205014 non-null  object 
 7   diagnosis      334839 non-null  int64  
 8   bodyPart       334839 non-null  int64  
 9   disposition    334839 non-null  int64  
 10  location       334839 non-null  int64  
 11  product        334839 non-null  int64  
dtypes: float64(1), int64(7), object(4)
memory usage: 30.7+ MB
```

Select column, multiple column, with condition

```
df.columns
```

```
#select single column
df['age']
```

	age
0	5
1	36
2	20
3	61
4	88
...	...
334834	7
334835	3
334836	38
334837	38
334838	5

334839 rows × 1 columns

dtype: int64

```
df.age
```

	age
0	5
1	36
2	20
3	61
4	88
...	...
334834	7
334835	3
334836	38
334837	38
334838	5

334839 rows × 1 columns

dtype: int64

```
#select multiple column
df[['treatmentDate','statWeight','age','sex']]
```

	treatmentDate	statWeight	age	sex
0	7/11/2015	15.7762	5	Male
1	7/6/2015	83.2157	36	Male
2	8/2/2015	74.8813	20	Female
3	6/26/2015	15.7762	61	Male
4	7/4/2015	74.8813	88	Female
...	...	...	...	...
334834	5/31/2015	15.0591	7	Male
334835	7/11/2015	5.6748	3	Female
334836	7/24/2015	15.7762	38	Male
334837	8/8/2015	97.9239	38	Female
334838	6/20/2015	49.2646	5	Female

334839 rows × 4 columns

Viewing the unique value

```
df.race.unique()
```

Describe

```
df['age'].describe()
```

Select row with condition

```
#select by condition
df[df['sex'] == 'Male']
```

```
#select by multiple condition
df[(df['sex'] == 'Male') & (df['age'] > 80)]
```

Select row with .iloc

```
# select row by .iloc
df.iloc[10:15]
```

```
# select column by .iloc
df.iloc[:,[0,1,2,3,4]]
```

Select column and row with .loc

```
# select column and low by .loc
df.loc[:6,'treatmentDate':'diagnosis']
```

	treatmentDate	statWeight	stratum	age	sex	race	diagnosis
0	7/11/2015	15.7762	V	5	Male	NaN	57
1	7/6/2015	83.2157	S	36	Male	White	57
2	8/2/2015	74.8813	L	20	Female	NaN	71
3	6/26/2015	15.7762	V	61	Male	NaN	71
4	7/4/2015	74.8813	L	88	Female	Other	62
5	7/2/2015	5.6748	C	1	Female	White	71
6	6/8/2015	15.7762	V	25	Male	Black	51

```
# select row by condition
df.loc[df['age']>80, ['treatmentDate', 'age']]
```

	treatmentDate	age
4	7/4/2015	88
8	7/16/2015	98
39	5/3/2015	88
46	4/15/2015	91
63	1/12/2015	97
...	...	...
334701	4/27/2015	86
334784	7/7/2015	82
334785	7/11/2015	86
334815	10/28/2015	85
334819	1/13/2015	85

20422 rows × 2 columns

[Q6] What is the difference between .iloc and .loc?

Ans: iloc is the integer based indexing loc is the user's label based indexing