UNIT-I : COMPUTER SYSTEMS AND ORGANISATION

CHAPTER-1
REVISION OF THE BASICS OF PYTHON

TOPIC-1
Python Basics

Revision Notes

➢ **Python was created by Guido Van Rossum in late '1980s' (Released in 1991) at National Research Institute in the Netherland. Python got its name from a BBC comedy series – "Monty Python's Flying Circus". Python is based on two programming languages:**

(i) ABC language

(ii) Modula 3

Some qualities of Python based on the programming fundamentals are given below:

➢ **Interactive Mode:** Interactive mode, as the name suggests, allows us to work interactively. It executes the code by typing one command at a time in Python shell.

➢ **Script Mode:** In script mode, we type Python program in a file and then execute the content of the file.

➢ **Indentation:** Blocks of code are denoted by line indentation, which is rigidly enforced.

➢ **Comments:** A hash sign (#) that is not inside a string literal begins a single line comment. We can use triple quoted string for giving multiple-line comments.

➢ **Variables:** A variable in Python is defined through assignment. There is no concept of declaring a variable outside of that assignment. Value of variable can be manipulated during program run.

➢ **Dynamic Typing:** In Python, while the value that a variable points to has a type, the variable itself has no strict type in its definition.

➢ **Static Typing:** In static typing, a data type is attached with a variable when it is defined first and it is fixed.

➢ **Multiple Assignment:** Python allows you to assign a single value to several variables and multiple values to multiple variables simultaneously.

For example:
```
a = b = c = 1
a, b, c = 1, 2, "john"
```

➢ **Token:** The smallest individual unit in a program is known as a Token or a lexical unit.

➢ **Identifiers: An identifier is a name used to identify a variable, function, class, module, or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).**

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Value and value are two different identifiers in Python.

Here are identifiers naming convention for Python:

➢ Class names start with an uppercase letter and all other identifiers with a lowercase letter.
➢ Starting an identifier with a single leading underscore indicates by convention that the identifier is meant to be private.
➢ Starting an identifier with two leading underscores indicates a strongly private identifier.
➢ If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

➢ **Reserved Words(Keywords):** The following list shows the reserved words in Python v3.0 or later

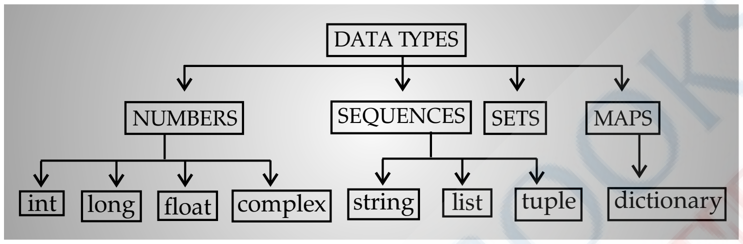
Python Keyword List

False	None	True	and	as	assert	async (v3.5 or later)
await (v3.5 or later)	break	class	continue	def	del	elif

else	except	finally	for	from	global	if
import	in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with	yield

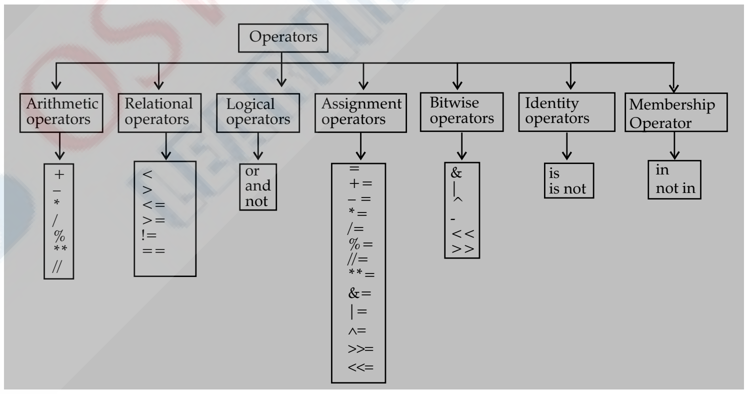
These reserved words should not be used as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only except False, None and True which have first letter capital.

- **Literal/Constant :** Literals (Often referred to as constant value) are data items that have a fixed value. Python allows several kind of literals as String literals, Numeric literals, Boolean literals, special literal None and literal Collections
- **Data Types :** Data type is a set of values and the allowable operations on those values. Python has a great set of useful data types. Python's data types are built in the core of the language. They are easy to use and straight forward.



- **Numbers** can be either integer or floating point numbers.
- A sequence is an ordered collection of items, indexed by integers starting from 0. Sequences can be grouped into strings, tuples and lists.
  - **Strings** are lines of text that can contain any character. They can be declared with double quotes.
  - **Lists** are used to group other data. They are similar to arrays.
  - **A tuple** consists of a number of values separated by commas.
- **A set** is an unordered collection with no **duplicate** items.
- **A dictionary** is an unordered set of key value pairs where the keys are unique.
- **Operator :** Operators are special symbols which perform some computation. Operators and operands form an expression. Python operators can be classified as given below :

Python Operators



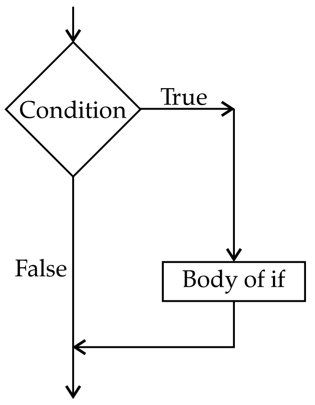
- **Expressions :** An expression in Python is any valid combination of operators, literals and variables.

TOPIC-2
Conditional Statements

Revision Notes

- **A conditional is a statement which is executed, on the basis of result of a condition.**
  - **if conditionals in Python have the following forms.**

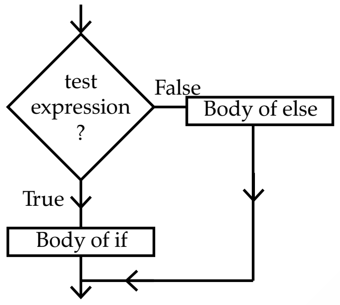
(A) Simple if



Syntax:

if < conditional expression>:
    [statements]

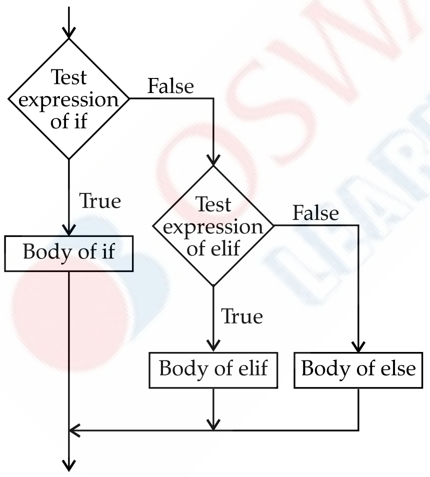
(B) The if-else conditional



Syntax:

if <conditional expression>:
    [statements]
else:
    [statements]

(C) The if-elif conditional statement



Syntax:
if <conditional expression>:
    Statement
    [statements]
elif <conditional expression>:
    statement
    [statements]
else:
    statement
    [statements]

(D) Nested if

- A nested if is an if which is inside another if's body or elif's body or else's body.
- Storing conditions – Complex and repetitive conditions can be named and then used in if statements.

TOPIC-3
Iteration Constructs

Revision Notes

➢ The iteration statements or repetition statements allow a set of instructions to be performed repeatedly.

➢ Python provides three types of loops

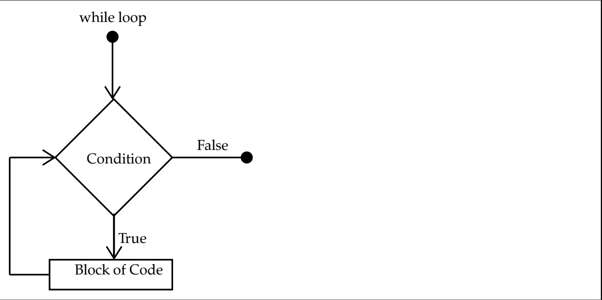
(A) Counting loops repeat a certain number of times e.g. for

(B) Conditional loops repeat until a certain condition is true e.g. while.

(C) Nested loops**.**

1. Python While Loop

A while loop in Python iterates till its condition becomes False. In other words, it executes the block of statements until the condition it takes is true.



Syntax

```
while <logical Expression>:
    loop body
```

When the program control reaches the while loop, the condition is checked. If the condition is true, the block of code under it is executed. After that, the condition is checked again. This continues until the condition becomes false. Then the first statement, if any after the loop is executed. Remember to indent all statements under the loop equally.

e.g.

```
>>> a=3
>>> while(a>0):
        print(a)
        a-=1
```

Output

```
3
2
1
```

(a) An Infinite Loop

Be careful while using a while loop. Because if you forget to increment or decrement the counter variable in Python, or write flawed logic, the condition may never become false. In such a case, the loop will run infinitely, and the conditions after the loop will starve. To stop execution, press Ctrl+C. However, an infinite loop may actually be useful.

(b) The else statement for while loop

A while loop may have an else statement after it. When the condition becomes false, the block under the else statement is executed. However, it doesn't execute if you break out of the loop or if an exception is raised.

e.g.

```
>>> a=3
>>> while(a>0):
         print(a)
         a-=1
       else:
        print("Reached 0")
```

Output

```
   3
   2
   1
   Reached 0
```

In the following code, we put a break statement in the body of the while loop for a==1. So, when that happens, the statement in the else block is not executed.

e.g.

```
>>> a=3
>>> while(a>0):
       print(a)
```

```
        a-=1
        if(a==1):
          break
        else:
          print("Reached 0")
```

Output

```
3
2
```

c. Single Statement while

Like an if statement, if we have only one statement in while loop's body, we can write it all in one line.

e.g.

```
>>> a=3
>>> while a>0: print(a); a-=1;
```

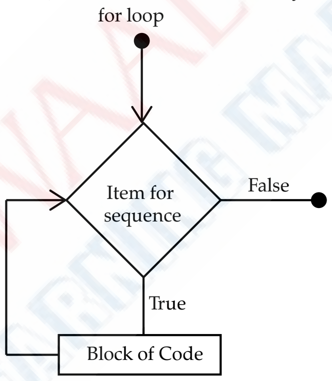
Output

```
3
2
1
```

You can see that there were two statements in while loop's body, but we used semicolons to separate them. Without the second statement, it would form an infinite loop.

2. Python FOR Loop

Python for loop can iterate over a sequence of items. The structure of a for loop in Python is different than that in C++ or Java. That is, for(int i=0;i<n;i++) won't work here. In Python, we use the 'in' keyword.



```
>>> for a in range(3):
        print(a)
```

Output

```
0
1
2
```

If we wanted to print 1 to 3, we could write the following code.

```
>>> for a in range(3):
        print(a+1)
```

Output

```
1
2
3
```

a. The range() function

This function yields a sequence of numbers. When called with one argument, say n, it creates a sequence of numbers from 0 to n-1.

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

We use the list function to convert the range object into a list object. Calling it with two arguments creates a sequence of numbers from first to second.

```
>>> list(range(2,7))
[2, 3, 4, 5, 6]
```

You can also pass three arguments. The third argument is the interval.

```
>>> list(range(2,12,2))
    [2, 4, 6, 8, 10]
```

Remember, the interval can also be negative.

```
>>> list(range(12,2,-2))
    [12, 10, 8, 6, 4]
```

3. **Nested Loops**

A loop may contain another loop in its body, this inner loop is called nested loop. But in a nested loop, the inner loop must terminate before the outer loop.

e.g.

```
for i in range(1,6):
    for j in range (1,i):
        print("*", end=" ")
    print()
```

- **Jump Statements** Python offers two jump statements-break and continue to be used within loops to jump out of loop iterations.
  - **break statement** It terminates the loop it lies within. It skips the rest of the loop and jumps over to the statement following the loop.
  - **continue statement** Unlike break statement, the continue statement forces the next iteration of the loop to take place, skipping any code in between.

TOPIC-4
Idea of Debugging

Revision Notes

- An error or a bug is anything in the code that prevents a program from compiling and running correctly.
- There are three types of errors

Compile Time errors occur at compile time.

These are of two types :

(i) Syntax errors occur when rules of programming language are misused.

(ii) Semantics errors occur when statements are not meaningful.

Run Time errors occur during the execution of a program.

Logical Errors occur due to programmer's mistaken analysis of the error.

To remove logical errors is called debugging.

TOPIC-5
List, Tuples and Dictionary

Revision Notes

- **List**
  - A list is a standard data type of Python that can store a sequence of values belonging to any type.
  - The lists are depicted through square brackets.
  - These are mutable (i.e. modifiable), you can change elements of a list in place.

- Lists store a reference at each index.
- We can index, slice and access individual list elements.
- len (L) returns the number of items in the list L.Membership operators in and not in can be used with list.
- To join two lists, use `+' (concatenation) operator.
- L [start: stop] creates a list slice with starting index as start till stop as stopping index but excluding stop.
- List manipulation functions are
  append(), insert(), extend(),sort(), remove(), reverse() and pop().

➢ **Tuples**
- Tuples are immutable Python sequences, *i.e.* you cannot change elements of a tuple in place.
- Tuples' items are indexed.
- Tuples store a reference at each index.
- Tuples can be indexed sliced and its individual items can be indexed.
- len (T) returns count of tuple elements.
- Tuple manipulation functions are: len(), max(), min(), and tuple().

➢ **Dictionaries**
- Dictionaries in Python are a collection of some key-value pairs.
- These are mutable and unordered collection with elements in the form of a key : value pairs that associate keys to values.
- The keys of dictionaries are immutable type and unique.
- To manipulate dictionaries functions are : len(), clear(), has_key(),items(), keys(), values(), update().
- The membership operators in and not in work with dictionary keys only.

➢ Sorting means arranging the elements in a specified order i.e. either ascending or descending order.

➢ Two sorting techniques are
(i) **Bubble Sort –** It compares two adjoining values and exchanges them if they are not in proper order.
(ii) **Insertion Sort –** Suppose a list A with n elements A[1], A[2],…..A[n] is in memory. The insertion sort algorithm scans A from A[1] to A[N] inserting each element A[x] into its proper position in the previously sorted sub list A[1], A[2]…….A[x-1]

TOPIC-6
Strings in Python

Revision Notes

➢ Strings in Python are stored as individual character in contiguous location, with two way index for each location.

➢ Strings are immutable and hence item assignment is not supported.

➢ Following operations can be used on strings.
(1) Concatenation `+'
(2) Replication `*'
(3) Membership Operators as in and not in
(4) Comparison Operators as ==, ! =, <, >, < =, > =

➢ **Built in functions**
ord() – returns ASCII value of passed character.
chr() – returns character corresponding to passed ASCII code

➢ String slice refers to a part of the string, where strings are sliced using a range of indices
Syntax : string [n:m].

TOPIC-7
Python modules

Revision Notes

➢ A Python module can contain objects like docstrings, variables constants, classes, objects, statements, functions.

➢ Modules are accessed by using the import statement. A module is loaded only once, regardless the number of times it is imported.

Syntax (i) import module_name **(ii)** from <module> import <object>

Mathematical functions

➢ Mathematical operations can be performed by importing the math module. Different types of mathematical functions:

(i) sqrt() : find the square root of a specified expression

(ii) pow() : compute the power of a number

(iii) fabs() : Returns the absolute value of x.

(iv) ceil(x) : returns smallest integer value greater than or equal to x.

(v) floor(x) : returns the largest integer value less than or equal to x.

Random Functions

➢ Python offers random module that can generate random numbers. Different random functions are as follows

(i) random() : Used to generate a float random number less than 1 and greater than or equal to 0.

(ii) choice() : Used to generate 1 random number from a container.

Statistics Module

➢ To access Python's statistics functions, we need to import the functions from the statistics module.

Some statistics functions are as follows:

(i) mean() : Returns the simple arithmetic mean of data which can be a sequence or an iterator.

(ii) median() : Calculates middle value of the arithmetic data in iterative order.

(iii) mode() : Returns the number with maximum number of occurrences.

Know the Terms

➢ **Slicing:** In Python it is a feature that enables accessing parts of sequences like strings, tuples and lists.

➢ Debugging is the process of detecting and removing of existing and potential errors in a software code that can cause it to behave unexpectedly or crash.

➢ Debugger is a computer program used by programmers to test and debug a target program.

➢ Control Structure is a programming language construct which affects the flow of the execution of program.

➢ **Packing:** In Python, tuples are collections of elements which are separated by commas. It packs elements or value together so, this is called packing.

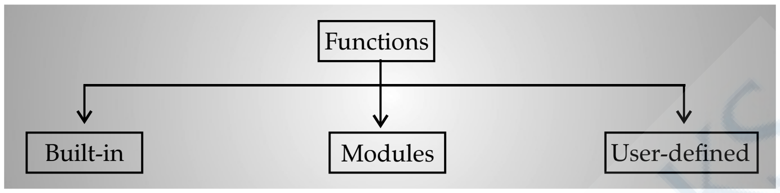
❑❑

CHAPTER-2
FUNCTIONS

Revision Notes

➢ A function is a named block of statements that can be invoked by its name. A function is organized and reusable code that is used to perform a single, given action. Functions provide better modularity for your application and a high degree of code reusability.

➢ The math module of Python provides mathematical functionality. Function blocks begin with the keyword def followed by the function name and parentheses e.g def sum( ): Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses. The first statement of a function can be an optional statement - the documentation string of the function or docstring, The code block within every function starts with a colon (:) and is indented. The statement return [expression] exit a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None. Defining a function only gives a name, specifies the parameters that are to be included in the function, a structure the blocks of code.



➢ **Examples of Some Built-in Functions**

(i) print() : It prints objects to the text stream file.

(ii) input() : It reads the input, converts it to a string and returns that.

(iii) sorted() : Returns a new sorted list from the items in iterable.

(iv) bool() : Returns a boolean value i.e., True or False.

(v) min() : Returns the smallest of two or more arguments.

(vi) any() : Returns True if any element of the iterable is True.

➢ The built-in functions of Python are always available, one needs not import any module for them. The math module of Python provides mathematical functionality.

- exp(x): Return e**x
- log(x,(base)): With one argument, returns the natural logarithm of x (to base e).
- With two arguments, returns logarithm of x to the given base calculate as log(x)/ log(base)
- log10(x): Returns logarithm of x at base 10. This is usually more accurate than log(x,10).
- pow(x, y): Returns x raised to the power y. In particular, pow(l.0, x) and pow(x, 0.0) always return 1.0, even when x is a zero or a NaN. If both x and y are finite, x is negative, and y is not an integer then pow(x, y) is undefined, and raises ValueError.
- sqrt(x): Returns the square root of x.
- cos(x): Returns the cosine of x radians.
- sin(x): Returns the sine of x radians.
- tan(x): Returns the tangent of x radians.
- degrees(x): Converts angle x from radians to degrees.
- radians(x): Converts angle x from degrees to radians.
- String Functions
  (i) partition(): It splits the string at the first occurrence of the given argument and returns a tuple containing three parts.
  (ii) join(): It takes a list of string and joins them as a regular string.
  (iii) split(): It splits the whole string into the items with separator as a delimeter.
- User-Defined Functions: User defined functions are those that we define ourselves in our program and then call them wherever we need.

➢ sys.stdin is the most widely used method to read input from the command line or terminal. The command line sys.argv argument is another way that we can grab input, and environment variables can also be used from within our programs.

➢ The scope of a variable determines the portion of the program where you can access a particular identifier.

There are two basic scopes of variables in Python :

1. Global variables
2. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope. All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable or the scope of variable

➢ **Passing different objects as arguments**

You can send any data types of argument to a function as string, number, list, dictionary etc., and it will be treated as the same data type inside a function.

e.g. List as an argument

```
   def fun(Fruit):
     for i in Fruit:
         print(i)
 Food = [“Mango”, “Cherry”, “Grapes”, “Banana”]
 fun(Food)
```

Output

```
     Mango
     Cherry
     Grapes
     Banana
```

➢ In Python, a number of mathematical operations can be performed with ease by importing a module named “math” which defines various functions which makes our task easier.

- ceil(x): Returns the ceiling of x as a float, the smallest integer value greater than or equal to x.
- floor(x) : Returns floor of x as a float, the largest integer value less than or equal to x.
- fabs(x): Returns the floating point absolute value of x.

➢ **Flow of Execution:** Flow of execution can be defined as the order in which the statements in a program are executed. The Python interpreter starts executing the instructions in a program from the first statement. The statements are executed one by one, in the order of appearance from top to bottom.

➢ If a def statement is encountered all the statements of the function are skipped but the function head is interpreted to check if it is valid.

➢ If a function call is encountered the statements in the called function are executed from top to bottom.

Know the Terms

➢ **Global Variables** are the one that are defined and declared outside a function and we need to use them inside a function.

➢ **Local Variables:** A variable declared inside the function's body and in the local scope is known as a local variable.

➢ **Doc Strings** are triple quoted string in Python module program which are displayed as document when help command is used.

➢ **Modularity:** The act of partitioning a program into individual components (modules) is called modularity.

➢ **Parameters** are variables listed within parentheses of a function header.

❑❑

CHAPTER-3
FILE HANDLING

Revision Notes

- Files are used to store huge collection of data and records permanently.
- Many applications require large amount of data. In such situation, we need to use some devices such as hard disk, compact disc etc, to store the data.
- **Need for a Data File**
  - It is a convenient way to deal with large quantities of data.
  - To avoid input of data multiple times during program execution.
  - To share data between various programs.
- **Types of files**
  - **Text files** store information in ASCII or Unicode characters. In text file, each line of text is terminated, (delimited) with a special character known as EOL (End of Line) character.
  - **Binary files** are just files that contain information in the same format in which the information is held in memory, i.e., In binary file, there is no delimiter for a line.
  - **CSV (Comma Separated Value) files** are a common file format for transferring and storing data.
- Access modes specify the type of operations to be performed on the opened file.
- **read**(), readline() and **readlines**() methods are available for reading data from the file.
- **write**() and **writelines**() are used for writing data in the file.
- **pickle** module is used in serialization of data. This allows us to store data in binary form in the file.
- **dump** and **load** functions are used to write and read data from file.
- The **open**() function creates a file object which would be utilized to call other methods associated with it.

  Syntax :

  file_object=open(filename[access_mode],[buffering])

  Here is the parameter details:
  - **filename:** The file name argument is a string value that contains the name of the file that you want to access.
  - **access_mode:** The access_mode determines the mode in which the file has to be opened i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).

File Opening Modes

MODES	DESCRIPTION
r	Opens a file for reading only in text format. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer will be at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer will be at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.

w+	Opens a file for both writing and reading. Overwrites the file if the file exists. If the file does not exist, creates a new file for reading and writing.
wb+	Opens a file for both writing and reading in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

➢ **Buffering:** If the buffering value is set to 0, no buffering will take place. If the buffering value is 1, line buffering will be performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action will be performed with the indicated buffer size. If negative, the buffer size is the system default (default behaviour).

➢ **The file object attributes:** Once a file is opened and you have one file object, you can get various information related to that file. Here is a list of all attributes related to the file object:

ATTRIBUTES	DESCRIPTION
file.closed	Returns True if file is closed, False otherwise.
file.mode	Returns access mode with which file was opened.
file.name	Returns name of the file.
file.softspace	Returns False if space explicitly required with print, True otherwise.

➢ **file () :** This is same as open ().

➢ **Random Access :** There are two functions that allow us to access a file in a non-sequential or random mode.
- **tell() :** It tells us the position of the file pointer.
- **seek() :** It moves the file pointer to the position specified.

➢ **Functions**

(a) read () : syntax: <file handle>.read([n])
It reads at most n bytes and returns the read bytes as string. If `n' is not specified it reads the entire file.

(b) readline () : syntax: <file handle>.readline ([n])
It reads a line of input, and returns it in the form of a string.

(c) readlines () : syntax: <file handle>.readlines ()
It reads all lines and returns them in a list.

(d) write () : syntax: <filehandle>.write (str1)
It writes string str1 to file referenced by <file handle>

(e) writelines () : syntax: <file handle>.writelines (L).
It writes all strings in list L as lines to file referenced by <file handle>

(f) flush () : syntax: <file object>.flush()
It forces the writing of data on disc that is still pending in output buffer.

(g) Importing sys module lets you read/write from the standard input/output device using sys.stdin.read () and sys.stdout.write().

(h) split () function splits a line in columns. It returns columns as items of a list.

(i) rename () function is used to rename a file existing on the disk.
syntax: os.remane(<current_file_name>,<new_file_name>)

(j) **remove ()** function is used to delete a file existing on the disk.

syntax: os.remove(<file_name>)

(k) **os.path.join ()** is used to assemble directory names into a path.

(l) **os.path.split ()** is used to split off the last path component.

(m) **os.path.splittext()** is used to split file name into primary name and extension.

(n) **os.path.exists ()** function checks if a path actually exists.

- **Absolute File Path :** It describes how to access a given file or directory starting from the root of the file system.
- **Relative File Path :** It is interpreted from the perspective of your current working directory.

Reading CSV files with CSV

Reading from a CSV file is done using the reader object. The CSV file is opened as a text file with Python's built in open() function, which returns a file object.

e.g.

```
import CSV
with open("Employee.txt") as CSV_file:
    CSV_reader = CSV.reader (CSV_file, delimiter = ',')
    line _count = 0
    for row in CSV_reader:
        if line_count = = 0:
            print (f' column names are {",".join(row}')
            line_count + = 1
        else :
            print (f '\t{row [0]} works in the {row[1]}
            department, and was born in {row[2]}.')
            line_count + = 1
    print (f'Processed {line_count} lines.')
```

Optional Python CSV reader Parameters

The reader object can handle different styles of CSV files by specifying additional parameters, some of which are shown below:

- Delimiter specifies the character used to separate each field. The default is the comma (',').
- quotechar specifies the character used to surround fields that contain the delimiter character. The default is a double quote (' " ').
- escapechar specifies the character used to escape the delimiter character, in case quotes are not used. The default is no escape character.

Writing CSV Files with CSV

You can also write to a CSV file using a writer object and the write_row() methods:

e.g.

```
import CSV
    with open ('Employee_file.CSV', mode = 'w') as Employee_file:
    Employee_writer = CSV.writer (Employee_file, delimiter
    = ',' quotechar = ' " ' quoting = CSV. Quote_Minimal)
    Employee_writer.writerow(['Rahul' , Manager' , 'April'])
    Employee_writer.writerow(['Neha' , 'IT' , 'June'])
```

Know the Terms

- CSV stands for Comma Separated Values.
- Pickle module can be used to store any kind of object in file as it allows us to store Python objects with their structure.
- File Handle serve as a link to a file residing on the computer.
- File Mode governs the type of operations possible in the operand file. The default mode is read ('r')
- flush () function forces the writing of data on disc still pending in output buffers.
- seek () method can be used to position the file object at particular place in the file.
- tell () method returns an integer giving the current position of file pointer in the file.

❑❑