Unit-III

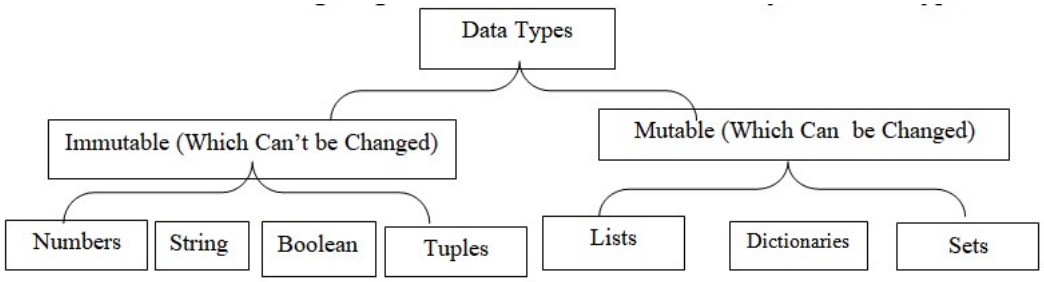
Syllabus: Data Structures Mutable and Immutable data structures, declaring and using numeric data types: int, float, complex. Strings, list, tuple, dictionary, set and string: usage, conversions, built-in methods and differences, list and dictionary comprehensions.

Learning Material

3.1 Mutable and Immutable data structures:

3.1.1 Classification

- The following diagram shows the classification of data types:



3.1.2 Differences

- An object whose internal state (data) can be changed is called a **Mutable object**.
- Similarly, an object whose internal state (data) can not be changed is called an **Immutable object**.
- lists, sets & dictionaries are mutable, so we can add, delete & modify its contents.

Example

```
my_list = [1,2,3]
print(my_list)
print(id(my_list))
my_list.append(4)
print(my_list)
print(id(my_list))
>>> [1,2,3]
>>> 1863010210504
>>> [1,2,3,4]
>>> 1863010210504
```

- Numbers, Strings & Tuples are immutable, so internal state (data) can not be changed.
- In the examples below, the memory location (address) gets changed because a new object gets created at different memory location during the operation.

Example

1. n = 10
print(n)
print(id(n))
n = n + 10
print(n)
print(id(n))
>>> 10
>>> 140713425216176
>>> 20
>>> 140713425216496

2. my_tuple = (10,20,30)
print(my_tuple)
print(id(my_tuple))
my_tuple = my_tuple + (40,)
print(my_tuple)
print(id(my_tuple))
>>> (10, 20, 30)
>>> 2689034125544
>>> (10, 20, 30, 40)
>>> 2689034073256

3.2 Declaring and using numeric data types: int, float, complex

- Python does not have type declaration statements.
- Data type of a variable is determined based on the value assigned (Dynamic Typing).

Examples:

```
>>> a = 10          >>> b = 3.6          >>> c = 3+4j
>>> a               >>> b               >>> c
10                  3.6                 (3+4j)
>>> type(a)         >>> type(b)         >>> type(c)
<class 'int'>       <class 'float'>     <class 'complex'>
```

3.3 Strings:

- String is a contiguous series of characters delimited by single, double or even triple quotes.
- Python has a built-in string class named "str" that has many useful features.

3.3.1 Traversing

- A string can be traversed by accessing character(s) from one index to another.
- For example, the following program uses indexing to traverse a string from first character to the last.

```
message = "Hello!"
index = 0
for i in message:
print("message[", index, "] = ", i)
index += 1
```

OUTPUT

```
message[ 0 ] =  H
message[ 1 ] =  e
message[ 2 ] =  l
message[ 3 ] =  l
message[ 4 ] =  o
message[ 5 ] =  !
```

3.3.2 Operations of Strings (Concatenating, Appending and Multiplying Strings)

Example: Program to concatenate two strings using + operator

```
str1 = "Hello "
str2 = "World"
str3 = str1 + str2
print("The concatenated string is : ", str3)
```

OUTPUT

```
The concatenated string is :  Hello World
```

Example: Program to repeat a string using * operator

```
str = "Hello"
print(str * 3)
```

OUTPUT

```
Hello Hello Hello
```

Example: Program to append a string using += operator

```
str = "Hello, "
name = input("\n Enter your name : ")
str += name
str += ". Welcome to Python Programming."
print(str)
```

OUTPUT

```
Enter your name : Arnav
Hello, Arnav. Welcome to Python Programming.
```

3.3.3 Built-in Methods

Function	Usage	Example

`capitalize()`	This function is used to capitalize first letter of the string.	`str = "hello"`
		`print(str.capitalize())`
		OUTPUT
		`Hello`

`center(width, fillchar)`	Returns a string with the original string centered to a total of `width` columns and filled with `fillchar` in columns that do not have characters.	`str = "hello"`
		`print(str.center(10, '*'))`
		OUTPUT
		`**hello***`

`find(str, beg, end)`	Checks if `str` is present in string. If found it returns the position at which `str` occurs in string, otherwise returns `-1`. You can either set `beg = 0` and `end` equal to the length of the message to search entire string or use any other value to search a part of it.	`message = "She is my best friend"`
		`print(message. find("my",0, len (message)))`
		OUTPUT
		`7`

`index(str, beg, end)`	Same as find but raises an exception if `str` is not found.	`message = "She is my best friend"`
		`print(message.index("mine", 0, len(message)))`
		OUTPUT
		`ValueError: substring not found`

`rfind(str, beg, end)`	Same as find but starts searching from the end.	`str = "Is this your bag?"`
		`print(str.rfind("is", 0, len(str)))`
		OUTPUT
		`5`

`rindex(str, beg, end)`	Same as `rindex` but start searching from the end and raises an exception if `str` is not found.	`str = "Is this your bag?"`
		`print(str.rindex("you", 0, len(str)))`
		OUTPUT
		`8`

`count(str, beg, end)`	Counts number of times str occurs in a string. You can specify beg as 0 and end as the length of the message to search the entire string or use any other value to just search a part of the string.	`str = "he"`
		`message = "helloworldhellohello"`
		`print(message. count (str,0, len (message)))`
		OUTPUT
		`3`

`endswith (suffix, beg, end)`	Checks if string ends with suffix; returns `True` if so and `False` otherwise. You can either set `beg = 0` and `end` equal to the length of the message to search entire string or use any other value to search a part of it.	`message = "She is my best friend"`
		`print(message.endswith("end", 0,len(message)))`
		OUTPUT
		`True`

`isalnum()`	Returns True if string has at least 1 character and every character is either a number or an alphabet and False otherwise.	`message = "JamesBond007"`
		`print(message.isalnum())`
		OUTPUT
		`True`

`isalpha()`	Returns True if string has at least 1 character and every character is an alphabet and False otherwise.	`message = "JamesBond007"`
		`print(message.isalpha())`
		OUTPUT
		`False`

`isdigit()`	Returns True if string contains only digits and False otherwise.	`message = "007"`
		`print(message.isdigit())`
		OUTPUT
		`True`

`islower()`	Returns True if string has at least 1 character and every character is a lowercase alphabet and False otherwise.	`message = "Hello"`
		`print(message.islower())`
		OUTPUT
		`False`

`isspace()`	Returns True if string contains only whitespace characters and False otherwise.	`message = "  "`
		`print(message.isspace())`
		OUTPUT
		`True`

`isupper()`	Returns True if string has at least 1 character and every character is an upper case alphabet and False otherwise.	`message = "HELLO"`
		`print(message.isupper())`
		OUTPUT
		`True`

`len(string)`	Returns the length of the string.	`str = "Hello"`
		`print(len(str))`
		OUTPUT
		`5`

`ljust(width[, fillchar])`	Returns a string left-justified to a total of `width` columns. Columns without characters are padded with the character specified in the `fillchar` argument.	`str = "Hello"`
		`print(str.ljust(10, '*'))`
		OUTPUT
		`Hello*****`

`rjust(width[, fillchar])`	Returns a string right-justified to a total of `width` columns. Columns without characters are padded with the character specified in the `fillchar` argument.	`str = "Hello"`
		`print(str.rjust(10, '*'))`
		OUTPUT
		`*****Hello`

3.4 List:

- List is the most basic data structure in Python.
- It is a sequence of values represented using [ ].
- Each element of a list is assigned a number - its position or index starting from 0.

3.4.1 Accessing List

- To access values in lists, square brackets are used.
- A list value can be accessed using positive or negative index.
- Positive index ranges from 0 to n-1 where n is the size of the list.
- Negative index ranges from -1 to –n.
- Index -1 refers to the last element in the list
- Index –n refers to the first elements in the list.

Example

```
li = [10, 20, 30, 40, 50]
print("list =", li, "\ntype =", type(li))
print("Accessing elements in a list using positive index")
n = len(li)
for i in range(n):
   print("li[",i,"] =",li[i], end=" ")
print("Accessing elements in a list using negative index")
print("First element:", li[-5])
print("Last element:", li[-1])
for i in range(-n,0):
   print("li[",i,"] =",li[i], end=" ")
print("\nAccessing elements in reverse order")
for i in range(-1,-n-1,-1):
   print("li[",i,"] =",li[i], end=" ")
```

Output

```
list = [10, 20, 30, 40, 50]
type = <class 'list'>
Accessing elements in a list using positive index
li[ 0 ] = 10 li[ 1 ] = 20 li[ 2 ] = 30 li[ 3 ] = 40 li[ 4 ] = 50
Accessing elements in a list using negative index
First element: 10
Last element: 50
li[ -5 ] = 10 li[ -4 ] = 20 li[ -3 ] = 30 li[ -2 ] = 40 li[ -1 ] = 50
Accessing elements in reverse order
li[ -1 ] = 50 li[ -2 ] = 40 li[ -3 ] = 30 li[ -4 ] = 20 li[ -5 ] = 10
```

- To slice along with index or indices to get value stored at that index or range of indices.

Syntax

```
list[start:stop:step]
```

Example

```
num_list=[1,2,3,4,5,6,7,8,9,10]
print(“num_list is:”,num_list)
print(“first elemnent in the list is”,num_list[0])
print(“num_list[2:5]=”,num_list[2:5])
print(“num_list[::2]=”,num_list[::2])
print(“num_list[1::3]=”,num_list[1::3])
```

Output:

```
num_list is: [1,2,3,4,5,6,7,8,9,10]
first elemnent in the list is 1
```

num_list[2:5]= [3,4,5]
num_list[::2]= [1,3,5,7,9]
num_list[1::3]= [2,5,8]

3.4.2 Updating List

- Elements of a list can be easily updated by giving the slice on the left-hand side of the assignment operator.
- You can also append new values in the list and remove existing values from the list using the append( ) method and del statement respectively.

Example

1) num_list= [1,2,3,4,5,6,7,8,9,10]
   print("list is:",num_list)
   num_list[5]=100
   print("List after updation is:",num_list)
   num_list.append(200)
   print("List after appending a value is: ",num_list)
   del num_list[3]
   print("List after deleting a value is:",num_list)

Output:

list is: [1,2,3,4,5,6,7,8,9,10]
List after updation is: [1,2,3,4,5,100,7,8,9,10]
List after appending a value is: [1,2,3,4,5,100,7,8,9,10,200]
List after deleting a value is: [1,2,3,5,100,7,8,9,10,200]

3.4.3 Basic List Operations

Operation	Description	Example	Output
len	Returns length of list	len([1,2,3,4,5,6,7,8,9,10])	10
concatenation	Joins two lists	[1,2,3,4,5]+[6,7,8,9,10]	[1,2,3,4,5,6,7,8,9,10]
repetition	Repeats elements in the lists	"Hello","World"*2	['Hello','World','Hello','World']
in	Checks if the value is present in the list	'a' in['a','e','i','o','u']	True
not in	Checks if the value is not present in the list	3 not in [0,2,4,6,8]	True
max	Returns maximum value in the list	num_list=[6,3,7,0,1,2,4,9] print(max(num_list))	9
min	Returns minimum	num_list=[6,3,7,0,1,2,4,9]	0

	value in the list	print(min(num_list))	
sum	Adds the values in the list that has numbers	num_list=[1,2,3,4,5,6,7,8,9,10] print("SUM=",sum(num_list))	SUM=55
all	Returns True if all elements of the list are true(or if the list is empty)	num_list=[0,1,2,3] print(all(num_list))	False
any	Returns True if any element of the list is true. if the list is empty return false	num_list=[6,3,7,0,1,2,4,9] print(any(num_list))	True
list	Converts iterable(tuple,string, set,dictionary)	list1=list("HELLO") print(list1)	['H','E','L','L','O']
sorted	Returns a new sorted list. The original list not sorted	list1=[3,4,1,2,7,8] list2=sorted(list1) print(list2)	[1,2,3,4,7,8]

3.5 Tuple:

- A tuple is a sequence of immutable objects. That is, you can change the value of one or more items in a list; you cannot change the values in a tuple.
- Tuples use parenthesis to define its elements.

3.5.1 Utility of Tuples

- Since tuples are immutable, iterating through tuples is faster than iterating over a list. This means that tuples perform better than a list.
- They are best suited for storing data that is write-protected.
- They can be used in place of lists where the number of values is known and small.
- Multiple values can be returned from a function using a tuple.
- They are used to format strings.
- Generally tuples are used as key for a dictionary.

3.5.2 Accessing

- The tuple values can be accessed using slice operation along with the index using square brackets.
- Positive or negative indexing can be used just like lists and strings.

Example

```
Tup1=(1,2,3,4,5,6,7,8,9,10)
print("Tup[5]=",Tup1[5])
```

```
print("Tup[-5]=",Tup1[-5])
print("Tup[3:6]=",Tup1[3:6])
print("Tup[:4]=",Tup1[:4])
print("Tup[4:]=",Tup1[4:])
print("Tup[:]=",Tup1[:])
```

Output

```
Tup[5]=6
Tup[-5]=6
Tup[3:6]=(4,5,6)
Tup[:4]=(1,2,3,4)
Tup[4:]=5,6,7,8,9,10)
Tup[:]=(1,2,3,4,5,6,7,8,9,10)
```

3.5.3 Updating

- It is not possible to update the values but we can just extract the values from a tuple to form another tuple or form a list from tuple, make changes and recreate the tuple.

Examples

1)
```
Tup1=(1,2,3,4,5)
Tup2=(6,7,8,9,10)
Tup3=Tup1+Tup2
print(Tup3)
```

Output:

```
(1,2,3,4,5,6,7,8,9,10)
```

2)
```
tup1 = (1,2,3,4,5)
print(tup1)
list1 = list(tup1)
list1[1] = 10
tup1 = tuple(list1)
print(tup1)
```

Output:

```
(1, 2, 3, 4, 5)
(1, 10, 3, 4, 5)
```

3.5.4 Deleting

- Deleting a single element in a tuple is not possible.
- Hence there is another option to delete a single element of a tuple i.e..,you can create a new tuple that has all elements in your tuple except the ones you don't want.

Example:

1)
```
Tup1=(1,2,3,4,5)
del Tup1[3]
print Tup1
```

Output:

```
Traceback (most recent call last):
File "test.py", line 9, in <module>
del Tup1[3]
Type error: 'tuple' object doesn't support item deletion
```

2) However, you can always delete the entire tuple by using del statement.

```
Tup1=(1,2,3,4,5)
del Tup1
print Tup1
```

Output:

```
Traceback (most recent call last):
File "test.py", line 9, in <module>
print Tup1;
NameError: name 'Tup1' is not defined
```

3.5.5 Basic Tuple Operations

Operation	**Expression**	**Output**
Length	len((1,2,3,4,5,6))	6
Concatenation	(1,2,3)+(4,5,6)	(1,2,3,4,5,6)
Repetition	('Good..')*3	'Good ..Good..Good'
Membership	5 in (1,2,3,4,5,6,7,8,9)	True
Iteration	for i in (1,2,3,4,5,6,7,8,9,10): print(i,end=' ')	1,2,3,4,5,6,7,8,910
Comparision(Use >,<,==)	Tup1=(1,2,3,4,5) Tup2=(1,2,3,4,5) print(Tup1>Tup2)	False
Maximum	max(1,0,3,8,2,9)	9
Minimum	min(1,0,3,8,2,9)	0
Convert to tuple(converts a sequence into a tuple)	tuple("Hello") tuple([1,2,3,4,5])	('H','e','l','l','o') (1,2,3,4,5)
Sorting(The sorted( ) function takes elements in a tuple and returns a new sorted list (does not sort the tuple itself)).	t=(4,67,9) sorted(t)	[4, 9, 67]

3.6 Dictionary:

- It is a data structure in which we store values as a pair of key and value.
- Each key is separated from its value by a colon (:), and consecutive items are separated by commas.
- The entire items in a dictionary are enclosed in curly brackets ({}).

  Syntax:

  dictionary_name = {key_1: value_1, key_2: value_2, key_3: value_3}
- If there are many keys and values in dictionaries, then we can also write just one key-value pair on a line to make the code easier to read and understand. This is shown below.

  dictionary_name = {key_1: value_1, key_2: value_2, key_3: value_3 , ….}
- Keys in the dictionary must be unique and be of any immutable data type (like Strings, numbers, or tuples), there is no strict requirement for uniqueness and type of values.
- Values of a key can be of any type.
- Dictionaries are not Sequences, rather they are mappings.
- **Mappings** are collections of objects that are store objects by key instead of by relative position.

3.6.1 Accessing

- In Dictionary, values are accessed through keys.

Example:

```
d={'Name': 'Arav', 'Course': 'B.tech', 'roll_no': '18/001'}
print('d[Name]:',d['Name'])
print('d[course]:',d['Course'])
print('d[roll_no]:',d['roll_no'])
```

Output:

```
d[Name]: Arav
d[course]: B.tech
d[roll_no]: 18/001
```

3.6.2 Adding and Modifying an item

- To add a new entry or a key-value pair in a dictionary, just specify the key-value pair as you had done for the existing pairs.

  Syntax: *dictionary_ variable[key]= val*

Example:

```
d={'Name': 'Arav', 'Course': 'B.tech', 'roll_no': '18/001'}
d['marks']=99                    #new entry
```

```
print('d[Name]:',d['Name'])
print('d[course]:',d['Course'])
print('d[roll_no]:',d['roll_no'])
print('d[marks]:',d['marks'])
```

Output:

```
d[Name]: Arav
d[course]: B.tech
d[roll_no]: 18/001
d[marks]: 99
```

- To modify an entry, just overwrite the existing value as shown in the following

Example:

```
d={'Name': 'Arav', 'Course': 'B.tech', 'roll_no': '18/001'}
d['marks']=99                    #new entry
print('d[Name]:',d['Name'])
print('d[course]:',d['Course'])
print('d[roll_no]:',d['roll_no'])
print('d[marks]:',d['marks'])
d['Course']='BCA'                #Updated entry
print('d[course]:',d['Course'])
```

Output:

```
d[Name]: Arav
d[course]: B.tech
d[roll_no]: 18/001
d[marks]: 99
d[course]: BCA
```

3.6.3 Deleting

- We can delete one or more items using the del keyword.
- To delete or remove all the items in just one statement, use the **clear ()** function.
- Finally, to remove an entire dictionary from the memory, we can gain use the del statement as **del Dict_name**.
- The syntax to use the del statement can be given as,

del dictionary_variable[key]

Example:

```
Dict = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}
print("Name is : ", Dict.pop('Name'))    # returns Name)
print("Dictionary after popping Name is : ", Dict)
print("Marks is :", Dict.pop('Marks', -1))   # returns default value
print("Dictionary after popping Marks is : ", Dict)
print("Randomly popping any item : ",Dict.popitem())
print("Dictionary after random popping is : ", Dict)
print("Aggregate is :", Dict.pop('Aggr'))    # generates error
print("Dictionary after popping Aggregate is : ", Dict)
```

```
OUTPUT
Name is :  Arav
Dictionary after popping Name is :  {'Course': 'BTech', 'Roll_No': '16/001'}
Marks is : -1
Dictionary after popping Marks is :  {'Course': 'BTech', 'Roll_No': '16/001'}
Randomly popping any item :  ('Course', 'BTech')
Dictionary after random popping is :  {'Roll_No': '16/001'}
Traceback (most recent call last):
  File "C:\Python34\Try.py", line 8, in <module>
    print("Aggregate is :", Dict.pop('Aggr'))
KeyError: 'Aggr'
```

3.6.4 Sorting Items

- The **keys()** method of dictionary returns a list of all the keys used in the dictionary in a arbitrary order.
- The **sorted()** function is used to sort the keys as shown below:

Example:

```
d={'roll_no':653,'name':'python','course':'b.tech'}
print(sorted(d.keys()))
```

Output:

['course', 'name', 'roll_no']

3.6.5 Looping Over Dictionaries

- We can loop over a dictionary to access only values, only keys, and both using the **for loop** as shown in the code given below:

Example

```
d={'roll_no':653,'name':'python','course':'b.tech'}
print("using KEYS:",end=' ')
for key in d:
        print(key,d[key],end=';')
print("\nVALUES only:",end=' ')
for val in d.values():
        print(val,end=' ')
print("\n Dictionary items:",end=' ')
for key,val in d.items():
        print(key,val,end=';')
```

Output:

using KEYS: course b.tech;name python;roll_no 653;
VALUES only: b.tech python 653
Dictionary items: course b.tech;name python;roll_no 653;

3.6.6 Built-in Methods

Operation	Description	Example	Output
`len(Dict)`	Returns the length of dictionary. That is, number of items (key-value pairs)	`Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}` `print(len(Dict1))`	`3`
`str(Dict)`	Returns a string representation of the dictionary	`Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}` `print(str(Dict1))`	`{'Name': 'Arav', 'Roll_No': '16/001', 'Course': 'BTech'}`

`Dict.clear()`	Deletes all entries in the dictionary	`Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}`
		`Dict1.clear()`
		`print(Dict1)`
			`{}`

`Dict.copy()`	Returns a shallow copy of the dictionary, i.e., the dictionary returned will not have a duplicate copy of `Dict` but will have the same reference	`Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}`
		`Dict2 = Dict1.copy()`
		`print("Dict2 : ", Dict2)`
		`Dict2['Name'] = 'Saesha'`
		`print("Dict1 after modification : ", Dict1)`
			`Dict2 : {'Course': 'BTech', 'Name': 'Arav', 'Roll_No': '16/001'}`
			`Dict1 after modification: {'Course': 'BTech',`

		`print("Dict2 after modification : ",Dict2)`	`'Name': 'Arav', 'Roll_No': '16/001'}` `Dict2 after modification: {'Course': 'BTech', 'Name': 'Saesha', 'Roll_No': '16/001'}`

`Dict.fromkeys(seq[,val])`	Create a new dictionary with keys from seq and values set to val. If no val is specified then, `None` is assigned as default value	`Subjects = ['CSA', 'C++', 'DS', 'OS']`
		`Marks = dict.fromkeys(Subjects,-1)`
		`print(Marks)`
			`{'OS': -1, 'DS': -1, 'CSA': -1, 'C++': -1}`

`Dict.get(key)`	Returns the value for the key passed as argument. If the key is not present in dictionary, it will return the default value. If no default value is specified then it will return None	`Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}` `print( Dict1.get('Name'))`	`Arav`
`Dict.has_key(key)`	Returns `True` if the key is present in the dictionary and False otherwise	`Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}` `print('Marks' in Dict1)`	`False`
`Dict.items()`	Returns a list of tuples (key-value pair)	`Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}` `print(Dict1.items())`	`[('Course', 'BTech'), ('Name', 'Arav'), ('Roll_No', '16/001')]`
`Dict.keys()`	Returns a list of keys in the dictionary	`Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}` `print(Dict1.keys())`	`['Course', 'Name', 'Roll_No']`
`Dict.setdefault(key, value)`	Sets a default value for a key that is not present in the dictionary	`Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'}`	`Arav has got marks = 0`

		Dict1. setdefault('Marks',0) print(Dict1['Name'], "has got marks = ", Dict1. get('Marks'))	
Dict1.update(Dict2)	Adds the key-value pairs of Dict2 to the key-value pairs of Dict1	Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'} Dict2 = {'Marks' : 90, 'Grade' : 'O'} Dict1.update(Dict2) print(Dict1)	{'Grade':'O', 'Course': 'BTech', 'Name': 'Arav', 'Roll_No': '16/001', 'Marks': 90}
Dict.values()	Returns a list of values in dictionary	Dict1 = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'} print(Dict1.values())	['BTech', 'Arav', '16/001']
Dict.iteritems()	Used to iterate through items in the dictionary	Dict = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'} for i,j in Dict. iteritems(): print(i, j)	Course BTech Name Arav Roll_No 16/001
in and not in	Checks whether a given key is present in dictionary or not	Dict = {'Roll_No' : '16/001', 'Name' : 'Arav', 'Course' : 'BTech'} print('Name' in Dict) print('Marks' in Dict)	True False

3.7 Sets:

- Set is a mutable and unordered collection of items represented using curly brackets { }.
- Set does not allow duplicate values.
- Since sets are unordered, indexing cannot be done.
- Like mathematical sets, python sets are also a powerful tool that have the ability to calculate union, differences and intersections between other sets.

3.7.1 Set Operations

- To find union of two sets S and T,
  S | T
- To find intersection of two sets S and T,
  S & T
- To find difference between sets S and T,
  S - T
- To find symmetric difference between sets S and T,
  S ^ T
- To find if two sets S and T are equivalent or not,
  S = = T or S != T
- To find if set S is subset of set T,
  S <= T
- To find if set S is superset of set T,
  S >= T

Example

```
>>> S = set([1,2,3,4,5])
>>> T = set([1,2,3,4,5,6,7,8,9,10])
>>> S
{1, 2, 3, 4, 5}
>>> T
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
>>> S | T
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
>>> S & T
{1, 2, 3, 4, 5}
>>> S - T
set()
>>> T - S
{6, 7, 8, 9, 10}
>>> S ^ T
{6, 7, 8, 9, 10}
>>> S == T
False
>>> S != T
True
>>> S <= T
True
>>> S >= T
False
```

3.8 List comprehensions

- List comprehensions are used to create lists in a concise way.
- This is mainly beneficial to make new lists where each element is obtained by applying some operations to each member of another sequence or iterable.
- It is also used to create a sub sequence of those elements that satisfy a certain condition.

Syntax

List = [expression for variable in sequence]

Examples

1. Program to make a list of cubes

```
Cubes = []
for i in range(11):
        cubes.append(i**3)
```

These 3 lines of code can be combined into one as below:

```
Cubes = [i ** 3 for i in range(11)]
```

>>>[0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

2. Program to combine elements of two lists

```
Print([(x,y) for x in [10, 20, 30] for y in [30, 10, 40] if x != y])
```

>>>[(10,30),(10,40),(20,30),(20,10),(20,40),(30,10),(30,40)]

3.9 Dictionary Comprehensions

- Dictionary comprehension is a method for transforming one dictionary into another dictionary.
- During this transformation, items within the original dictionary can be conditionally included in the new dictionary and each item can be transformed as needed.

Syntax

Dict = {key:value for (key,value) in dictionary.items()}

Examples

```
    dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
    # Double each value in the dictionary
    double_dict1 = {k:v*2 for (k,v) in dict1.items()}
    print(double_dict1)
>>> {'e': 10, 'a': 2, 'c': 6, 'b': 4, 'd': 8}
    dict1_keys = {k*2:v for (k,v) in dict1.items()}
    print(dict1_keys)
>>> {'dd': 4, 'ee': 5, 'aa': 1, 'bb': 2, 'cc': 3}
```

Assignment cum Tutorial Questions

I. Objective Questions

1) In Python a string is appended to another string by using which operator? [ ]

   a) + b)* c)[ ] d)+=

2) Which error is generated when a character in a string variable is modified? [ ]

   a) IndexError b) NameError c) TypeError d)BoundError

3) "Cool" become "COOL", which two functions must have been applied? [ ]

   a) strip() and upper() b) strip() and lower()

   c) strip() and capitalize() d) lstrip() and rstrip()

4) Find the error in following Python code. [ ]

```
str = "Hello world"
str[6] = 'w'
print(str)
```

   a) Hello world c) in line 2 use double quotes

   b) 'str' object does not support item assignment d) Hello wworld

5) If List=[1,2,3,4,5] and rewrite List[3]=List[1], then what will be the List[3] [ ]

   a) 1 (b) 3 (c) 2 (d) 4

6) print len((1,2,3,4,5,6)) is [ ]

   a) 5 (b) 6 (c) 21 (d) 7

7) tuple=('abcd',23,2.4,1)

   print tuple[:3]

   What is the output? [ ]

   a) ('abcd',23,2.4) b) (1) c) (23,2.4,1) d) ('abcd',23,2.4,1)

8) what is the output of print tuple[2:] if tuple=('abcd',786,2.23,1,2) [ ]

   a) (cd,786,2.23,1,2) b) (2.23,1,2) c) (786,2.23,1,2) d) (1,2)

9) Suppose t = (1, 2, 4, 3), which of the following is incorrect? [ ]

   a) print(t[3]) b) t[3] = 45

   c) print(max(t)) d) print(len(t))

10) What is the output of the program: [ ]

```
for fruit in ['apple','banana','mango']:
    print("I like",fruit)
```

   a) [apple','banana','mango'] b) I like 'apple' c) I like apple d) I like

I like ‘banana’ I like banana I like

I like ‘mango’ I like mango I like

11) What is the output of the program [ ]

```
my_list = ['p','r','o','b','l','e','m']
print('p' in my_list)
print('a' in my_list)
print('c' not in my_list)
```

a) True b)True c)False d) False

False True True True

True False False True

12) What is the output of the program [ ]

```
my_tuple = ('p','e','r','m','i','t')
print(my_tuple[-1])
print(my_tuple[-6])
```

a) t b) t c) p d) t

p t p NULL

13) What is the output of the program [ ]

```
my_tuple = ('p','r','o','g','r','a','m','i','z')
print(my_tuple[1:4])
print(my_tuple[:-7])
print(my_tuple[7:])
print(my_tuple[:])
```

a) ('r', 'o', 'g')
('p', 'r')
('i', 'z')
('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

b) (‘p’,’r’,’o’)
(‘r’,’p’)
(‘z’,’’i’)
('p','r','o','g','r','a','m','i','z')

c)('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
('r', 'o', 'g')
('p', 'r')
('i', 'z')

d) ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
('i', 'z')
('p', 'r')
('r', 'o', 'g')

14) What is the output of the program [ ]

```
print((1, 2, 3) + (4, 5, 6))
print(("Repeat",) * 3)
```

a) (1, 2, 3, 4, 5, 6)
('Repeat', 'Repeat', 'Repeat')

b) (‘Repeat’,Repeat’,’Repeat’)
(1,2,3,4,5,6)

c)(1,2,3)+(4,5,6)
“Repeat”

d) (“Repeat”,)*3
(1,2,3)+(4,5,6)

15) What is the output of the program [ ]

```
my_tuple = ('a','p','p','l','e',)
print(my_tuple.count('p'))
print(my_tuple.index('l'))
```

a) 2
3

b) 2
2

c) 3
2

d) 3
3

16) What is the output of the program [ ]

```
pow2 = [2 ** x for x in range(10)]
print(pow2)
```

a) [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
b) [512,256,128,64,32,16,8,4,2,1]
c) [1,2,3,4,5,6,7,8,9,10]
d) [1,3,5,7,9]

17) What is the output of the program [ ]

```
my_list = ['p','r','o','b','e']
print(my_list[-1])
print(my_list[-5])
```

a) e
NULL

b) e
p

c) p
e

d) e
e

18) What is the output of the program [ ]

```
odd=[1,3,5]
Print(odd+[9,7,5])
Print([“re”]*3)
```

a) [1,3,5,9,7,5]

b) [1,3,5,9,7]

c) (odd+[9,7,5])

d) [1,3,5]

[“re”,”re”,”re”] [“re”,”re”,”re”] ([“re”]*3) ([“re”]*3)

19) What is the output of the program [ ]

```
odd = [1, 9]
odd.insert(1,3)
print(odd)
odd[2:2] = [5, 7]
print(odd)
```

a) [1, 3, 9] b) [1,3,5,7,9] c) [1,9,3] d) [1,9,1,3]
[1, 3, 5, 7, 9] [1,3,5,7,9] [1,9,3,5,7] [1,9,1,3,5,7]

20) What is the output of the following code?

```
A = {1:"A",2:"B",3:"C"}
for i,j in a.items():
    print(i,j,end=" ")
```

a) 1 A 2 B 3 C b) 1 2 3 c) A B C d) 1:”A” 2:”B” 3:”C”

21) Suppose d = {“john”:40, “peter”:45}, to delete the entry for “john” what command do we use [ ]

a) d.delete(“john”:40) b) d.delete(“john”) c) del d[“john”] d) del d(“john”:40)

22) Suppose d = {“john”:40, “peter”:45}, what happens when we try to retrieve a value using the expression d[“susan”]? [ ]

a) Since “susan” is not a value in the dictionary, Python raises a KeyError exception
b) It is executed fine and no exception is raised, and it returns None
c) Since “susan” is not a key in the dictionary, Python raises a KeyError exception
d) Since “susan” is not a key in the set, Python raises a syntax error

23) What gets printed?

```
foo = {1:'1', 2:'2', 3:'3'}
del foo[1]
foo[1] = '10'
del foo[2]
print(len(foo))
```

a) 1 b) 2 c) 3 d) 4 e) An Exception is thrown

24) If Dict = {1:2, 3:4, 4:11, 5:6, 7:8}, then **print(Dict[Dict[3]])** will print ?

a) 2 b) 8 c) 11 d) 6

25) Which Data type does not support indexing?

a) List b) Tuple c) Dictionary d) Set

II. Descriptive Questions

1) With the help of an example, explain how we can create string variables in Python. **[BL:2]**
2) Write any 5 Built-in string methods and functions usage and example. **[BL:2]**
3) Analyze different ways to manipulate strings in python. **[BL:3]**
4) What is negative index in list and tuple? **[BL:2]**
5) What is tuple? What are the different operations performed on tuple? Explain with an example? **[BL:2]**
6) Illustrate the ways of creating the tuple and the tuple assignment with suitable programs. **[BL:2]**
7) What is tuple? How to change and delete a tuple in python. Explain with examples. **[BL:2]**
8) Summarize basic List operations with examples. **[BL:2]**
9) How can you access and update values in a list? / Explain mutability of lists? **[BL:2]**
10) Explain list slicing and list mutability with suitable programming examples. **[BL:2]**
11) "Tuples are immutable". Explain with examples. **[BL:2]**
12) Explain the importance of Dictionary data type in python? **[BL:2]**
13) List-out various operations can be performed on Dictionary Data type? **[BL:2]**
14) List-out the Built-in functions and methods of Dictionary Data type in python? **[BL:2]**
15) How to delete items from a dictionary? Explain with an example. **[BL:2]**
16) Illustrate how dictionaries are created, accessed and modified with suitable program examples. **[BL:2]**
17) Differentiate dictionary data structure with list. **[BL:2]**
18) Illustrate looping through a dictionary with suitable programming examples (looping through all key-value pairs, only keys, only values and looping through a dictionary in order). **[BL:2]**
19) Define Python Dictionary. What happens if we try to update a key which does not exist in the dictionary? **[BL:3]**
20) Explain how to create a set and access its elements. **[BL:2]**

21) List various operations that can be performed on sets. **[BL:2]**

22) Write a set of commands that covers at least five tuple functions and five list functions? **[BL:2]**

III. Programs **[BL:3]**

1) Write a python program that uses function to reverse the given string without using slicing.
2) Write a python program that accepts a string from a user and re-displays the same after removing vowels from it.
3) Write a python program to remove the ith Occurrence of the given word in a list where words can repeat?
4) Write a Python program to multiply two matrices using list.
5) Write a program to find sum of all even numbers in a list?
6) Write a program that reverses a list using a loop?
7) Write a program to find whether a particular element is present in the list?
8) Write a program that finds the sum of all the numbers in a list using a while loop?
9) Write a program that forms a List of first character of every word present in another List.
10) Write a program that creates a list[‘a’,’b’,’c’], then create a tuple from that list.
11) Write a program that converts a list of characters into their corresponding ASCII values using map( ) function.
12) Write a code snippet in Python to Access Elements of a Tuple.
13) Write code snippets in Python for modifying and deleting Elements of Tuple.
14) Write a Python script to sort (ascending and descending) a dictionary by value.
15) Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form (x, x*x).

    Sample:

    Dictionary (n = 5):

    Expected Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

16) Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.

    Sample Dictionary:

    {1: 1, 2: 5, 3: 9, 4: 15, 5: 25, 6: 36, 7: 49, 80: 64, 9: 80, 90: 100, 11: 121, 12: 144, 13: 169, 14: 200, 105: 225}

17) Write a Python program to map two lists into a dictionary.

18) Write a python program to check if all dictionaries in a list are empty or not? (Nov-2018)

19) Write a Python program to combine two dictionary adding values for common keys.

d1 = {'a': 100, 'b': 200, 'c':300}

d2 = {'a': 300, 'b': 200,'d':400}

Sample output: {'a': 400, 'b': 400,'d': 400, 'c': 300}

20) Write a Python program to create and display all combinations of letters, selecting each letter from a different key in a dictionary

Sample data: {'1':['a','b'], '2':['c','d']}

Expected Output:

ac

ad

bc

bd

21) Write a Python program to get the top three items in a shop.

Sample data: {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}

Expected Output:

item4: 55

item1: 45.5

item3: 41.3

22) Define a function which can print a dictionary where the keys are numbers between 1 and n (both included) and the values are the cubes of keys.

23) Write a python program that uses dictionary to return the name of the employee of an organization when the project name is given.

24) Write a Python program to access dictionary key's element by index.

25) Write a python program to remove key values pairs from a list of dictionaries.

26) Given the following dictionary:

```
inventory = {
'gold' : 500,
'pouch' : ['flint', 'twine', 'gemstone'],
'backpack' : ['xylophone','dagger', 'bedroll','bread loaf']
```

}

Try to do the followings:

i) Add a key to inventory called 'pocket'.

ii) Set the value of 'pocket' to be a list consisting of the strings 'seashell', „strange berry“, and 'lint'.

iii) sort the items in the list stored under the 'backpack' key.

27) Write a Python program to check if two given sets have no elements in common.

28) Write a Python program to remove the intersection of a 2nd set from the 1st set.

29) Write a Python program to remove an item from a set if it is present in the set.