Assignment: Python Practice 1

Name: Rama Krishna Kamma

CWID: 50321021

Date: 03-03-2023

Instructor: Dr. Pooja Rani

Basic/ Primitive Data types in Python:

Every value in Python has a datatype.

1. Numbers
2. Strings
3. Boolean

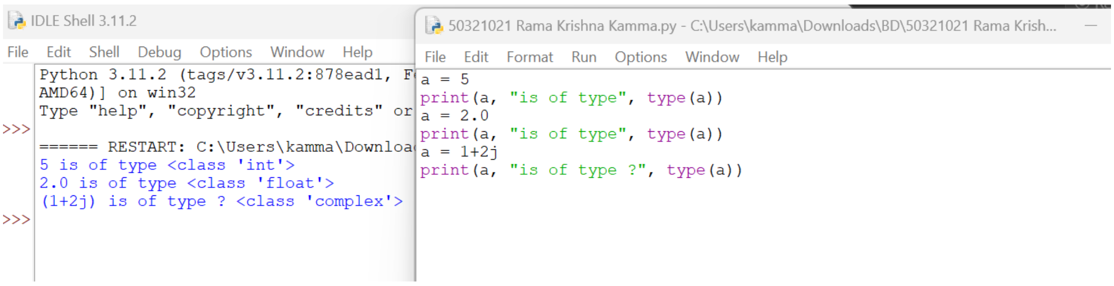
Examples:

- Integers, floating point numbers and complex numbers falls under Python numbers category. They are defined as int, float and complex class in Python.
- Integers can be of any length; it is only limited by the memory available. A floating-point number is accurate up to 15 decimal places.

Code:

```
a = 5
print(a, "is of type", type(a))
a = 2.0
print(a, "is of type", type(a))
a = 1+2j
print(a, "is of type ?", type(a))
```

Output:



IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

```
Python 3.11.2 (tags/v3.11.2:878ead1, F
AMD64)] on win32
Type "help", "copyright", "credits" or
>>>
====== RESTART: C:\Users\kamma\Downloa
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is of type ? <class 'complex'>
>>>
```

50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Krish...
File Edit Format Run Options Window Help

```
a = 5
print(a, "is of type", type(a))
a = 2.0
print(a, "is of type", type(a))
a = 1+2j
print(a, "is of type ?", type(a))
```

Strings:

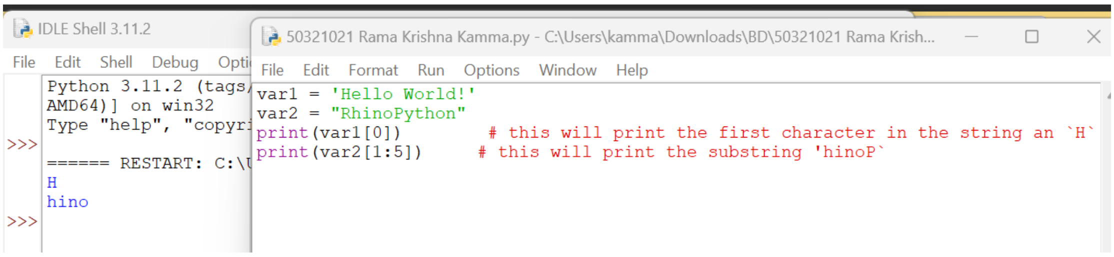
- String is sequence of Unicode characters.
- We can use single quotes or double quotes to represent strings.
- Strings can be accessed as a whole string, or a substring of the complete variable using brackets '[]'.

Code:

```
var1 = 'Hello World!'
```

```
var2 = "RhinoPython"
print(var1[0])      # this will print the first character in the string an `H`
print(var2[1:5])    # this will print the substring 'hinoP`
```

Output:


IDLE Shell 3.11.2
File Edit Shell Debug Opti
50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Krish...
File Edit Format Run Options Window Help

```
Python 3.11.2 (tags/
AMD64)] on win32
Type "help", "copyri
>>>
====== RESTART: C:\U
H
hino
>>>
```

```
var1 = 'Hello World!'
var2 = "RhinoPython"
print(var1[0])          # this will print the first character in the string an `H`
print(var2[1:5])      # this will print the substring 'hinoP`
```

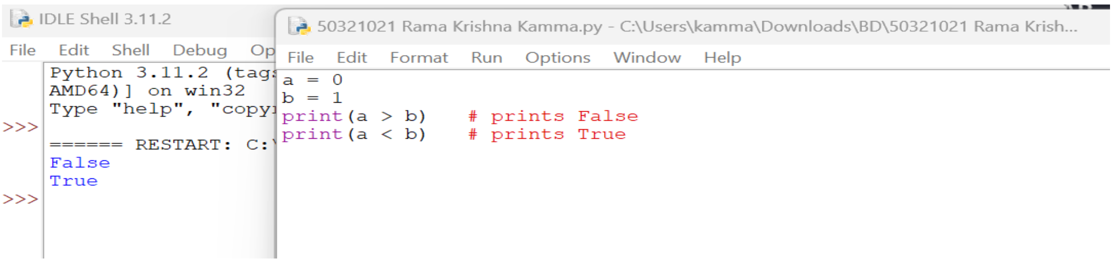
Boolean:

- Python 3 provides a Boolean data type.
- Objects of Boolean type may have one of two values, True or False

Code:

```
a = 0
b = 1
print(a > b)   # prints False
print(a < b)   # prints True
```

Output:


IDLE Shell 3.11.2
File Edit Shell Debug Op
50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Krish...
File Edit Format Run Options Window Help

```
Python 3.11.2 (tags
AMD64)] on win32
Type "help", "copyi
>>>
====== RESTART: C:\
False
True
>>>
```

```
a = 0
b = 1
print(a > b)    # prints False
print(a < b)    # prints True
```

Observation:

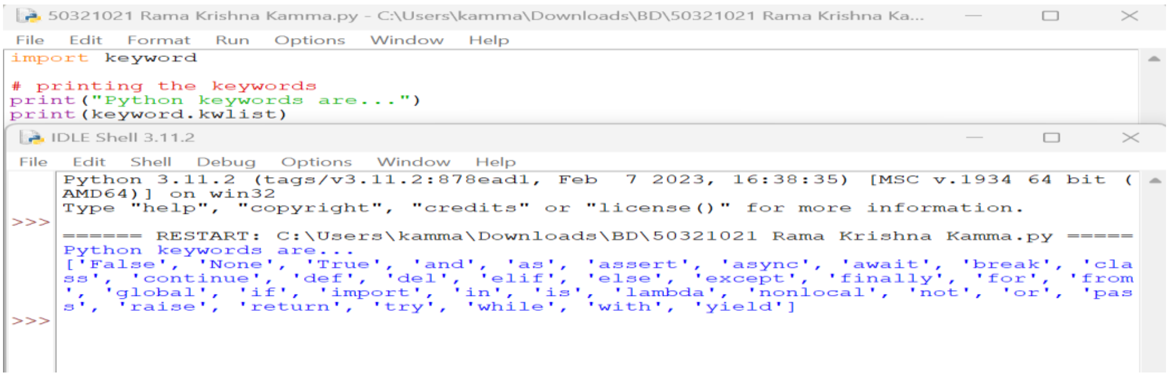
From the above, perform various operations on python Datatypes like Numbers, Strings, Boolean data types. How to declare datatypes and initialize the variable and perform various operations. The default datatype in python is string.

Keywords in Python:

Code:

```
import keyword
# printing the keywords
print("Python keywords are...")
print(keyword.kwlist)
```

Output:


50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Ka...
File Edit Format Run Options Window Help

```
import keyword

# printing the keywords
print("Python keywords are...")
print(keyword.kwlist)
```

IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

```
    Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (
    AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>>
    ====== RESTART: C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Kamma.py =====
    Python keywords are...
    ['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'cla
    ss', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from
    ', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pas
    s', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>>
```

Observation:

Imported the Keyword module and list down all keywords in python3.

Operators:

Operators are used to perform operations on variables and values.

Python provides the following operators:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

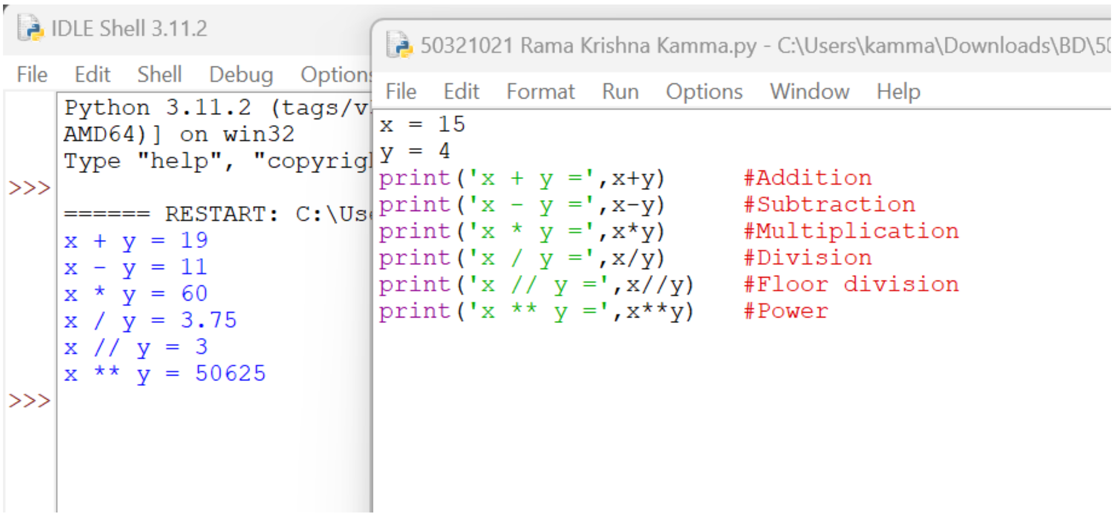
Arithmetic Operators:

Arithmetic operators are used with numeric values to perform common mathematical operations

Code:

```
x = 15
y = 4
print('x + y =',x+y)    #Addition
print('x - y =',x-y)    #Subtraction
print('x * y =',x*y)    #Multiplication
print('x / y =',x/y)    #Division
print('x // y =',x//y)  #Floor division
print('x ** y =',x**y)  #Power
```

Output:



IDLE Shell 3.11.2
File Edit Shell Debug Option
50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\5
File Edit Format Run Options Window Help

```
Python 3.11.2 (tags/v
AMD64)] on win32
Type "help", "copyrig
>>>
====== RESTART: C:\Us
x + y = 19
x - y = 11
x * y = 60
x / y = 3.75
x // y = 3
x ** y = 50625
>>>
```

```
x = 15
y = 4
print('x + y =',x+y)      #Addition
print('x - y =',x-y)      #Subtraction
print('x * y =',x*y)      #Multiplication
print('x / y =',x/y)      #Division
print('x // y =',x//y)    #Floor division
print('x ** y =',x**y)    #Power
```

Observation:

Performed the various arithmetic operations like addition, subtraction, multiplication, division, power etc.

Assignment Operators:

Assignment operators are used to assign values to variables.

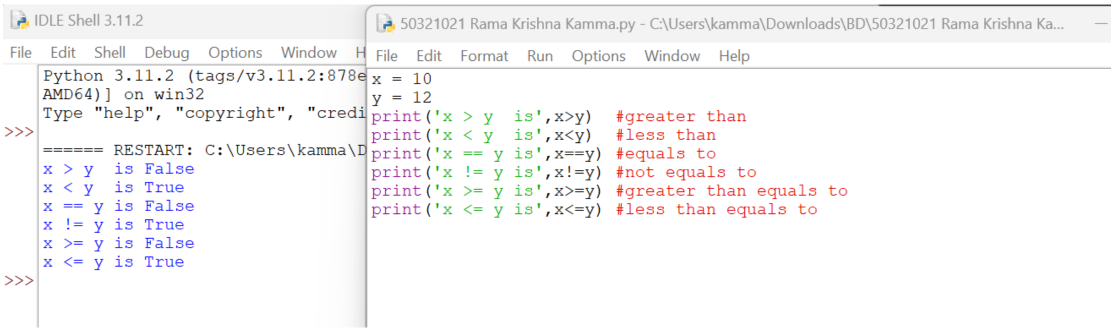
Comparison Operators:

Comparison operators are used to compare two values: It either returns True or False according to the condition.

Code:

```
x = 10
y = 12
print('x > y  is',x>y)  #greater than
print('x < y  is',x<y)  #less than
print('x == y is',x==y) #equals to
print('x != y is',x!=y) #not equals to
print('x >= y is',x>=y) #greater than equals to
print('x <= y is',x<=y) #less than equals to
```

Output:


IDLE Shell 3.11.2
File Edit Shell Debug Options Window H
50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Ka...
File Edit Format Run Options Window Help

```
Python 3.11.2 (tags/v3.11.2:878e
AMD64)] on win32
Type "help", "copyright", "credi
>>>
====== RESTART: C:\Users\kamma\D
x > y  is False
x < y  is True
x == y is False
x != y is True
x >= y is False
x <= y is True
>>>
```

```
x = 10
y = 12
print('x > y  is',x>y)  #greater than
print('x < y  is',x<y)  #less than
print('x == y is',x==y) #equals to
print('x != y is',x!=y) #not equals to
print('x >= y is',x>=y) #greater than equals to
print('x <= y is',x<=y) #less than equals to
```

Observation:

Performed various comparison operation like greater than, less than, equals, not equals to, greater than equals to, less than equals to etc.

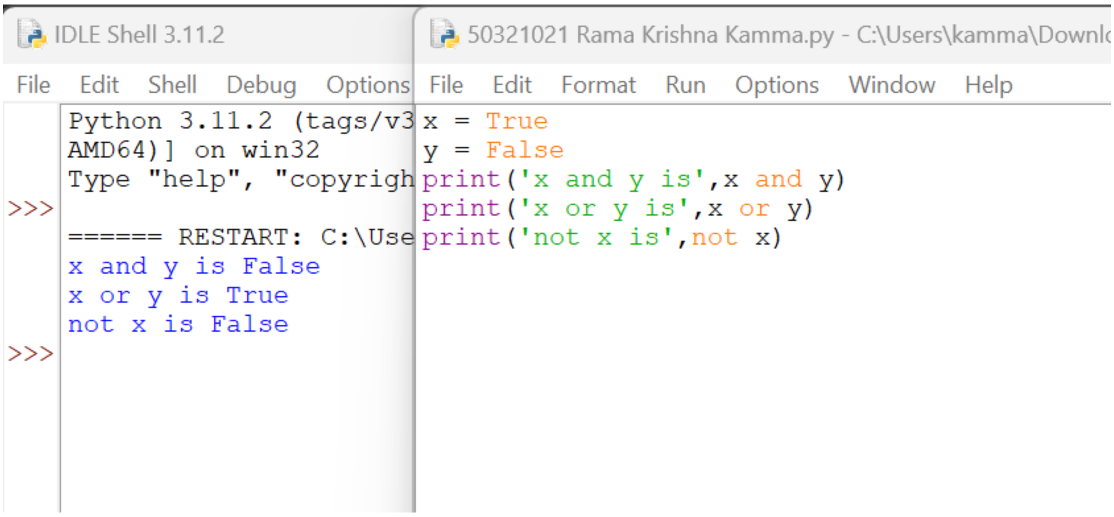
Logical Operators:

Logical operators are used to combine conditional statements.

Code:

```
x = True
y = False
print('x and y is',x and y)
print('x or y is',x or y)
print('not x is',not x)
```

Output:


IDLE Shell 3.11.2
File Edit Shell Debug Options
50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downl
File Edit Format Run Options Window Help

```
Python 3.11.2 (tags/v3
AMD64)] on win32
Type "help", "copyrigh
>>>
====== RESTART: C:\Use
x and y is False
x or y is True
not x is False
>>>
```

```
x = True
y = False
print('x and y is',x and y)
print('x or y is',x or y)
print('not x is',not x)
```

Observation:

Performed the various logical operations like and, or, not etc.

Membership Operator:

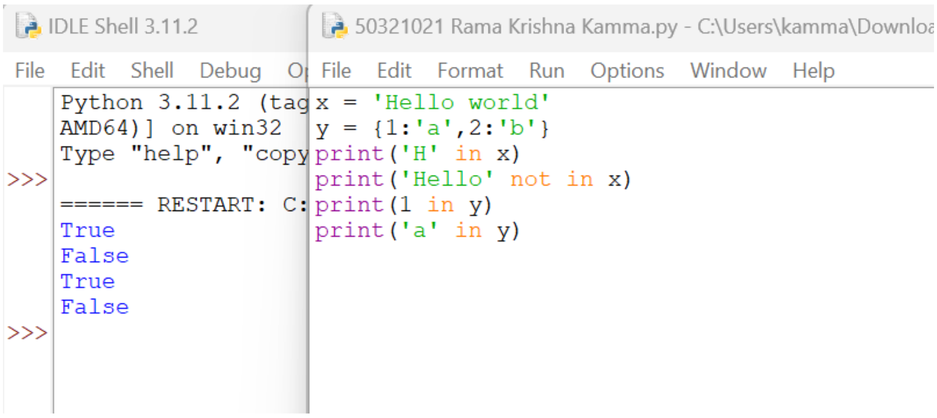
- in and not in are the membership operators in Python.

- They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

Code:

```
x = 'Hello world'
y = {1:'a',2:'b'}
print('H' in x)
print('Hello' not in x)
print(1 in y)
print('a' in y)
```

Output:



IDLE Shell 3.11.2
File Edit Shell Debug O
50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloa
File Edit Format Run Options Window Help

```
Python 3.11.2 (tag
AMD64)] on win32
Type "help", "copy
>>>
====== RESTART: C:
True
False
True
False
>>>
```

```
x = 'Hello world'
y = {1:'a',2:'b'}
print('H' in x)
print('Hello' not in x)
print(1 in y)
print('a' in y)
```

Observation:

Checks if element present in dictionary or not by using the membership operator.

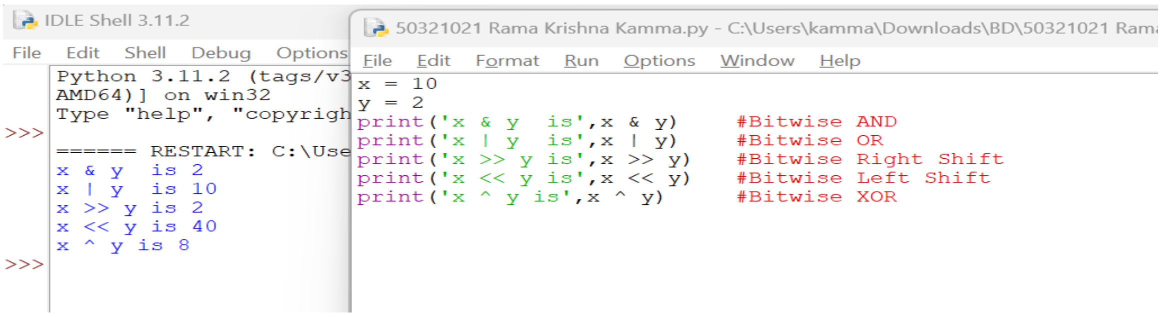
Bitwise Operators:

- Bitwise operators act on operands as if they were string of binary digits.
- It operates bit by bit, hence the name.
- For example, 2 is 10 in binary and 7 is 111.

Code:

```
x = 10
y = 2
print('x & y  is',x & y)    #Bitwise AND
print('x | y  is',x | y)    #Bitwise OR
print('x >> y is',x >> y)   #Bitwise Right Shift
print('x << y is',x << y)   #Bitwise Left Shift
print('x ^ y is',x ^ y)     #Bitwise XOR
```

Output:


IDLE Shell 3.11.2
File Edit Shell Debug Options
50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama
File Edit Format Run Options Window Help

```
Python 3.11.2 (tags/v3
AMD64)] on win32
Type "help", "copyrigh
>>>
====== RESTART: C:\Use
x & y  is 2
x | y  is 10
x >> y is 2
x << y is 40
x ^ y is 8
>>>
```

```
x = 10
y = 2
print('x & y  is',x & y)     #Bitwise AND
print('x | y  is',x | y)     #Bitwise OR
print('x >> y is',x >> y)    #Bitwise Right Shift
print('x << y is',x << y)    #Bitwise Left Shift
print('x ^ y is',x ^ y)      #Bitwise XOR
```

Observation:

Performed the various bitwise operators like and, or, right shift, left shift, xor. Here all the operations are performed in binary numbers instead of decimal values.

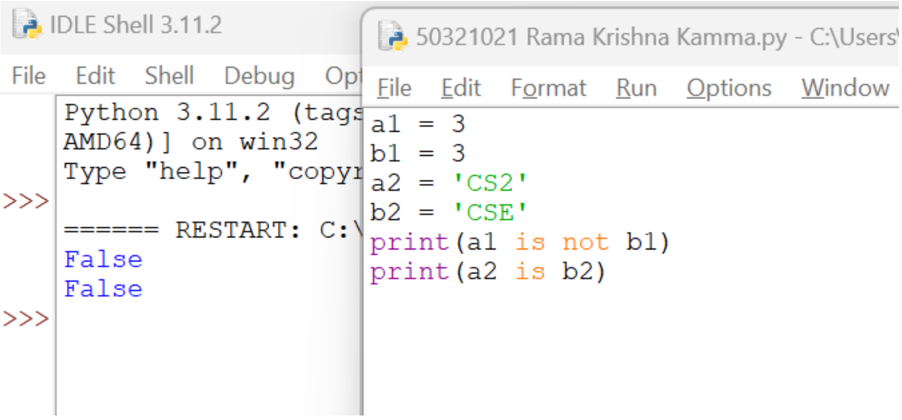
Identity Operators:

- is and is not are the identity operators in Python.
- They are used to check if two values (or variables) are located on the same part of the memory.

Code:

```
a1 = 3
b1 = 3
a2 = 'CS2'
b2 = 'CSE'
print(a1 is not b1)
print(a2 is b2)
```

Output:


IDLE Shell 3.11.2
File Edit Shell Debug Opt
50321021 Rama Krishna Kamma.py - C:\Users
File Edit Format Run Options Window

```
Python 3.11.2 (tags
AMD64)] on win32
Type "help", "copyr
>>>
====== RESTART: C:\
False
False
>>>
```

```
a1 = 3
b1 = 3
a2 = 'CS2'
b2 = 'CSE'
print(a1 is not b1)
print(a2 is b2)
```

Observation:

Performed the identity operators is and is not. The key difference between membership and identity operators is here we check it points to same memory location or not whereas in the case of membership check if the element is existing or not.

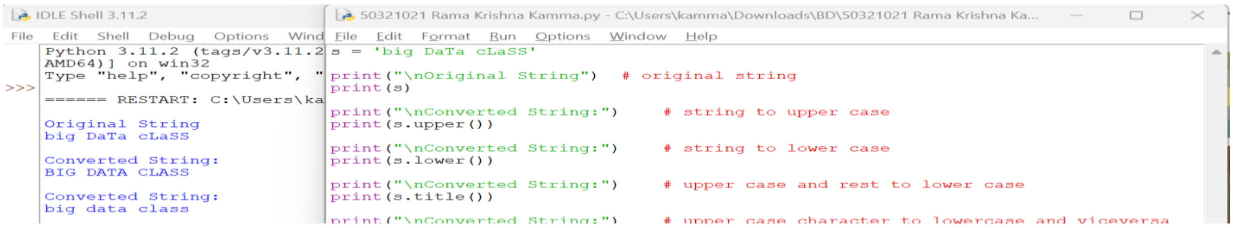
String Methods:

There are different string methods are available in python to perform various tasks. Some of them are lower (), upper(), title(), swapcase(), center(), count(), format(), isdecimal(), isalpha().

Code:

```
s = 'big DaTa cLaSS'
print("\nOriginal String")  # original string
print(s)
print("\nConverted String:")    # string to upper case
print(s.upper())
print("\nConverted String:")    # string to lower case
print(s.lower())
print("\nConverted String:")    # upper case and rest to lower case
print(s.title())
print("\nConverted String:")    # upper case character to lowercase and viceversa
print(s.swapcase())
```

Output:


IDLE Shell 3.11.2
File Edit Shell Debug Options Wind
50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Ka...
File Edit Format Run Options Window Help

```
Python 3.11.2 (tags/v3.11.2
AMD64)] on win32
Type "help", "copyright", "
>>>
====== RESTART: C:\Users\ka

Original String
big DaTa cLaSS

Converted String:
BIG DATA CLASS

Converted String:
big data class
```

```
s = 'big DaTa cLaSS'

print("\nOriginal String")   # original string
print(s)

print("\nConverted String:")     # string to upper case
print(s.upper())

print("\nConverted String:")     # string to lower case
print(s.lower())

print("\nConverted String:")     # upper case and rest to lower case
print(s.title())

print("\nConverted String:")     # upper case character to lowercase and viceversa
```

Observation:

Here taken some sample string and performed various string operations like upper(), lower(), isalpha(),isdecimal(),swapcase() etc.

Non-Primitive Datatypes:

- List
- Tuple
- Set
- Dictionary

List:

- List is an ordered collection of items(elements)
- Lists in Python are used to store collection of heterogeneous items.
- These are mutable, which means that you can change their content without changing their identity. It Allows duplicate members.
- list indices start at 0.

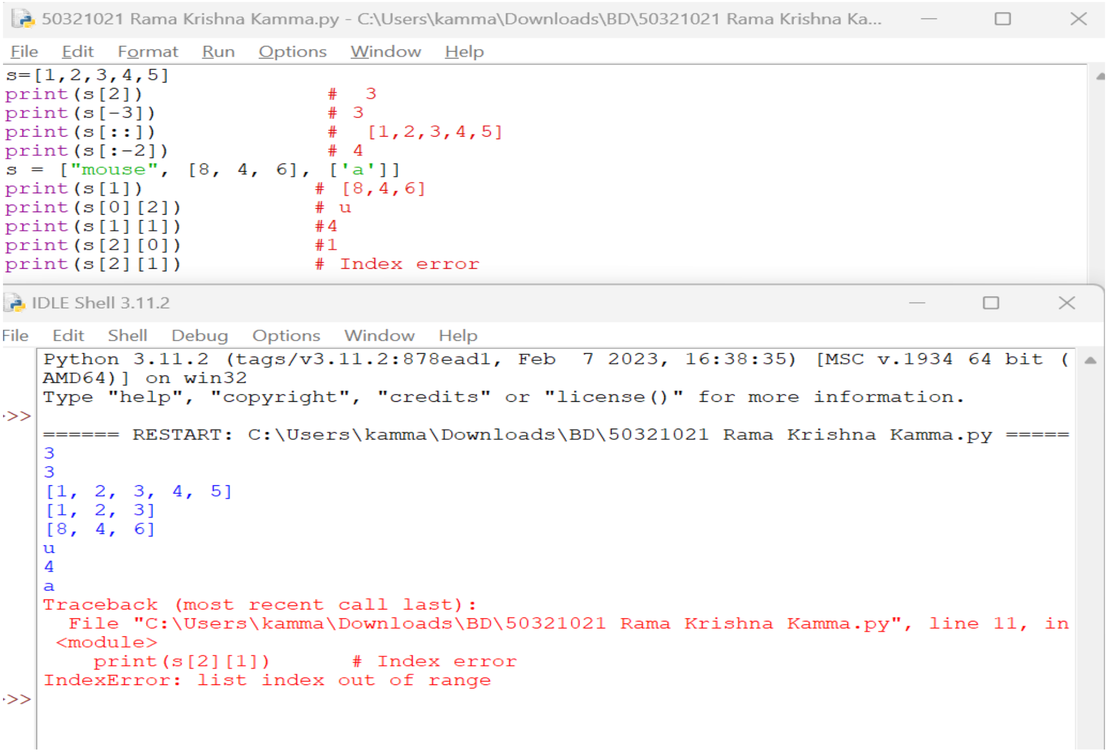
Negative indexing

- Python allows negative indexing for its sequences.
- The index of -1 refers to the last item, -2 to the second last item and so on.

Code:

```
s=[1,2,3,4,5]
print(s[2])                 #  3
print(s[-3])                # 3
print(s[::])                #  [1,2,3,4,5]
print(s[:-2])               # 4
s = ["mouse", [8, 4, 6], ['a']]
print(s[1])                # [8,4,6]
print(s[0][2])          # u
print(s[1][1])      #4
print(s[2][0])      #1
print(s[2][1])      # Index error
```

Operations:


50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Ka...
File Edit Format Run Options Window Help
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

```
s=[1,2,3,4,5]
print(s[2])                 #  3
print(s[-3])                # 3
print(s[::])                #  [1,2,3,4,5]
print(s[:-2])               # 4
s = ["mouse", [8, 4, 6], ['a']]
print(s[1])                # [8,4,6]
print(s[0][2])             # u
print(s[1][1])             #4
print(s[2][0])             #1
print(s[2][1])             # Index error
```

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Kamma.py =====
3
3
[1, 2, 3, 4, 5]
[1, 2, 3]
[8, 4, 6]
u
4
a
Traceback (most recent call last):
  File "C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Kamma.py", line 11, in
 <module>
    print(s[2][1])      # Index error
IndexError: list index out of range
>>>
```

Observation:

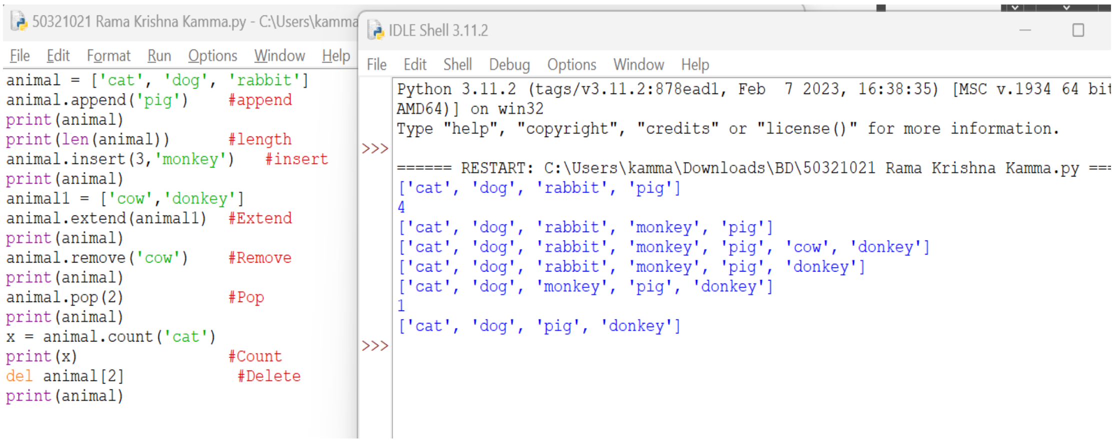
Taken the sample list performed indexing, slicing, and negative indexing.

Methods:

Code:

```
animal = ['cat', 'dog', 'rabbit']
animal.append('pig')    #append
print(animal)
print(len(animal))      #length
animal.insert(3,'monkey')   #insert
print(animal)
animal1 = ['cow','donkey']
animal.extend(animal1)  #Extend
print(animal)
animal.remove('cow')    #Remove
print(animal)
animal.pop(2)           #Pop
print(animal)
x = animal.count('cat')
print(x)                #Count
del animal[2]           #Delete
print(animal)
```

Output:


50321021 Rama Krishna Kamma.py - C:\Users\kamma
File Edit Format Run Options Window Help
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

```
animal = ['cat', 'dog', 'rabbit']
animal.append('pig')     #append
print(animal)
print(len(animal))       #length
animal.insert(3,'monkey')    #insert
print(animal)
animal1 = ['cow','donkey']
animal.extend(animal1)   #Extend
print(animal)
animal.remove('cow')     #Remove
print(animal)
animal.pop(2)            #Pop
print(animal)
x = animal.count('cat')
print(x)                 #Count
del animal[2]             #Delete
print(animal)
```

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Kamma.py ===
['cat', 'dog', 'rabbit', 'pig']
4
['cat', 'dog', 'rabbit', 'monkey', 'pig']
['cat', 'dog', 'rabbit', 'monkey', 'pig', 'cow', 'donkey']
['cat', 'dog', 'rabbit', 'monkey', 'pig', 'donkey']
['cat', 'dog', 'monkey', 'pig', 'donkey']
1
['cat', 'dog', 'pig', 'donkey']
>>>
```

Observation:

Performed various list operations like append, insert, extend, remove, pop, count methods on sample list.

Tuple:

- A Tuple is a collection of Python objects(elements) separated by commas.
- These are immutable, which means that you cannot change their content.

Negative Indexing:

Python allows negative indexing for its sequences.

The index of -1 refers to the last item, -2 to the second last item and so on.

Slicing:

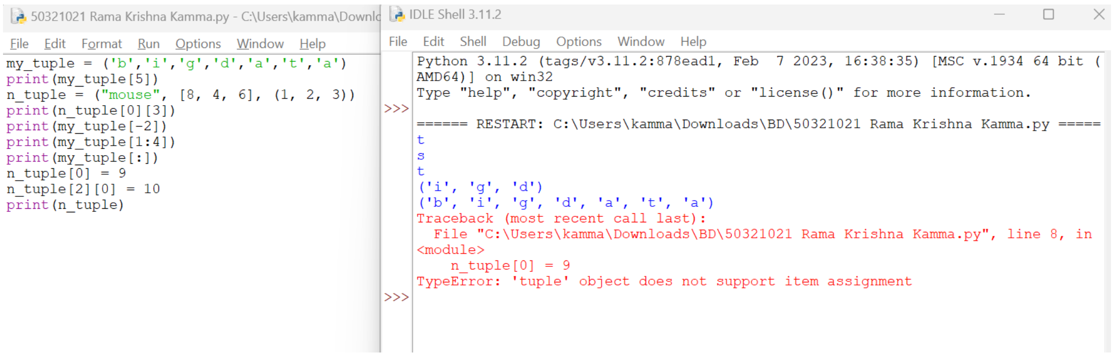
We can access a range of items in a tuple by using the slicing operator - colon ":".

Operations on Tuple:

Code:

```
my_tuple = ('b','i','g','d','a','t','a')
print(my_tuple[5])
n_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(n_tuple[0][3])
print(my_tuple[-2])
print(my_tuple[1:4])
print(my_tuple[:])
n_tuple[0] = 9
n_tuple[2][0] = 10
print(n_tuple)
```

Output:


50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downlo
File Edit Format Run Options Window Help
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

```
my_tuple = ('b','i','g','d','a','t','a')
print(my_tuple[5])
n_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(n_tuple[0][3])
print(my_tuple[-2])
print(my_tuple[1:4])
print(my_tuple[:])
n_tuple[0] = 9
n_tuple[2][0] = 10
print(n_tuple)
```

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Kamma.py =====
t
s
t
('i', 'g', 'd')
('b', 'i', 'g', 'd', 'a', 't', 'a')
Traceback (most recent call last):
  File "C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Kamma.py", line 8, in
<module>
    n_tuple[0] = 9
TypeError: 'tuple' object does not support item assignment
>>>
```

Observation:

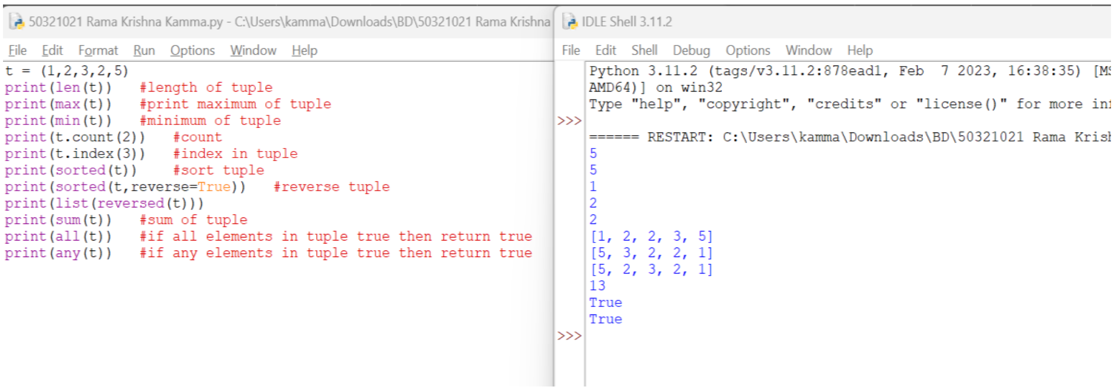
Taken the sample tuple performed indexing, slicing, and negative indexing.

Methods:

Code:

```
t = (1,2,3,2,5)
print(len(t))   #length of tuple
print(max(t))   #print maximum of tuple
print(min(t))   #minimum of tuple
print(t.count(2))   #count
print(t.index(3))   #index in tuple
print(sorted(t))    #sort tuple
print(sorted(t,reverse=True))   #reverse tuple
print(list(reversed(t)))
print(sum(t))   #sum of tuple
print(all(t))   #if all elements in tuple true then return true
print(any(t))   #if any elements in tuple true then return true
```

Output:



50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Krishna
File Edit Format Run Options Window Help
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

```
t = (1,2,3,2,5)
print(len(t))   #length of tuple
print(max(t))   #print maximum of tuple
print(min(t))   #minimum of tuple
print(t.count(2))   #count
print(t.index(3))   #index in tuple
print(sorted(t))    #sort tuple
print(sorted(t,reverse=True))   #reverse tuple
print(list(reversed(t)))
print(sum(t))   #sum of tuple
print(all(t))   #if all elements in tuple true then return true
print(any(t))   #if any elements in tuple true then return true
```

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MS
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf
>>>
====== RESTART: C:\Users\kamma\Downloads\BD\50321021 Rama Krish
5
5
1
2
2
[1, 2, 2, 3, 5]
[5, 3, 2, 2, 1]
[5, 2, 3, 2, 1]
13
True
True
>>>
```

Observation:

Performed various tuple operations like len, max, min, count, sorted, index, all, any methods on sample tuple.

Set:

- A set is an unordered collection of items.
- All the elements in the set are unique (no duplicates) and must be immutable (which cannot be changed).
- However, the set itself is mutable. We can add or remove items from it.
- Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.
- We cannot access or change an element of set using indexing or slicing.

Set Operations:

Sets can be used to carry out mathematical set operations like union, intersection, difference and symmetric difference.

Set Union

- Union of A and B is a set of all elements from both sets.
- Union is performed using | operator.
- Same can be performed using the method union().

Set Intersection:

- Intersection of A and B is a set of elements that are common in both sets.
- Intersection is performed using & operator.
- Same can be accomplished using the method intersection().

Set Difference:

- Difference of A and B (A - B) is a set of elements that are only in A but not in B. Similarly, B - A is a set of element in B but not in A.
- Difference is performed using - operator.
- Same can be accomplished using the method difference().

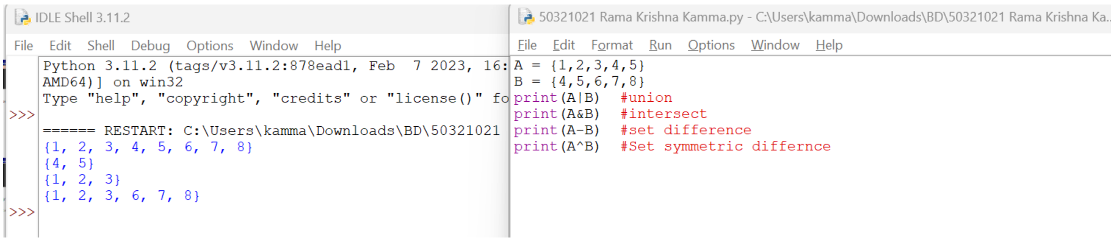
Set Symmetric Difference:

- Symmetric Difference of A and B is a set of elements in both A and B except those that are common in both.
- Symmetric difference is performed using ^ operator.
- Same can be accomplished using the method symmetric_difference().

Code:

```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
print(A|B)  #union
print(A&B)  #intersect
print(A-B)  #set difference
print(A^B)  #Set symmetric difference
```

Output:



IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Ka..
File Edit Format Run Options Window Help

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" fo
>>>
====== RESTART: C:\Users\kamma\Downloads\BD\50321021
{1, 2, 3, 4, 5, 6, 7, 8}
{4, 5}
{1, 2, 3}
{1, 2, 3, 6, 7, 8}
>>>
```

```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
print(A|B)  #union
print(A&B)  #intersect
print(A-B)  #set difference
print(A^B)  #Set symmetric differnce
```

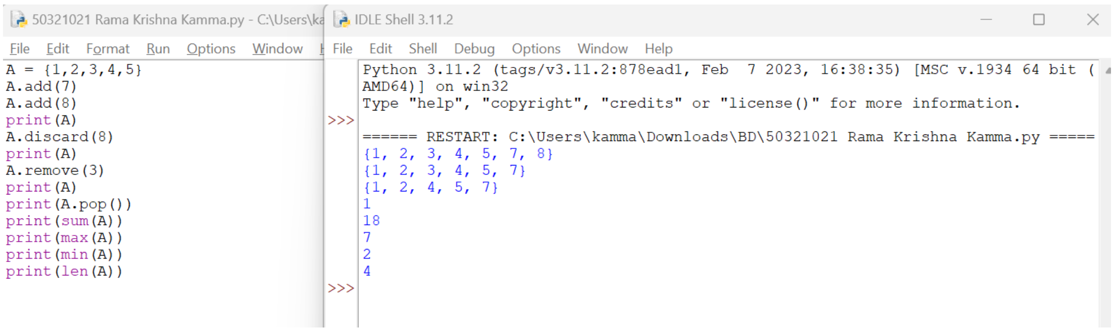
Observation:

Performed various set operations like union, intersect, set difference, set symmetric difference in set.

Code:

```
A = {1,2,3,4,5}
A.add(7)         #add
A.add(8)
print(A)
A.discard(8)     #discard
print(A)
A.remove(3)      #remove element
print(A)
print(A.pop())   #pop element
print(sum(A))    #sum of elements
print(max(A))    #max in set
print(min(A))    #min in set
print(len(A))    #len in set
```

Output:


50321021 Rama Krishna Kamma.py - C:\Users\ka
File Edit Format Run Options Window
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

```
A = {1,2,3,4,5}
A.add(7)
A.add(8)
print(A)
A.discard(8)
print(A)
A.remove(3)
print(A)
print(A.pop())
print(sum(A))
print(max(A))
print(min(A))
print(len(A))
```

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Kamma.py =====
{1, 2, 3, 4, 5, 7, 8}
{1, 2, 3, 4, 5, 7}
{1, 2, 4, 5, 7}
1
18
7
2
4
>>>
```

Observation:

Performed various set methods like add, discard, remove, pop, sum, max, min, len etc.

Dictionaries:

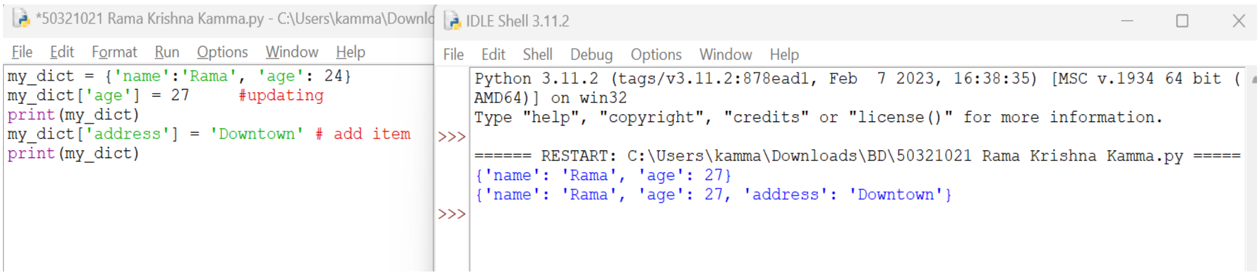
- Python dictionary is an unordered collection of items.
- While other data structures have only value as an element, a dictionary has a key: value pair.
- In Python dictionaries are written with curly brackets, and they have keys and values. It consists of key value pairs.
- The value can be accessed by unique key in the dictionary.
- No duplicate key is allowed.
- The values in the dictionary can be of any type while the keys must be immutable.

Operations:

Code:

```
my_dict = {'name':'Rama', 'age': 24}
my_dict['age'] = 27     #updating
print(my_dict)
my_dict['address'] = 'Downtown' # add item
print(my_dict)
```

Output:


*50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downlo
File Edit Format Run Options Window Help
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

```
my_dict = {'name':'Rama', 'age': 24}
my_dict['age'] = 27     #updating
print(my_dict)
my_dict['address'] = 'Downtown' # add item
print(my_dict)
```

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Kamma.py =====
{'name': 'Rama', 'age': 27}
{'name': 'Rama', 'age': 27, 'address': 'Downtown'}
>>>
```

Observation:

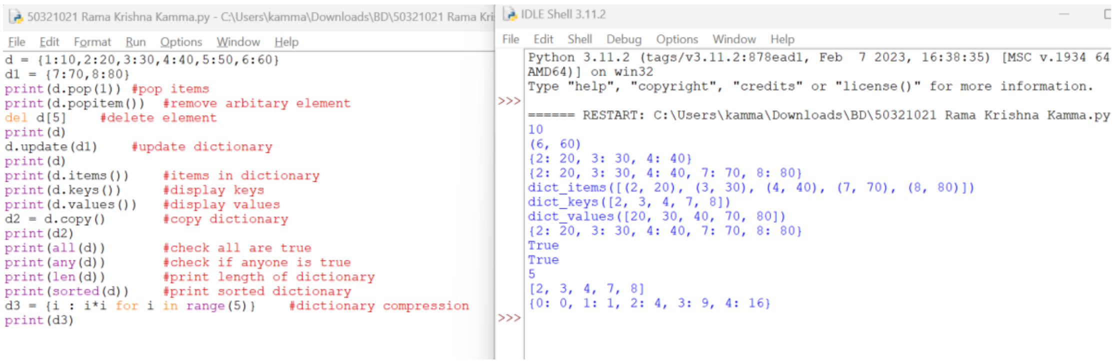
Declaring the dictionary, updating the dictionary and perform various operations.

Methods:

Code:

```
d = {1:10,2:20,3:30,4:40,5:50,6:60}
d1 = {7:70,8:80}
print(d.pop(1)) #pop items
print(d.popitem())  #remove arbitary element
del d[5]    #delete element
print(d)
d.update(d1)    #update dictionary
print(d)
print(d.items())    #items in dictionary
print(d.keys())     #display keys
print(d.values())   #display values
d2 = d.copy()       #copy dictionary
print(d2)
print(all(d))       #check all are true
print(any(d))       #check if anyone is true
print(len(d))       #print length of dictionary
print(sorted(d))    #print sorted dictionary
d3 = {i : i*i for i in range(5)}    #dictionary compression
print(d3)
```

Output:



50321021 Rama Krishna Kamma.py - C:\Users\kamma\Downloads\BD\50321021 Rama Kri
File Edit Format Run Options Window Help
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help

```
d = {1:10,2:20,3:30,4:40,5:50,6:60}
d1 = {7:70,8:80}
print(d.pop(1)) #pop items
print(d.popitem())  #remove arbitary element
del d[5]    #delete element
print(d)
d.update(d1)    #update dictionary
print(d)
print(d.items())    #items in dictionary
print(d.keys())     #display keys
print(d.values())   #display values
d2 = d.copy()       #copy dictionary
print(d2)
print(all(d))       #check all are true
print(any(d))       #check if anyone is true
print(len(d))       #print length of dictionary
print(sorted(d))    #print sorted dictionary
d3 = {i : i*i for i in range(5)}    #dictionary compression
print(d3)
```

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\kamma\Downloads\BD\50321021 Rama Krishna Kamma.py
10
(6, 60)
{2: 20, 3: 30, 4: 40}
{2: 20, 3: 30, 4: 40, 7: 70, 8: 80}
dict_items([(2, 20), (3, 30), (4, 40), (7, 70), (8, 80)])
dict_keys([2, 3, 4, 7, 8])
dict_values([20, 30, 40, 70, 80])
{2: 20, 3: 30, 4: 40, 7: 70, 8: 80}
True
True
5
[2, 3, 4, 7, 8]
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
>>>
```

Observation:

Performed the various dictionary methods like pop, popitem, delete, items, keys, values, copy, all, any, len, sorted etc.