


UNIT-II

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1




Python Data Types

Built-in Data Types

- In programming, data type is an important concept.
- Variables can store data of different types, and different types can do different things.
- Python has the following data types built-in by default, in these categories:

Data Type	Keyword used in Python
Text Type	str
Numeric Type	int, float, complex
Sequence Type	list, tuple, range
Mapping Type	Dict
Set Type	set, frozenset
Boolean Type	bool
Binary Types	bytes, bytearray, memoryview

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python String

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'Powerful' is same as "Powerful"

You can display a string literal with the `print()` function.

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

```
a = "Hello" or a = 'Hello'
```


You can assign a multiline string to a variable by using three quotes:

```
a = """ Python was designed for readability, and has some similarities
to the English language with influence from mathematics. """
```

Or

```
a = ''' Python was designed for readability, and has some similarities
to the English language with influence from mathematics. '''
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python String continued...


Strings are Arrays

- Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.
- However, Python does not have a character data type, a single character is simply a string with a length of 1.
- Square brackets can be used to access elements of the string.
- String data types are immutable. Which means a string value cannot be updated.

Get the character at position 1 (remember that the first character has the position 0):

```
a = "Hello, World!"
print(a[1])
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1




Python String continued...

String functions or operations on strings:

- Looping Through a String
for x in "BVICAM":
print(x)
- String Length – len()
- Check String – returns boolean
txt = "The best things in life are free!"
print("free" in txt)
- Check if NOT - returns boolean
txt = "The best things in life are free!"
print("expensive" not in txt)
- Slicing
b = "Hello, World!"
print(b[2:5])


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python String continued...

Upper Case – upper()
Lower Case – lower()
Remove whitespace – strip(), lstrip(), rstrip()
Replace String – str.replace(old_str, new_str)
Split String – str.split() , str.split('-')
String Concatenation – plus(+) operator
Strings Appending – str1 +=str2
Deleting string – del str
count(str,beg,end)
find(str,beg,end)
index(str,beg,end)

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python String continued...


String Format - we can combine strings and numbers by using the format method.

1. **The format() method** takes the passed arguments, formats them, and places them in the string where the placeholders {} are:


```
code= 106
txt = "Paper Code of Python Programming is MCA - {}"
print(txt.format(code))
```
2. **The format operator %** allows users to construct strings, replacing parts of strings with the data stored in variables. The syntax for the string formatting operation is:


```
print('Course_name=%s and Course_code= %d' % (cname,ccode))
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Slice Operation

A substring of a string is called a **Slice**.

We can take a subset of a string from the original string by using [] operator also known as **slicing operator**.


The syntax of slice operation in s[start : end : stride]

Stride refers to the number of characters to move forward after the first character is retrieved from the string. The default value of stride is 1.

We can specify a negative value for the third argument. This is useful to print the original string in reverse order.

```
print(str[::-1])
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1




Slice Operation

Try this:

```
Str='PYTHON'
```

```
print(str[-1])
print(str[-6])
print(str[-2:])
print(str[: -2])
print(str[-5 : -2])
print(str[: : -3])
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python String continued...

ord() Function:

This function returns the ASCII code of the character


```
ch='R'
print(ord(ch))
```

chr() Function:

This functions returns character represented by ASCII number.

```
print(chr(82))
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



String Comparison

This is done using the following operators:

==: This checks whether two strings are equal

!=: This checks if two strings are not equal

<: This checks if the string on its left is smaller than that on its right


<=: This checks if the string on its left is smaller than or equal to that on its right

>: This checks if the string on its left is greater than that on its right

>=: This checks if the string on its left is greater than or equal to that on its right

- String comparison in Python takes place character by character. That is, characters in the same positions are compared from both the strings.
- If the characters fulfill the given comparison condition, it moves to the characters in the next position. Otherwise, it merely returns False.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Iterating String


```
string_name = "Python String"

for element in string_name:
    print(element, end=' ')
print("\n")

for element in range(0, len(string_name)):
    print(string_name[element])

Index=0
While index < len(string_name)
    letter= string_name[index]
    print(letter)
    index+=1
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1




Sequence Type

Sequences allow you to store multiple values in an organized and efficient manner. There are several **sequence types**: strings, lists and tuples. Two most popular sequence types are **lists** and **Tuples**.

- Each element of a sequence is assigned a number - its position or index. The first index is zero, the second index is one, and so forth.
- Operations that can be performed on sequence types include indexing, slicing, adding, multiplying, and checking for membership.
- In addition, Python has built-in functions for finding the length of a sequence and for finding its largest and smallest elements.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1




Python Lists

Python lists
The list is written as a list of comma-separated values (items) between square brackets.
Important thing about a list is that items in a list need not be of the same type.
Creating a list is as simple as putting different comma-separated values between square brackets

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7, 8]
list3 = ["a", "b", "c", "d"]
print "list1[0]: ", list1[0]
print "list2[1:5]: ", list2[1:5]
list1[2] = 2021
del list1[2]
```


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Lists continued..

- List items are **ordered**, **changeable**, and allow **duplicate** values.
- When we say that lists are ordered, it means that the items have a defined order, and that order will not change. If you add new items to a list, the new items will be placed at the end of the list.
- The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.
- Since lists are indexed, lists can have items with the same value
- To determine how many items a list has, use the len() function
print(len(list1))
- To remove a list element, you can use either the **del** statement if you know exactly which element(s) you are deleting or the **remove()** method if you do not know.
del list1[2]
list1.remove('chemistry')


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Basic List Operations

1. Len
2. Concatenation using +
3. Repetitions using *
4. In
5. Not in
6. Max
7. Min
8. Sum
9. All
10. Any
11. List
12. sorted


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Lists Methods

Sr.No.	Methods with Description
1	list.append(obj) Appends object obj to list
2	list.count(obj) Returns count of how many times obj occurs in list
3	list.extend(seq) Appends the contents of seq to list
4	list.index(obj) Returns the lowest index in list that obj appears
5	list.insert(index, obj) Inserts object obj into list at offset index
6	list.pop([index]) Removes and returns last object or obj from list
7	list.remove(obj) Removes object obj from list
8	list.reverse() Reverses objects of list in place
9	list.sort() Sorts objects of list
10	list.index(element, start, end) returns the index of the specified element in the list.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Some Key Points

Insert(), remove() and sort() methods only modify the list and do not return any value. If you print the return value of these methods, you will get **None**.


The sort() method uses ASCII values to sort the values in the list.

Items in a list can also be deleted by assigning an empty list to a slice of elements:

```
list1 = ['a','e','m','n','p','q','i','o','u']
list1[2:6] = []
print(list1) .....???
```

When one list is assigned to another list using the assignment operator(=), then a new copy of list is NOT made. Instead, assignment makes the two variables point to the same list in memory. This is known as **aliasing**.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Cloning a list in Python


There are various ways of copying or cloning a list in Python. These various ways of copying takes different execution time, so we can compare them on the basis of time.

- Using slicing technique**
 This is the easiest and the fastest way to clone a list. This method is considered when we want to modify a list and also keep a copy of the original. This process is also called cloning. This technique takes about 0.039 seconds and is the fastest technique.


```
list2=list1[:]
```
- Using the extend() method**
 The lists can be copied into a new list by using the extend() function. This appends each element of the iterable object (e.g., another list) to the end of the new list. This takes around 0.053 second to complete.


```
list2.extend(list1)
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Cloning a list in Python


- Using the list() method**
 This is the simplest method of cloning a list by using the builtin function list(). This takes about 0.075 seconds to complete.


```
list2 = list(list1)
```
- Using the copy() method**
 The inbuilt method copy is used to copy all the elements from one list to another. This takes around 1.488 seconds to complete.


```
list2= list1.copy()
```
- Using list comprehension**
 The method of list comprehension can be used to copy all the elements individually from one list to another. This takes around 0.217 seconds to complete.


```
list2 = [i for i in list1]
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1




Cloning a list in Python

- Using the append() method**
 This can be used for appending and adding elements to list or copying them to a new list. It is used to add elements to the last position of list. This takes around 0.325 seconds to complete and is the slowest method of cloning.


```
list2=[]
for item in list1: list2.append(item)
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Tuples


Python Tuples

- A tuple is a collection of objects which ordered and immutable.
- Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.
- Creating a tuple is as simple as putting different comma-separated values.

```
tup1= ('physics', 'chemistry', 1997, 2000 )
tup2 = (1, 2, 3, 4, 5, 6,7,8 )
tup3 = "a", "b", "c", "d"
tup4 =(50,)
print "tup1[0]: ", tup1[0]
print "tup2[1:5]: ", tup2[1:5]
```

- Please note, **Tuples are immutable** which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Tuples

Tuple Assignment, Packing, and Unpacking

A literal tuple containing several items can be assigned to a single object:

```
tup1= ('physics', 'chemistry', 1997, 2000 )
```

When this occurs, it is as though the items in the tuple have been "packed" into the object.

If that "packed" object is subsequently assigned to a new tuple, the individual items are "unpacked" into the objects in the tuple:


```
(s1,s2,s3,s4) = tup1
```

When unpacking, the number of variables on the left must match the number of values in the tuple

Create Tuple With One Item

- To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Tuples

- Advantages of Tuple over List**
- Since tuples are quite similar to lists, both of them are used in similar situations. However, there are certain advantages of implementing a tuple over a list. Below listed are some of the main advantages:
 - We generally use tuples for heterogeneous (different) data types and lists for homogeneous (similar) data types.
 - Since tuples are immutable, iterating through a tuple is faster than with list. So there is a slight performance boost.
 - Tuples that contain immutable elements can be used as a key for a dictionary. With lists, this is not possible.
 - If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1




Python Mapping Type

Python Dictionary

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and does not allow duplicates.
- Dictionaries are written with curly brackets, and have keys and values:
- Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces.
- Keys are unique within a dictionary while values may not be.
- Dictionary keys are case-sensitive.
- The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

```
dict = {
    'Name': 'Arjun',
    'Age': 21,
    'Course': 'MCA'}
Print(dict["Name"])
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 26



Creating Python Dictionary

Creating a Dictionary:

- 1. The syntax to create an empty dictionary is:**


```
Dictionary_variable = { }
```
- 2. The syntax to create a dictionary with key-value pair is:**

```
Dictionary_variable = { key1:val1, key2:val2,....}
```
- 3. To create a dictionary with one or more key-value pairs, we can use dict() function.**
 The dict() function creates a dictionary from a sequence of key-value pairs:

```
Print(dict([('Roll_No', '00411604421'), ('Name', 'Aayush'), ('Course', 'MCA')]))
```


 The output will be?

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 26



Creating Python Dictionary

- 4. Dictionary comprehension is another way of creating a dictionary.** It is a syntactic construct which creates a dictionary based on existing sequence:


```
D = {variable: expression for variable in sequence [ if condition]}
```

 It has three parts – for loop, condition and expression
 The expression generates elements(values) of dictionary from items in the sequence that satisfy the condition

```
dict1 = { x : 2*x for x in range(1,10)}
print(dict1)
```


 The output will be?

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 27



Python Dictionary continued...

Accessing Items

```
x = dict1.get("Name")
print(dict1["Name"])
```

If you try to access an item with a key, which is not specified in the dictionary, a `KeyError` is generated.

Change Values


```
dict1["Name"] = 'Aayush Khurana'
dict1.update({"Course": "MBA"})
```

Adding Items

```
dict1["Marks"] = 91
dict1.update({"City": "New Delhi"})
```

***Dictionary is an associative array** also known as hashes since any key of the dictionary can be associated or mapped to a value.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 28



Python Dictionary continued...

Removing Items

```
del dict1["City"]
del dict1
```

To delete or remove all items in just one statement, use `clear()` function.

```
dict1.clear()
```

We can use `pop()` to delete a particular key from the dictionary:

```
dict1.pop("Marks")
```


Loop through a Dictionary:

- Print all key names in the dictionary, one by one:


```
for x in dict:
    print(x)
```
- Print all *values* in the dictionary, one by one:


```
for x in dict:
    print(dict[x])
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 29



Python Dictionary continued...

Copy a Dictionary


You cannot copy a dictionary simply by typing `dict1 = dict2`

- One way is to use the built-in Dictionary method `copy()`

```
dict1 = { 'Name': 'Arjun', 'Age': 21, 'Course': 'MCA' }
dict2 = dict1.copy()
print(dict2)
```
- Another way to make a copy is to use the built-in function `dict()`

```
dict1 = { 'Name': 'Arjun', 'Age': 21, 'Course': 'MCA' }
dict2 = dict(dict1)
print(dict2)
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 30




Python Dictionary continued...

Nested Dictionaries

A dictionary can contain dictionaries, this is called nested dictionaries.

```
My_Library = {
  "Book1": {
    "Title": "Fundamentals of IT",
    "Author": "Kapoor",
    "Publisher": "TMH",
    "Year": 2006
  },
  "Book2": {
    "Title": "Introduction to Python Programming",
    "Author": "Timothy A. Budd",
    "Publisher": "S.Chand",
    "Year": 2016
  },
  "Book3": {
    "Title": "PL/SQL Programming",
    "Author": "Ivan Bayros",
    "Publisher": "Pearson",
    "Year": 2010
  }
}
```


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 31



Python Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 32



Python Dictionary Methods

Sorting a Dictionary in Python


sorted() function

- The sorted() function can accept three parameters: the iterable, the key, and reverse. **sorted(iterable, key)**

```
sales = {'apple':10, 'banana':12, 'orange':6, 'grapes':100}
print(sorted(sales))

print(sorted(sales,key=sales.get))
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 33



Python Dictionary Methods


Dictionaries and lists share the following characteristics:

- Both are mutable.
- Both are dynamic. They can grow and shrink as needed.
- Both can be nested. A list can contain another list. A dictionary can contain another dictionary. A dictionary can also contain a list, and vice versa.

Dictionaries differ from lists primarily in how elements are accessed:

- List elements are accessed by their position in the list, via indexing.
- Dictionary elements are accessed via keys.


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 34



Python Sets

- Sets are used to store multiple items in a single variable.
- 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.
- A set is a collection which is both *unordered* and *unindexed*.
- The items in a set do not have a defined order. Set items can appear in a different order every time you use them, and cannot be referred to by index or key.
- Every set element is unique (no duplicates) and must be immutable (cannot be changed). However, a set itself is mutable. We can add or remove items from it.
- Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc.
- The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set. This is based on a data structure known as a hash table.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 35



Python Sets continued ...

Creating Python Sets


A set is created by placing all the items (elements) inside curly braces {}, separated by comma, or by using the built-in set() function.

It can have any number of items and they may be of different types (integer, float, tuple, string etc.). But a set cannot have mutable elements like lists, sets or dictionaries as its elements.

```
my_set={1,2,3}
print(my_set)

my_set={1.0, "Hello", (1,2,3)}
print(my_set)
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 36



Python Sets continued ...


Creating an empty set is a bit tricky. Empty curly braces { } will make an empty dictionary in Python. To make a set without any elements, we use the set() function without any argument.

```
my_set=set()
```

Modifying a set in Python

- Sets are mutable. However, since they are unordered, indexing has no meaning.
- We cannot access or change an element of a set using indexing or slicing. Set data type does not support it.
- We can add a single element using the add() method, and multiple elements using the update() method. The update() method can take tuples, lists, strings or other sets as its argument. In all cases, duplicates are avoided.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Sets continued ...

Modifying a set in Python


```
my_set={1,3}
print(my_set)

my_set.add(2)
print(my_set)

my_set.update([2,3,4,5,6])
print(my_set)

my_set.update([2,3],[7,8,9])
print(my_set)
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Sets continued ...


Removing elements from a set

- A particular item can be removed from a set using the methods **discard()** and **remove()** method
- The only difference between the two is that the **discard()** function leaves a set unchanged if the element is not present in the set. On the other hand, the **remove()** function will raise an error in such a condition (if element is not present in the set).

```
my_set={1,2,3,5,6}
print(my_set)

my_set.discard(4)
print(my_set)
my_set.remove(4)
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1




Python Sets continued ...

Python Set Operations

- Sets can be used to carry out mathematical set operations like union, intersection, difference and symmetric difference. We can do this with operators or methods.
- Let us consider the following two sets for the following operations.
 $A = \{1, 2, 3, 4, 5\}$
 $B = \{4, 5, 6, 7, 8\}$

Set Union - `print(A | B)`
Set Intersection - `print(A & B)`
Set Difference - `print(A - B)`
Set Symmetric Difference - `print(A ^ B)`
Set Membership Test - `my_set=set("apple")`
`print('a' in my_set)`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Sets continued ...

Set Membership Test


```
my_set=set("apple")
print('a' in my_set)
print('A' in my_set)
print('T' not in my_set)
```

Iterating Through a Set

```
for letter in my_set:
    print(letter)
```

```
for n in A:
    print(n)
```


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



File Handling in Python

- Files are named locations on disk to store related information. They are used to permanently store data in a non-volatile memory (e.g. hard disk).
- Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files.
- Python treats file differently as text or binary and this is important.
- Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma (,) or newline character. It ends the current line and tells the interpreter a new one has begun.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1




File Handling in Python

File Path:
Most of the file systems used today stores files in a tree structure. At the top of the tree, is one root node. Under the root node, there are other files and folders.

Every file is identified by its path, that begins from the root node or root folder. The file path is also known as **pathname**.

1. Relative path
2. Absolute path

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 43



File Handling in Python


Types of File

1. **Text files** – A stream of characters that can be sequentially processed by a computer in forwards direction.

In a text file, each line of data ends with a newline character. Each file ends with a special character called the end-of-file(EOF) marker.
2. **Binary files** – It may contain any type of data, encoded in binary form. It includes file such as images, videos, zip files and other executable programs.

Like text file, binary file also ends with an end-of-file(EOF) marker.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 44



File Handling in Python

In Python, a file operation takes place in the following order:


- Open a file
- Read or write (perform operation)
- Close the file

We use **open ()** function in Python to open a file in read or write mode. `open ()` will return a file object.

open() function accepts two arguments, file name and the mode, whether to read or write. There are three kinds of mode, that Python provides and how files can be opened:

- "r", for reading.
- "w", for writing.
- "a", for appending.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 45



File Handling in Python


We must keep in mind that the mode argument is not mandatory. If not passed, then Python will assume it to be "r" by default.

```
f= open("test.txt")      # opens file in current directory
f= open("C:/python37/test.txt") # specifying full path
```

- The default is reading in text mode. In this mode, we get strings when reading from the file.
- On the other hand, binary mode "b" returns bytes and this is the mode to be used when dealing with non-text files like images or executable files.

```
f= open("img.bmp", 'rb+')      # read and write in binary mode
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



File Handling in Python

Working of read() mode


```
f= open("D:/Python Programming/Scripts/test.txt", mode='r')
print(f.read())
print(f.read(5))
```

Creating a file using write() mode

```
f= open("D:/Python Programming/Scripts/test.txt", mode='w')
f.write("Testing write operation\n")
f.write("This command will add this line in file")
f.close()
```

- The close() command terminates all the resources in use and frees the system of this particular program.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



File Handling in Python


Working of append() mode

```
f= open("D:/Python Programming/Scripts/test.txt", mode='a')
f.write("This command will add this line in file")
```

Using read along with with() function

```
with open("D:/Python Programming/Scripts/test.txt") as file:
    data= file.read()
print(data)
type(data)
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



File Built-in Methods


Python provides following built-in functions, those are

- `close()`
- `read()`
- `readline()`
- `write()`
- `writelines()`
- `tell()`
- `seek(offset [, from])`

Python Supports following built-in attributes, those are

- `file.name` - returns the name of the file which is already opened.
- `file.mode` - returns the access mode of opened file.
- `file.closed` - returns true, if the file closed, otherwise false.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 48



File Handling in Python

split() using file handling


We can also split lines using file handling in Python. This splits the variable when space is encountered. You can also split using any characters as we wish.

```
with open("D:/Python Programming/Scripts/test.txt") as file:
    data= file.readlines()
    for line in data:
        word=line.split()
        print(word)
```

Deleting a file

```
import os
os.remove("D:/Python Programming/Scripts/test.txt")
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 50



Python Directory Management


Directories are a way of storing, organizing, and separating the files on a computer.

The directory that does not have a parent is called a **root directory**. The way to reach the file is called the **path**.

The path contains a combination of directory names, folder names separated by slashes and colon and this gives the route to a file in the system.

Python contains several modules that has a number of built-in functions to manipulate and process data. Python has also provided modules that help us to interact with the operating system and the files.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1 51



Python Directory Management continues....


os and os.path Module

- The os module is used to handle files and directories in various ways. It provides provisions to create/rename/delete directories. This allows even to know the current working directory and change it to another. It also allows one to copy files from one directory to another. The major methods used for directory management is explained below.

Creating new directory:

- os.mkdir(name)** method to create a new directory.
- The desired name for the new directory is passed as the parameter.
- By default it creates the new directory in the current working directory.
- If the new directory has to be created somewhere else then that path has to be specified and the path should contain forward slashes instead of backward ones.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Directory Management continues....


Getting Current Working Directory (CWD):

- os.getcwd()** can be used.
- It returns a string that represents the path of the current working directory.
- The method does not require any parameters to be passed.

Renaming a directory:

- os.rename()** method is used to rename the directory.
- The parameters passed are old_name followed by new_name.
- If a directory already exists with the new_name passed, OSError will be raised in case of both Unix and Windows.
- If a file already exists with the new_name, in Unix no error arises, the directory will be renamed. But in Windows the renaming won't happen and error will be raised.
- os.rename('old_name','dest_dir/new_name')** method works similar to **os.rename()** but it moves the renamed file to the specified destination directory(dest_dir).

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Directory Management continues....


Changing Current Working Directory (CWD):

- Every process in the computer system will have a directory associated with it, which is known as Current Working Directory(CWD).
- os.chdir()** method is used to change it.
- The parameter passed is the path/name of the desired directory to which one wish to shift.

Listing the files in a directory

- A directory may contain sub-directories and a number of files in it. To list them, **os.listdir()** method is used.
- It either takes no parameter or one parameter.
- If no parameter is passed, then the files and sub-directories of the CWD is listed.
- If files of any other directory other than the CWD is required to be listed, then that directory's name/path is passed as parameter.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1



Python Directory Management continues....

Removing a directory

- **os.rmdir()** method is used to remove/delete a directory.
- The parameter passed is the path to that directory.
- It deletes the directory if and only if it is empty, otherwise raises an **OSError**.

To check whether it is a directory:

- Given a directory name or Path, **os.path.isdir(path)** is used to validate whether the path is a valid directory or not.
- It returns boolean values only. Returns **true** if the given path is a valid directory otherwise **false**.

To get size of the directory:

- **os.path.getsize(path_name)** gives the size of the directory in bytes.
- **OSError** is raised if, invalid path is passed as parameter.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Saumya, Assistant Professor – Unit 1
