Experiment 02

Write a python program to understand

2.1) Different List, Tuple, Dictionary operations using Built-in functions

2.2) Built-in set and String, Range functions

|  |  |
| --- | --- |
| Roll No. | 01 |
| Name | Aamir Ansari |
| Class | D10-A |
| Subject | Python Lab |
| LO Mapped | LO1: Understand the structure, syntax, and semantics of the Python language  LO2: Interpret advanced data types and functions in python |

**Aim**:

Write a python program to understand

2.1) Different List, Tuple, Dictionary operations using Built-in functions

2.2) Built-in set and String, Range functions

**Introduction**:

1. A data type in which the values are made of components, or elements, that are themselves values

2. A compound data type is immutable, i.e elements cannot be assigned new values

3. A default value is given if no argument for it is provided in the function call

4. Examples: List, Tuple, Dictionary, String, Set

## 2.1) Different List, Tuple, Dictionary operations using Built-in functions

### 2.1.1) Lists:

I. Lists are used to store multiple items in a single variable

II. Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage

Lists are created using square brackets:

ex. listOne = ["apple", "banana", "cherry"]

List Items

I. List items are ordered, changeable, and allow duplicate values.

II. List items are indexed, the first item has index [0], the second item has index [1] etc

III. List items can be of any data type

IV. A list can contain different data types

Operations on list:

**(1) append()**

list.append(obj)

Add an element to the end of the list

**(2) extend()**

list.extend(sequence)

Add all elements of a list to the another list

**(3) insert()**

list.insert(index, obj)

Insert an item at the defined index

**(4) remove()**

list.remove(obj)

Removes an item from the list

**(5) pop()**

list.pop(obj)

Removes and returns an element at the given index

**(6) clear()**

list.clear()

removes all the items from the list

**(7) index()**

list.index(obj)

Returns the index of the first matched iter

**(8) count()**

list.cont(obj)

Returns the count of number of items passes as an argument

**(9) sort()**

list.sort([func])

Sort items in a list in ascending order

**(10) cmp()**

cmp(list1, list2)

Compares elements of both lists

**(11) list()**

list(seq)

Converts tuple into list

**(12) len()**

len(list)

Returns total length of list

**(13) max()**

max(list)

Returns item from the list with the max value

**(14) min()**

min(list)

Returns item from the list with the min value

\*\*\*

### 2.1.2) Tuple:

I. A Tuple is a collection of Python objects separated by commas

II. A Tuple is similar to a list in terms of indexing, nested objects and repetition

III. A tuple is immutable unlike lists which are mutable

Operations on Tuple

**(1) all()**

all(tuple)

Return True if all elements of the tuple are true (or if the tuple is empty)

**(2) any()**

any(tuple)

Return True if any elements of the tuple is true if the tuple is empty returns false

**(3) enumerate()**

enumerate(tuple)

Return an enumerate object. It contains the index and value of all the items of tuple as pairs

**(4) len()**

len(tuple)

Returns the number of items in the tuple

**(5) max()**

max(tuple)

Returns the largest item in the tuple

**(6) min()**

min(tuple)

Returns the smallest item in the tuple

**(7) sorted()**

sorted(tuple)

Take elements it the tuple and return a new sorted list

**(8) sum()**

sum(tuple)

Returns the sum of all elements in a tuple

**(9) tuple()**

tuple(interable)

Convert an iterable (list, string, set, dictionary) to a tuple

**(10) cmp()**

cmp(tuple1, tuple2)

Compares elements of both tuple

\*\*\*

### 2.1.3) Dictionary:

I. Dictionary in Python is an unordered collection of data value

II. Dictionary is used to store data values like a map, which unlike other Data Types that hold only single value as an element

III. Dictionary holds key : value pair

IV. Key value is provided in the dictionary to make it more optimized.

V. Keys in a dictionary doesn’t allows Polymorphism

Operations on Dictionary

**(1) all()**

all(dictionary)

Return True if all elements of the tuple are true (or if the tuple is empty)

**(2) any()**

any(dictionary)

Return True if any elements of the tuple is true if the tuple is empty returns false

**(3) len()**

len(dictionary)

Return the number of items in the dictionary

**(4) cmp()**

cmp(dictionary, dictionary)

Compares the items of two dictionaries

**(5) sorted()**

sorted(dictionary)

Returns a new sorted list of keys in the dictionary

**(6) clear()**

clear()

Removes all the elements from the dictionary

**(7) copy()**

copy()

Returns a copy of the dictionary

**(8) get()**

dictionary.get(obj)

Returns the value of specified key

**(9) items()**

idictionary.tems()

Returns a list containing the dictionary’s key value pair

**(10) keys()**

dictionary.keys)

Returns a list containing the dictionary’s key

**(11) pop()**

dictionary.pop(key)

Removes the element with specified key

**(12) update()**

dictionary.update(obj)

Updates the dictionary with the specified key-value pairs

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 2.2) Built-in set and String, Range functions

### 2.2.1) Set:

1. Sets are used to store multiple items in a single variable.
2. Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.
3. A set is a collection which is both unordered and unindexed.
4. Sets are written with curly brackets.

Operations on Set

**(1) add()**

add(obj)

Adds an element to the set

**(2) difference()**

difference(set1, set2,...)

Returns a set containing the difference between two or more sets

**(3) discard()**

discard(obj)

Remove the specified item

**(4) intersection()**

intersection(set1, set2)

Returns a set, that is the intersection of two other sets

**(5) isdisjoint()**

isdisjoint(set1, set2)

Returns whether two sets have a intersection or not

**(6) issubset()**

issubset(set1, set2)

Returns whether another set contains this set or not

**(7) issuperset()**

issuperset(set1, set2)

Returns whether this set contains another set or not

**(8) union()**

union(set1, set2)

Return a set containing the union of sets

**(9) update()**

update(set1, set2)

update the set with the union of this set and others

**(10) clear()**

clear(set)

Removes all the elements from the set

**(11) pop()**

pop(obj)

Return an element from the set

**(12) remove()**

remove(obj)

Removes the specified element

### 2.2.2) String:

1. A string is a sequence of characters
2. In Python, a string is a sequence of Unicode characters
3. Strings can be created by enclosing characters inside a single quote or double-quotes
4. Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

Operations on String

**(1) capitalize()**

Converts the first character to uppercase

**(2) casefold()**

Converts string into lower case

**(3) center()**

Returns a centered string

**(4) count()**

Returns the number of times a specified value occurs in a string

**(5) encode()**

Returns an encoded version of the string

**(6) endswith()**

Returns true if the string ends with the specified value

**(7) expandtabs()**

Sets the tab size of the string

**(8) find()**

Searches the string for a specified value and returns the position of where it was found

**(9) format()**

Formats specified values in a string

**(10) format\_map()**

Formats specified values in a string

**(11) index()**

Searches the string for a specified value and returns the position of where it was found

**(12) isalnum()**

Returns True if all characters in the string are alphanumeric

**(13) isalpha()**

Returns True if all characters in the string are in the alphabet

**(14) isdecimal()**

Returns True if all characters in the string are decimals

**(15) isdigit()**

Returns True if all characters in the string are digits

**(16) isidentifier()**

Returns True if the string is an identifier

**(17) islower()**

Returns True if all characters in the string are lower case

**(18) isnumeric()**

Returns True if all characters in the string are numeric

**(19) isprintable()**

Returns True if all characters in the string are printable

**(20) isspace()**

Returns True if all characters in the string are whitespaces

**(21) istitle()**

Returns True if the string follows the rules of a title

**(22) isupper()**

Returns True if all characters in the string are upper case

**(23) join()**

Joins the elements of an iterable to the end of the string

**(24) ljust()**

Returns a left justified version of the string

**(25) lower()**

Converts a string into lower case

**(26) lstrip()**

Returns a left trim version of the string

**(27) maketrans()**

Returns a translation table to be used in translations

**(28) partition()**

Returns a tuple where the string is parted into three parts

**(29) replace()**

Returns a string where a specified value is replaced with a specified value

**(30) rfind()**

Searches the string for a specified value and returns the last position of where it was found

**(31) rindex()**

Searches the string for a specified value and returns the last position of where it was found

**(32) rjust()**

Returns a right justified version of the string

**(33) rpartition()**

Returns a tuple where the string is parted into three parts

**(34) rsplit()**

Splits the string at the specified separator, and returns a list

**(35) rstrip()**

Returns a right trim version of the string

**(36) split()**

Splits the string at the specified separator, and returns a list

**(37) splitlines()**

Splits the string at line breaks and returns a list

**(38) startswith()**

Returns true if the string starts with the specified value

**(39) strip()**

Returns a trimmed version of the string

**(40) swapcase()**

Swaps cases, lower case becomes upper case and vice versa

**(41) title()**

Converts the first character of each word to upper case

**(42) translate()**

Returns a translated string

**(43) upper()**

Converts a string into upper case

**(44) zfill()**

Fills the string with a specified number of 0 values at the beginning

\*\*\*

### 2.2.2) Range:

1. The range() function is a built-in-function used in python
2. range is used to generate a sequence of numbers
3. If the user wants to generate a sequence of numbers given the starting and the ending values then they can give these values as parameters of the range() function
4. The range() function will then generate a sequence of numbers according to the user's requirement
5. Syntax

**range ( start , stop , step );**

There are three parameters inside the range():

start

stop

step

When you think about these three-parameters, it resembles a real-life scenario that would be discussed down below.

**Start**: Optional :: An integer number that specifies where to start (Default value is 0)

**Stop**: Required :: An integer number that specifies where to stop.

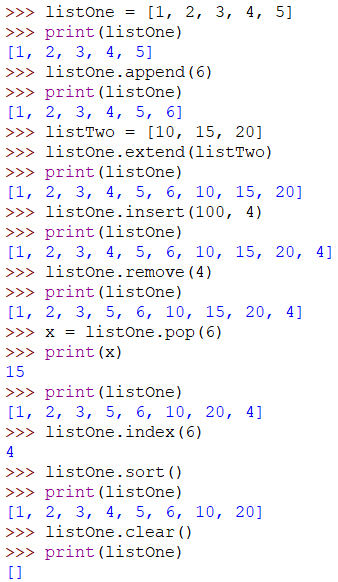
**Step**: Optional :: An integer number that specifies how much to increment the number (Default value is 1)

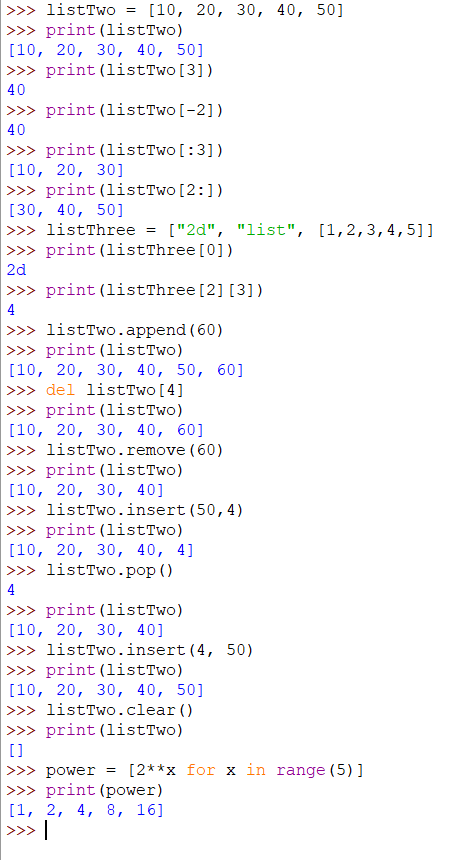
The return value of the range function is a sequence of numbers depending upon the parameters defined

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

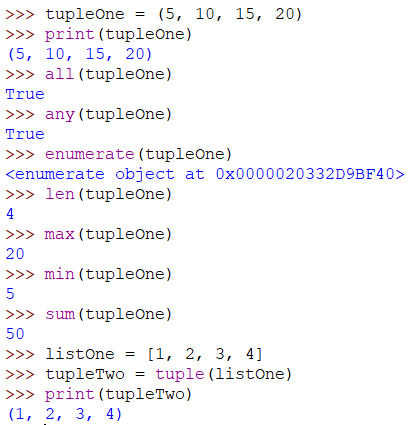
## **Results**:

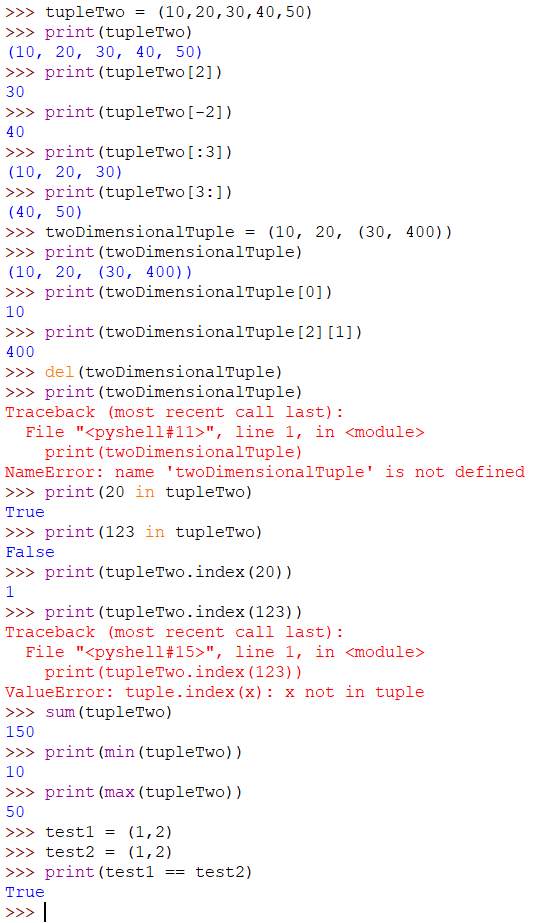
### List



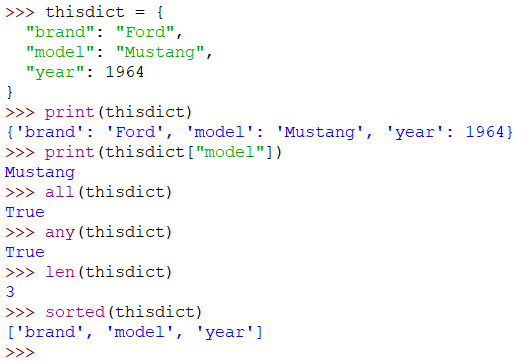


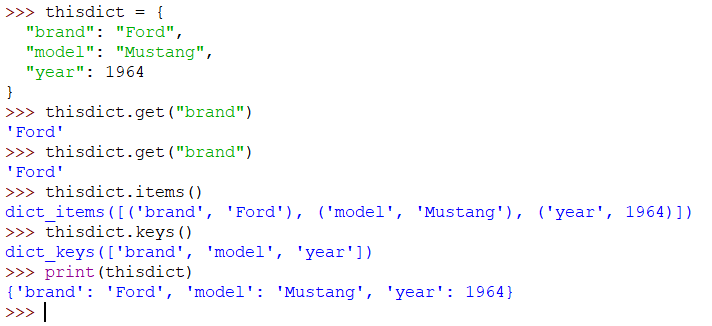
### Tuple

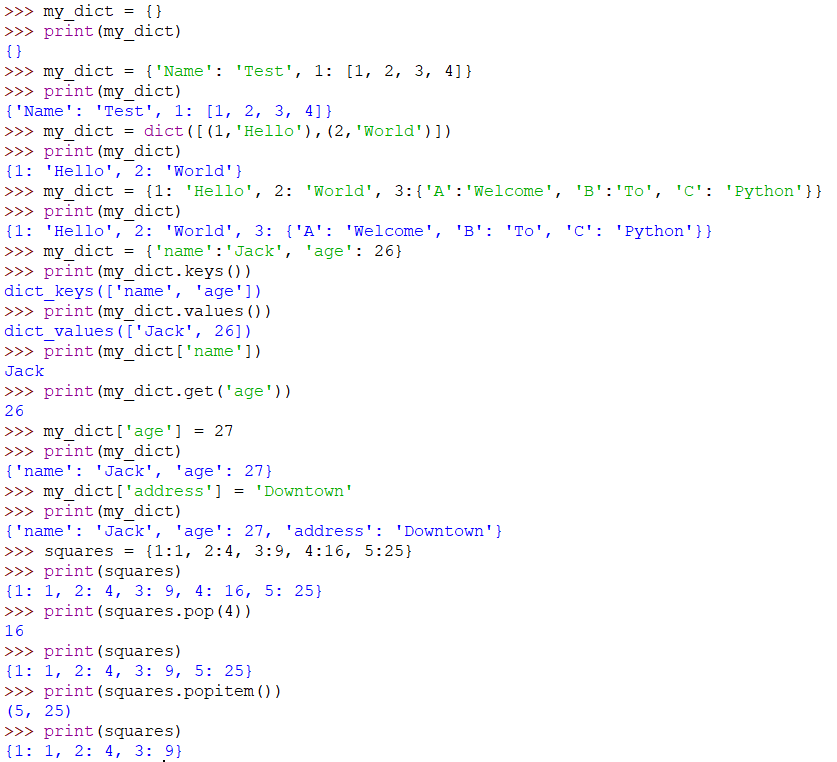


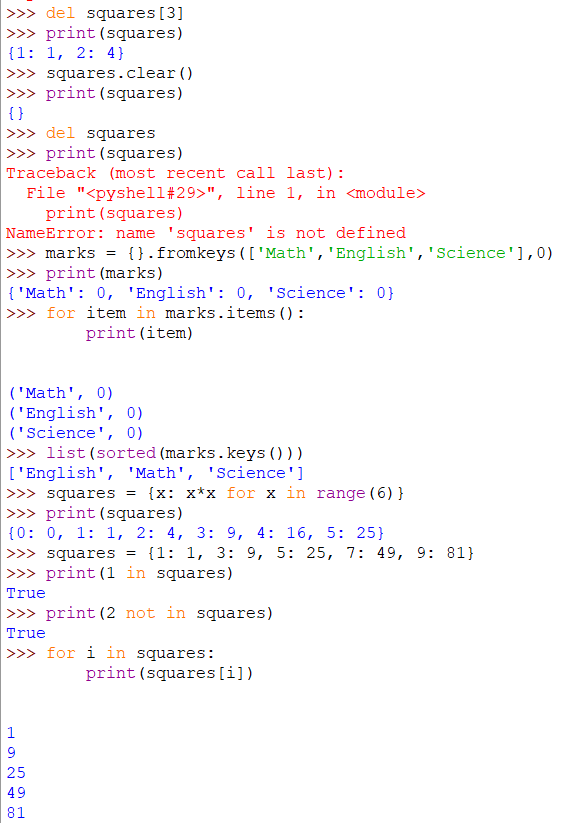


### Dictionary

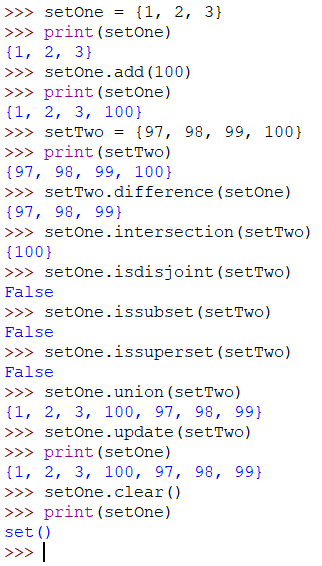




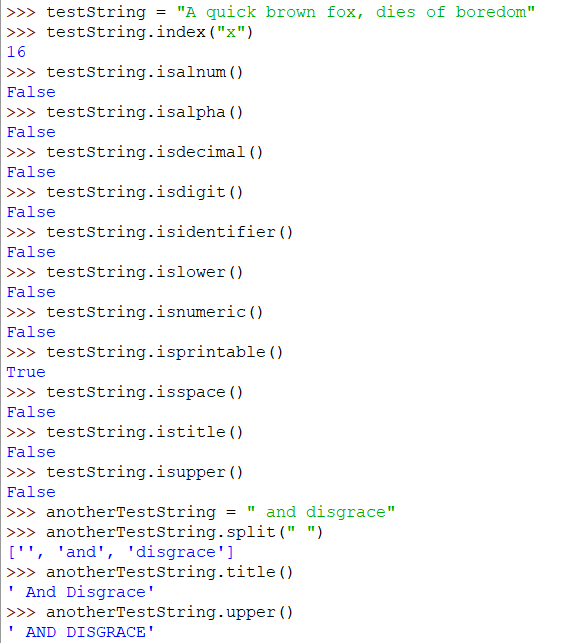


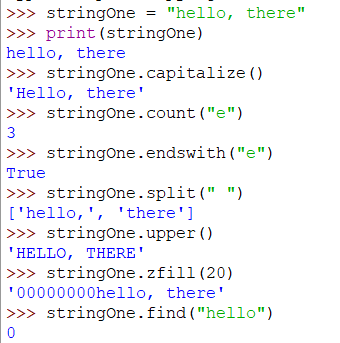


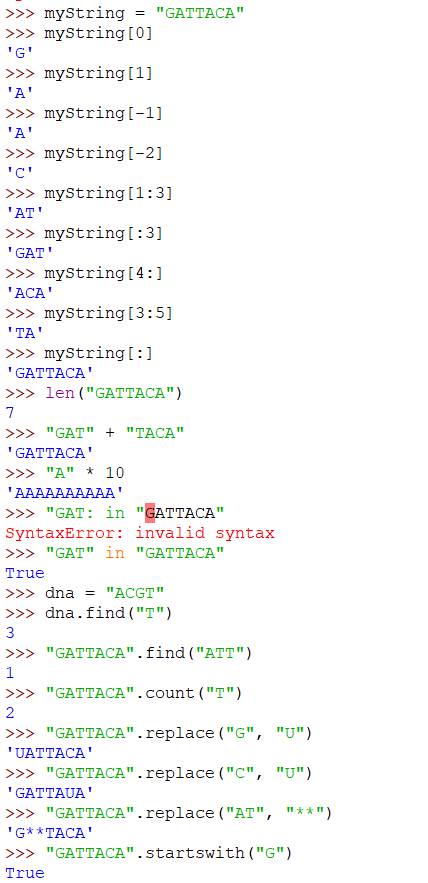
### Set

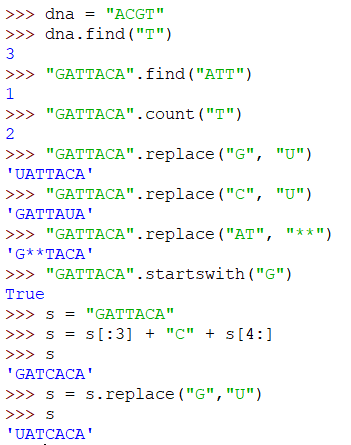


### String

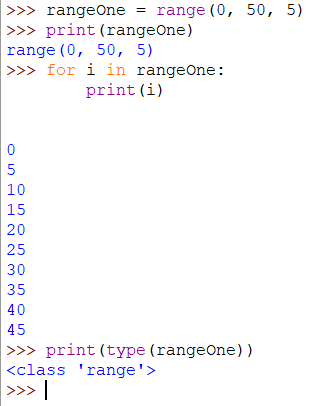


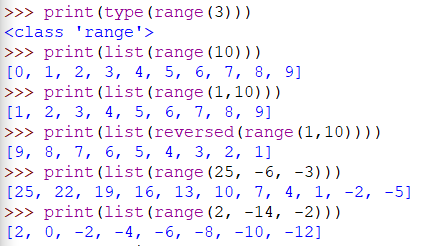






### Range





## **Conclusion**:

Hece, we have successfully studied operations on List, Tuple and Dictionary, Set, String and Range using Built-in functions