**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

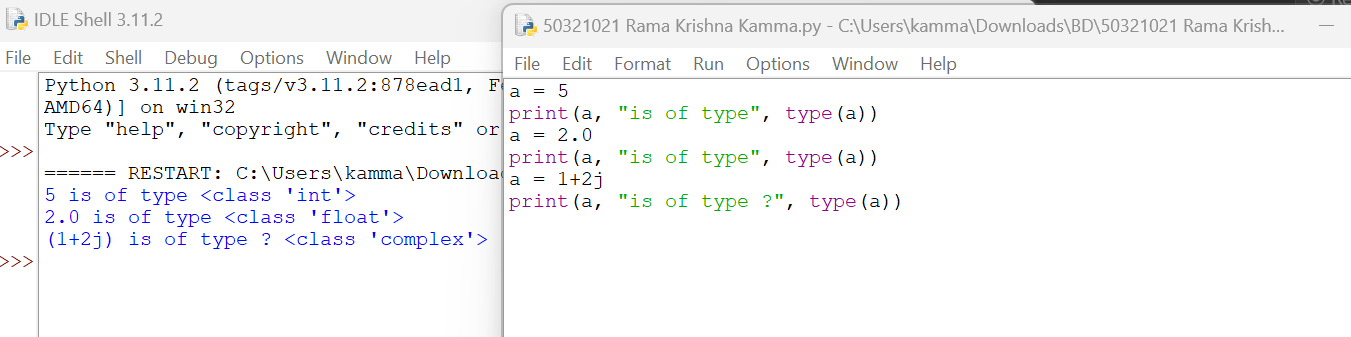
print(a, "is of type", type(a))

a = 2.0

print(a, "is of type", type(a))

a = 1+2j

print(a, "is of type ?", type(a))

Output: 

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:

Graphical user interface, text, application

Description automatically generated

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:

Graphical user interface, application

Description automatically generated

**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:

Graphical user interface, text, application

Description automatically generated

**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:

Text

Description automatically generated

**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:

Text

Description automatically generated

**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:

Text, application

Description automatically generated

**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:

Text

Description automatically generated

**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:

A picture containing scatter chart

Description automatically generated

**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

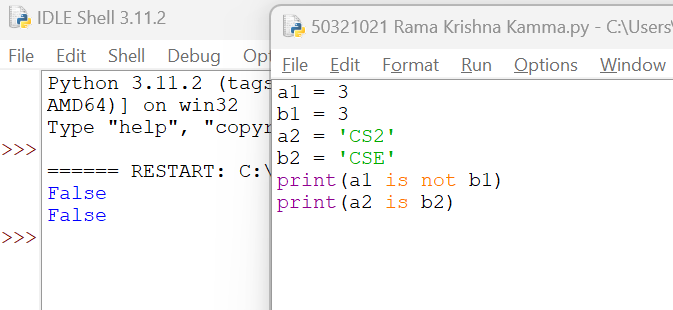
a2 = 'CS2'

b2 = 'CSE'

print(a1 is not b1)

print(a2 is b2)

Output:



**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:

Graphical user interface, text, application

Description automatically generated

**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:

Graphical user interface, text, application

Description automatically generated

**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:

Text

Description automatically generated

**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:

Graphical user interface, application

Description automatically generated with medium confidence

**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:

Graphical user interface, application, Word

Description automatically generated

**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}

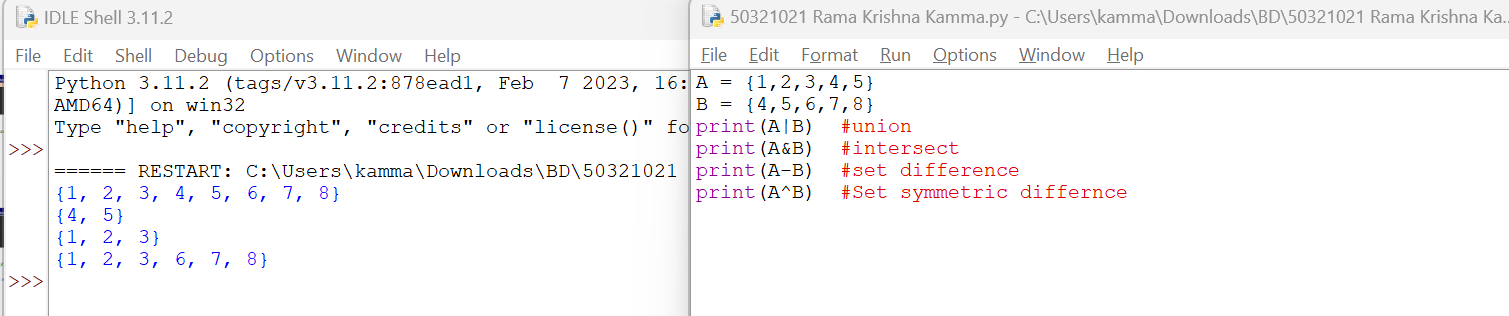
print(A|B) #union

print(A&B) #intersect

print(A-B) #set difference

print(A^B) #Set symmetric difference

Output:



**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:

A picture containing text

Description automatically generated

**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}

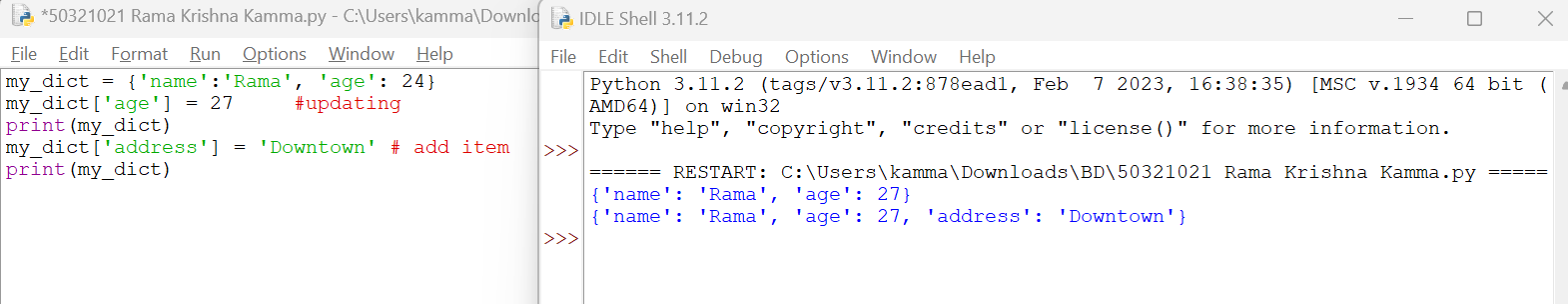
my\_dict['age'] = 27 #updating

print(my\_dict)

my\_dict['address'] = 'Downtown' # add item

print(my\_dict)

Output:



**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:

Graphical user interface, application

Description automatically generated

**Observation:**

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