Differentiate Between

1.

Type Casting ***(Explicit Conversion)*** and Automatic type conversion ***(Implicit Conversion)***

Type Casting(Explicit Conversion)	Automatic Type Conversion(Implicit Conversion)
It is an explicit process of conversion of a data from one type to another. (It is performed by the programmer.)	It is an implicit process of conversion of a data from one data type to another. It is performed by the compiler. It is also called as type promotion.
e.g. k=123.456 i= int(k)	e.g. i=2.3; j=10 i=j
Explicit conversion do require cast operator to perform conversion.	Implicit conversion do not require any special syntax.
In explicit conversion, data loss may or may not be take place during data conversion. Hence there is a risk of information loss.	In Implicit conversion, no data loss take place during the data conversion.

2.

Assignment and Arithmetic Assignment Operator

Arithmetic Assignment Operator	**Assignment**
Used to assign values to the variables after performing arithmetic operations.	**Used to assign values to the variables.**
respresented by (+=,-=,*=,/=,%=,//=)	**Represented by (=)**

3.

Global variable Vs Local Variable

Global variable	**Local Variable**
Global variables are defined outside of all the functions, generally on top of the program.	A local variable is declared within the body of a function or a block
The global variables will hold their value throughout the life-time of your program.	Local variable only use within the function or block where it is declare.

```
a=10            //global variable
def fun():
   b=20
   print(b)     //local variable
print(a)
```

4.

Actual and formal parameter/argument

Actual parameter/argument	Formal parameter/argument
The arguments that are passed in a function call are called actual arguments.	The formal arguments are the parameters/arguments in a function declaration.
These arguments are defined in the calling function.	The scope of formal arguments is local to the function definition in which they are used.

```
def sum(i, j, k):                    //called function //formal parameter/ arguments
   s = i + j + k;
   print(s)

a = 5;
sum(3, 2 , a);                       //calling function // actual parameter /arguments
```

3,2, a are actual arguments and i, j, k are formal arguments.

5.

Default argument / parameter

A default parameter (also called an optional parameter or a default argument) is a function parameter that has a default value provided to it. If the user does not supply a value for this parameter, the default value will be used. If the user does supply a value for the default parameter, the user-supplied value is used instead of the default value.

e.g.

```
def printValues(int x, int y=10):
   print(x)
   print(y)

printValues(1)          // y will use default parameter of 10
printValues(3, 4)               // y will use user-supplied value 4
```

6.

Void and Non Void Functions

Non Void Functions	Void Functions
Those functions which are returning values to the calling function	Those functions which are not returning values to the calling function.

Value return can be literal, variable , expression	We may use return but it will return none value to the function call
e.g. def fun(a,b): c=a+b return c x=int(input()) y= int(input()) z=fun(x,y) print(z)	e.g. def fun(a,b): c=a+b print(c) return x=int(input()) y= int(input()) z=fun(x,y) print(z)

7.

Arguments and Parameters

Arguments	Parameters
passed values in function call.	received values in function definition.
It can be of three types 1. Literals 2. Variables 3. Expressions	It should be of variable types.

def fun(a,b):
c=a+b
print(c)
 	
x=2
y=4
fun(x,y) #variables
fun(5,6) #literals
fun(x+3,y+6) #expressions
	def fun(a,b): #parameters
	c=a+b
	print(c)
	
	x=2
	y=4
	fun(x,y)

8.

Mutable and Immutable Data types

Mutable Data types	Immutable Data types
Object can be changed after it is created,	Object can't change its value in position after it is created.

Mutable is behaving like pass by reference	Immutable is behaving like pass by value
Mutable objects: list, dictionary	Immutable objects: int, float, complex, string, tuple
Everything in Python is an object ,and every objects in Python can be either mutable or immutable.	

9.

Module and Package

Module	Package
A module is a single file (or files) that are imported under one import and used.	A package is a collection of modules in directories that give a package hierarchy.
No _init_.py is required in module	In a package _init_.py file should be included

---

10.

Import Statement and From Import Statement

Import Statement	From Import Statement
import all the modules from that package	only imports the required module as specified

---

11.

r+ and w+

r+	w+
Opens a file for reading and writing, placing the pointer at the beginning of the file.	Opens a file for writing and reading, overwrites the existing file if the file exists. If the file does not exist, creates a new file for writing and reading

---

12.

r and a

r	a
Reading only	for appending

Sets file pointer at beginning of the file	Move file pointer at end of the file
This is the default mode.	Creates new file for writing,if not exist
e.g. f=open("abc.dat",'r')	e.g. f=open("abc.dat",'a')

13.

TEXT FILE AND BINARY FILE

TEXT FILE	BINARY FILE
A file whose contents can be viewed using a text editor is called a text file. (.txt)	**A binary file stores the data in the same way as as stored in the memory.**
A text file is simply a sequence of ASCII or Unicode characters.	**Best way to store program information.**
EOL (new line character i.e. enter) or internal translation occurs	**No EOL or internal translation occurs(not converted into other form becoz it is converted into computer understandable form i.e. in binary format)**
e.g. Python programs, contents written in text editors	**e.g. exe files,mp3 file, image files, word documents**

14.

Relative and Absolute Path

Relative Path	Absolute Path
The relative path is the path to some file with respect to current working directory	**The absolute path is the full path to some place on your computer.**
e.g. Relative path: **"function.py"** **or** **"..\function.py**	**For example: Absolute path:** **C:\Users\hp\Desktop\cs\function.py**

15.

seek() and tell()

seek()	tell()
takes the file pointer to the specified byte position	**it gives current position within file**
Syntax: **seek("no_of_bytes_to_move", "from_where")** **"from_where"- has 3 values**	**Syntax** **fileobjectname.tell()** **Example:** **f.tell()**

from=0 -means to move from the beginning of file. It is default also **from=1 means to move the pointer at the current position** **from=2 means to move pointer at end of file**	

16.

LIST and STRING

LIST	STRING
Lists are mutable	strings are immutable.
In consecutive locations, list stores the references of its elements.	In consecutive locations, strings store the individual characters
lists can store elements belonging to different types.	Strings store single type of elements-all characters
It is represented by []	It is represented by " " or ' '
e.g. L=[1,2,3,4]	e.g. s="hello" s1='world'

17.

LIST and TUPLES

LIST	TUPLES
lists are mutable.	Tuples are immutable
List can grow or shrink	tuples cannot grow or shrink
For list []symbol is used	For tuples () symbol is used
e.g. L=[1,2,3,4]	e.g. T=(1,2,3,4)

18.

LIST and DICTIONARY

LIST	DICTIONARY
lists are sequential collections(ordered)	dictionaries are non-sequential collections(unordered).
In LIST the values can be obtained using positions	But in dictionaries the values can be obtained using keys

this thing is not possible in list.	By changing the sequence of key we can shuffle the order of elements of dictionary
e.g. L=[1,2,3,4]	e.g. d={1:"hello",2:"world"}

19.MODULE, PACKAGE AND LIBRARY

MODULE	PACKAGE	LIBRARY
• Module is a file which contains python functions, global variables etc. • It is nothing but .py file which has python executable code / statement	• Package is a collection of modules. • Each package in Python is a directory which MUST contain a special file called __init__.py. • This file can be empty, and it indicates that the directory it contains is a Python package, so it can be imported the same way a module can be imported	• Library is a collection of packages. • There is no difference between package and python library conceptually. • Some examples of library in Python are: Python standard library containing math module, random module, statistics module etc.

20.DOCSTRING AND COMMENT

Docstring	Comment
Docstrings are similar to commenting, but they are enhanced, more logical, and useful version of commenting. Docstrings act as documentation for the class, module, and packages.	Comments are mainly used to explain non-obvious portions of the code and can be useful for comments on Fixing bugs and tasks that are needed to be done.
Docstrings are represented with opening and closing quotes	comments can start with a # at the beginning.
docstring can be accessed with the help function.	The comments cannot be accessed with the help function

21. TEXT FILE, BINARY FILE AND CSV FILE.

TEXT FILE	BINARY FILE	CSV FILE
A file whose contents can be viewed using a text editor is called a text file. (.txt)	**A binary file stores the data in the same way as as stored in the memory. (.dat)**	Data is stored in the form of rows and column i.e in tabular form. (.csv)
A text file is simply a sequence of ASCII or Unicode characters.	**Can store different types of data (audio, text,image) in a single file.**	They are plain text files having ASCII/Unicode Characters
EOL (new line character i.e. enter) or internal translation occurs	**No EOL occurs**	Language which support text files will also support csv files.

Widely used file format and can be opened in any text editor.	**Developed for an application and can be opened in that application only.**	CSV file can be open in excel, word, notepad or any text editor or any programming language which supports file processing
e.g. Python programs, contents written in text editors	**e.g. exe files,mp3 file, image files, word documents**	e.g. CSV is processed by almost all existing applications

22. TEXT FILE AND CSV FILE

TEXT FILE	CSV(COMMA SEPARATED VALUES) FILE
Text files contain text which can be opened by any text editor and there is plain text with no format	**CSV also contain text data but in a format where each line is considered as row/record which has many fields(columns).**
EOL (new line character i.e. enter) or internal translation occurs	**fields are the values separated by a delimiter like , " ' ,"*","/" ,"\n"etc.**
No title is required	**the first record is the title of each field.**
No need to import any module	**csv module must be imported**

23. 'a' AND 'w' MODE

'w'	'a'
'w' Open a file for writing	'a' Open for appending at the end of the file without truncating it.
Creates a new file if it does not exist or truncates the file if it exists.	Creates a new file if it does not exist.

24. write() and writelines()

write()	writelines()
write() function write a single string at a time	writelines() methods can be used to write a sequence of strings

25. PICKLING AND UNPICKLING

PICKLING	UNPICKLING
Pickling is the process whereby a Python object is converted into a byte stream.	Unpickling is the process by which a byte stream is converted back into the desired object.

26. readline() and readlines()

readline()	readlinelines()

The readline() method reads one line(i.e. till newline) at a time from a file and returns that line	The readlines()method reads the entire content of the file in one go and returns **a list of lines** of the entire file.
It reads the file till newline including the newline character.	
The readline() method returns an empty string when the end of file is reached.	This method returns an empty value when an end of file (EOF) is reached.

27. read() and readline()

read()	**readline()**
The read() method reads the entire file content of the file in one go	The readline() method reads **one line(i.e. till newline) at a time** from a file
it reads info character by character	It reads the info line by line
	The readline() method returns an empty string when the end of file is reached.

28. DDL AND DML

DDL	**DML**
Data definition language	Data manipulation language
Create Drop Alter	Insert Update Delete Select

29. where and having clause

where	**having**
Where- Where clause is used to specify condition on single row.	**having-** It is used to mention condition in Group
Where clause is used mostly with Select, Update and Delete command/query	Having clause is used only with group by clause