Name: Rahul Kulkarni
SRN: PES2UG22EC105

ASSIGNMENT – 1

Basics

Week-1

1. What is Data Science?
   - It is the science of analyzing the raw data and extracting insights from it through statistics, machine learning, AI or any other methods.

2. What are the steps in Data Science after getting the data?
   - Read Data
   - Data processing and cleaning
   - Summarizing data
   - Visualization
   - Deriving insights from data

3. Why Python for Data Science?
   - Python libraries provide key features essential for data science
   - Data Manipulation and Pre-processing
     - Python's pandas library offers a variety of functions for data wrangling and manipulation
   - Data Summary
   - Visualization
     - plotting libraries like matplotlib and seaborn aid in condensing statistical information and helps in identifying trends and relationships
   - Machine learning libraries using sci-kit learn offer a bouquet of machine learning algorithms

4. What is a variable?
   - an identifier containing a known informtion
   - variable name used to point to a memory location and used to reference the stored value

5. Commenting – #, ctrl+1
   Clearing Console – %clear, ctrl+L
   Deleting single variable: del b
   For all variables: del a,b,c,….etc or %reset

6. Basic libraries in Python – NumPy, Pandas, Matplotlib, Sklearn
   To see modules within a library –
   - import numpy
   - content = dir(numpy)
   - print(content)

VARIABLES AND DATA TYPES:

1. Declaring and assigning variables
   - a = 1
   - a,b,c = 1,2,3
2. Datatypes
   - Boolian
   - Integer
   - Floating
   - Complex
   - String

Identifying object data type – type(variable_name)

3. Coercing a datatype
   - change the datatype by keyword of the datatype -
   - a = 5
   - b = float(a)
   - type(a) = float

However conversion of string to other data type is only possible when the content of that string variable is numeric.

OPERATORS:

1. Different types of operators
   - Arithematic (*, /, %, **, //)
   - Assignment (=, += , -=, *=, /=)
   - Relational/Comparator (<, <=, >, >=, ==, !=)
   - Logic (or, and, not) - operands are conditions
   - Bitwise (|, &, ^) - operands are integers

2. Precedence of Operators
   - ()
   - **
   - /
   - *
   - +,-
   - &
   - |
   - Relationals (==, !=, >, >=, <, <=)
   - not
   - and
   - or

CODING QUESTIONS:

1) Let a = 5 (101 in binary) and b = 3 (011 in binary). What is the result of the following operation?

```
a = 5
b = 3
print(a&b)
```

a) 1
b) 0
c) Compilation error
d) Execution error

2) What will be the output of the following code snippet?

```
a = 15
b = 3
c = 4
result = a + b * c // (c % b) - 5
print(result)
```

a) Compilation Error
b) 22
c) Floating point negative number
d) 6

Week2
SEQUENCE DATA TYPES:

Sequence allows the user to store multiple values in organised and efficient fashion.
Different types of sequence:

1) Strings	– Sequence of characters ' or ''
2) Unicode Strings	
3) Lists	– Sequence of multi-data type objects
4) Tuples	– Sequence of unicode data
5) Arrays	– Sequence of constrained lists of objects
6) Range objects	– Used for looping

Dictionaries and sets are containers for non-sequential data.

1.
```
strsample = "hello"
print(strsample)
```
2.
```
lstnumber = [1,2,2,3,4,5,5]
print(lstnumber)          or          lstnumber[:]
```
3.
```
from array import *
arr = array('i', [1,2,3,4,5])
for x in arr: print(x) # prints the numbers
print(arr) # prints same as what is in arr
```

4. tup = (1, 2, 3, 'hi') # cannot be changed once assigned
tup2 = 1, 2, 3, 'hello' # can be used without paranthesis

5. Dictionaries:
# Unordered collection of data values like a map, 1,2,3,4 – key and first,second,.. are the key value pairs
dictsample = {1: 'first', 2: 'second, 3: 3, 4: 'four'}
or dictsample2 = dict([('first', 1), ('second', 2), (3, 3), ('four', 4)])

6. set = {'hello', 1, 2, 'going merry', 21.5} # list or another setcannot be an element of a set
set('hello')
> {'e', 'h', 'l', 'o'}

7. rangesample = range(1, 12, 4)
for x in rangesample: print(x)
```
> 1
  5
  9
```

OPERATIONS ON SEQUENCE DATATYPES:

1. String indexing: accessing elements
```
str = "learning"
str.index('l')      > 0
str.index('ning')   > 4
str.index(1)        > e
str.index(-2)       > n
str.index(-9)       > Index error: string index out of range
```

2. List Indexing
```
lst = [1, 2, 'hi'  ,4, 'hello']
lst.index('hi')     > 2
lst.index(3)        > 4
```

3. Array indexing
```
from array import *
arr = array('i', [1, 2, 3, 4])
arr[-3]             > 2
```

4. Tuple indexing
```
tupl = (1, 2, 3, 4, 'flower')
tupl.index('flower')  > 4
```

5. Set doesn't support indexing (because non- sequential data)

6. Dictionary Indexing: Using keys
```
dict = {1: 'first', 'second': 2, 3:3, 4: 'four'}
dict[2]             > key errordict[1]          > 'first'
```

7. Range indexing

```
rng = range(1, 12, 4)
rng.index(0)          > error - 0 not in range
rng.index(9)          > 2 (the position of 9 is 2)
rng[1]                > 5
```

SEQUENCE DATA OPERATIONS:

1. Slicing: To slice a given sequence

```
str = “learning'
str[slice(1,4,2)]     > 'er'  # starts from 1 (e) till 3(r) due to stepsize 2
str[:]                > prints 'learning'

lst = [1, 2, 3, 'hi', 4]
print(lst[2:])        > [3, 'hi, 4]
print(lst[:2])        > [1, 2]
```

Dictionary and sets cannot be sliced

```
arr = array('i', [1, 2, 3, 4])
for x in arr: print(x) > 1 2 3 4
arr[1:]                > array('i', [2, 3, 4])
arr[1: -1]            >  array('i', [2, 3])

rang = range(1, 12, 4)
print(rang[:-1])      > range(1, 9, 4)
print(rang[1:-1])     > range(5, 9, 4)
print(rang[2:-1: 2])          > range(5, 9, 8)
```

2. Concatenation: Joing the two datas - + += ,

```
str = “learning”
print(str + ' ', 'python')    > learning python
new1 = str + ' ' , “python”
print(new)                    > ('learning', 'python')

print(lst)            > [1, 2, 3, 'hi',  4]
lst+['py']            > [1, 2, 3, 'hi', 4, 'py']

arr = array('i', [1,2,3,4])
print(arr+[5])                > error - can only append array (not list) to array
print(arr+array('i', [5])     > array('i', [1,2,3,4,5])

tupl = (1,2,3,'flower')
tupl+=('hi')
print(tupl)           > (1,2,3,'flower','hi')

setsample = {1, 2, 3, 'word', 5}
setsample = setsampe, 7
print(setsample)    > ( {1, 2, 3, 'word', 5}, 7 )
```