Theory notes

Q What is Python?

Ans Python is a programming language with objects, modules, threads, exceptions and automatic memory management.

Q What are the benefits of using Python?
The benefits of pythons are

- it is simple and easy,
- portable, extensible,
- build-in data structure and
- it is an open source.

Q How Python is interpreted?
Ans
Python language is an interpreted language. Python program runs directly from the source code.
It converts the source code that is written by the programmer into an intermediate language, which is again translated into machine language that has to be executed.

Q Is python a case sensitive language?
Ans Yes, python is a case sensitive language, it means python considers lowercase and uppercase differently.
Like if we have written
Num=3
num=24
python will consider both the variables differently though their pronunciation is same.

Q What are Character Sets in Python?
Ans Character sets are the combination of numbers, alphabets and special symbols. These are set of valid characters recognized by python
character sets are:

1. letters- A-Z,a-z
2. digits- 0-9
3. special symbols- &,%,,$,@,#,;,{,[,),?,/,:,”,’,>,<,+,!,^ etc
4. whitespaces- tab,blank space(white space),escape sequences(newline,alert,formfeed,vertical tab,carriage return)etc.

Q What is indentation in Python?
Ans

- Indentation refers to the **spaces applied at the beginning** of a code line.
- In Python, it is very important.
- Python uses indentation to **indicate a block of code or used in block** of codes.
  e.g.
  if 3 > 2:
  print(“Three is greater than two!") #syntax error due to not indented

Q What are Tokens?
Ans
Smallest piece of code.
OR
Smallest individual unit in a program

Q What are the different types of Tokens?
Ans

1. Keyword(system defined names)
2. Identifier(user defined names)
3. Literals

4. Operators
5. Punctuators

Q Explain Keyword .
Ans Keywords are the reserve words/pre-defined words/special words of python which have a special meaning for the interpreter.
e.g

False	**True**	**None**	**def**	**if**
lambda	**class**	**yield**	**continue**	**else**
assert	**or**	**while**	**break**	**elif**
del	**from**	**is**	**not**	**pass**
For	**global**	**finally**	**import**	**as**
in	**nonlocal**	**return**	**with**	**and**
int	**except**	**raise**	**print**	**csv**
pickle	**reader**	**writer**	**dump**	**load**
sys	**connector**	**cursor**	**execute**	**fetch**

Q What are identifiers?
Ans
Identifiers are the name given to the different programming elements like variables, functions, lists, dictionaries etc.

Q What are the naming rules of Identifiers?
Ans
1. Spaces are not allowed
2. Special symbols like $%^&#@! Not allowed
3. Must made up of only letters,numbers and underscore(_)
4. Can't begin with a number
5. Keywords are not allowed
6. Can start with underscore(_)

e.g.
some valid identifiers are:

myfile, _admno, no1, roll_no, d_3_21, For

some invalid identifiers are:
roll-no (special character hyphen(-))
3_d_21 (start with a digit)
for (keyword)
admno. (contains special character .(dot))
no.1 (contains special character .(dot))

Q Explain the different types of Literals in Detail.

Ans

1. Numeric Literals- **are numeric values like integer floating point number or a complex number**
   (a) Integer literals- whole numbers. These can be in the form of
      - decimal (0-9 e.g. 2345, -2345)
      - octal(begin with Oo e.g.Oo21)
      - Hexadecimal (begin with Ox. e.g OxBD)
   (b) Floating literal – integer with decimal (e.g.-13.0,3.5)
   (c) Complex (e.g. 2+3j here 2 and 3 are real and j are imaginary)

2. String literal- **is a sequence of characters surrounded by Quotes (Single, Double or Triple Quotes).**

String literal can be

(a) single line strings- must terminate in one line

e.g. t=”hello world”

(b) multi line strings- spread across multiple lines

e.g. t= “hello
world”
t1=’’’hello world’’’ or “””hello world”””

3. Special literal –none (empty legal value)- **is to indicate absence of value**

4. Boolean literal- **to represent one of the two Boolean Values i.e. True or False**

Q What are operators?

Ans

Operators are the symbols or words that perform some kind of operation on given values (operands) in an expression and returns the result.

Types of operators are:

arithmetic	**+,-,/,*,%,**,//**
bitwise	**&, ^, |**
Identity	**is, is not** **(these are used to compare the memory locations of two objects). These can be used in place of == (is) and != (is not)**
Relational (comparison)	**>,<,>=,<=,==,!=** **(these operators are used to compare the values)**
logical	**and, or, not** **(these are used to perform logical operations on the given two variables or values.)**
shift	**<<, >>**
Assignment	**=** **(these are used to assign values)**
Membership	**in, not in** **(these operators used to validate whether a value is found within a sequence such as such as strings, lists, or tuples.)**
arithmetic-assignment	+=, -=, //=, **=, *=, /=

Q Define Python Operator Precedence .

Ans

PEMDAS

Parentheses|Exponentiation|Multiplication|Division|Addition|Subtraction

Operators	**Meaning**
()	Parentheses
****	Exponent
***,/, //, %**	Multiplication, Division, Floor, Division, Modulus
+,-	Addition, Subtraction
==,!=,>,>=,<,<=, is, is not, in, not in	Relational,Identity,Membership Operators
Not	Logical NOT
And	Logical AND
Or	Logical OR

Q What are Punctuators?

Ans

Punctuators are the symbols that are used in programming language to organize sentence structure, indicate the rhythm and emphasis of expressions, statements and Program Structure.

Common Punctuators are:

‘ ’ “” ( ) { } [ ] # / , : ; . @

Question 1:

What is the difference between a keyword and an identifier ?

Answer:

Keyword is a special word that has a special meaning and purpose. Keywords are reserved and are few. For example : if, else, elif etc.

Identifier is the user-defined name given to a part of a program like variable, object, functions etc. Identifiers are not reserverd. These are defined by the user but they can have letters, digits and a symbols underscore. They must begin with either a letter or underscore. For example : chess, _ch, etc.

Question 2:

What are literals in Python ? How many types of literals are allowed in Python ?

Answer:

Literals mean constants i.e. the data items that never change value during a program run. Python allow five types of literals :

String literals

Numeric literals

Boolean literals

Special literal (None)

Literal collections like tuples, lists etc.

Question 3:

How many ways are there in Python to represent an integer literal ?

Answer:

Python allows three types of integer literals :

Decimal (base 10) integer literals.

Octal (base 8) integer literals.

Hexadecimal (base 16) integer literals.

For example, decimal 12 will be written as 14 as octal integer and as OXC as hexa decimal integer.

$(12)_{10} = (14)_8 = (OXC)_{16}$. (as hexa decimal)

Question 4:

How many types of strings are supported in Python ?

Answer:

Python allows two string types :

Single line strings : Strings that are terminated in single line. For example :

str = ‘Oswal Books’

Multiple strings : Strings storing multiple lines of text. For example :

str = ‘Owal \

Books’

or str = ” ” ” Oswal

Books

” ” “

Question 5:

What is “None” literal in Python ?

Answer:

Python has one special literal called ‘None’. The ‘None’ literal is used to indicate something that has not yet been created. It is also used to indicate the end of lists in Python.

Question 6:

What factors guide the choice of identifiers in Programs ?

Answer:

An identifier must start with a letter or underscore followed by any number of digits or/ and letters.

No special character (other than under-score) should be included in the identifier.
No reserved word or standard identifier should be used.
Upper and lower case letters are different. All characters are significant.

Question 7:

What will be the size of the following constants : "\a". "\a", "Manoj\'s", '\", "XY\ YZ"

Answer:

'\a' – size is 1 as there is one character and it is a string literal enclosed in single quotes.
"\a" – size is 1 as there is one character enclosed in double quotes.
"Manoj\'s" – size is 7 because it is a string having 7 characters enclosed in double quotes.
"\" – size is 1. It is a character constant and is containing just one character \".
"XY\ – size is 4. It is a multiline string create YZ" with \ in the basic string.

Question 8:

What is used to represent Strings in Python ?

Answer:

Using Single Quotes (')

You can specify strings using single quotes such as 'Quote me on this'. All white space i.e. spaces and tabs are preserved as it is.

Using Double Quotes (")

Strings in double quotes work exactly the same way as strings in single quotes. An example is "What's your name?"

Using Triple Quotes o("' or " " ")

You can specify multi-line strings using triple
quotes. You can use single quotes and double
quotes freely within the triple quotes. An example
is
"'This is a multi-line string. This is the first line. This is the second line.
"What's your name?," I asked.
He said "syed saif naqvi."

Question 9:

Which of the following variable names are invalid ? Justify.
(a) try
(b) 123 Hello
(c) sum
(d) abc@123

Answer:

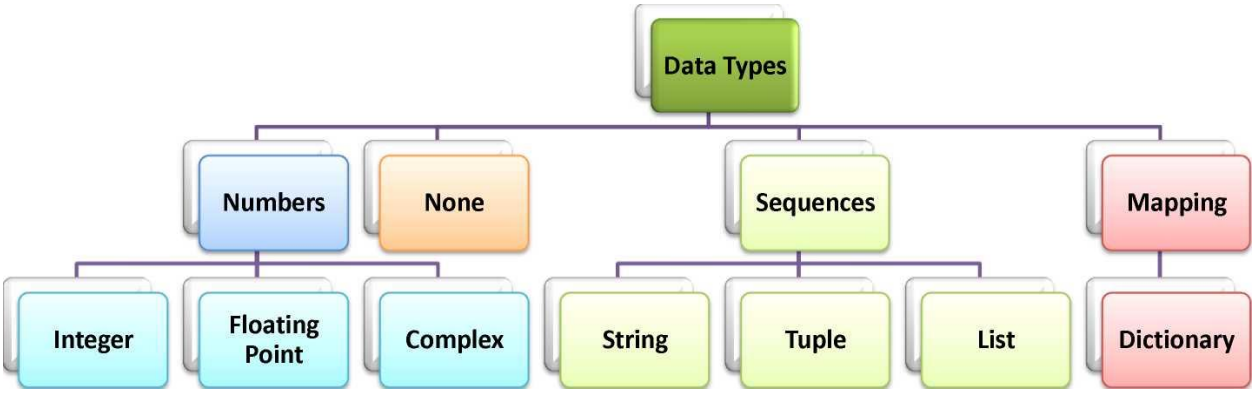
(a) try : is a keyword can't be used as an identifier.
(b) 123 Hello : Variable names can't start with a digit.
(c) abc@123 : Special characters aren't allowed in variable names.

DATA TYPES

Q What do you mean the term Data types in Python?
Ans
Data types are used to identify the type of data and set of valid operations which can be performed on it.

Q How many types of data types in Python?
Ans

1. **Numbers(integer(whole no), floating(number with decimal)**
2. **String**
3. **List**
4. **Tuple**
5. **Dictionary**



Q Which function is used to find the data type of an variable
Ans type() function is used to find the data type of any variable,object or function.

e.g.

```
>>>x=10
>>>type(x)
<class 'int'>

>>>y=12.3
>>>type(y)
<class 'float'>

>>>type(12.45)
<class 'float'>
```

Q Explain the concepts of different python data types in detail.
Ans

1. Number data type- **used to store numeric values.**
   a. **Integer- it occupies** Unlimited range (subject to available (virtual) in computer memory.
   b.
      i. **signed integer –** stores signed or unsigned
         e.g. -123, 342

      ii. **Booleans-** True or False
         1 or 0

```
>>>a=3>10
>>>print(a)
False
```

   c. **Floating point Numbers-** Represents double precision floating point numbers (15 digit precision) e.g. -1234.56, 1234.56.

   d. **Complex Numbers-** represents Complex Numbers in the form A + Bj

```
e.g.
>>>x=5+4j
>>>type(x)
<class 'complex'>
```

2. String data type- **is an ordered data type that can hold any known character like letters, numbers, special characters etc**

   e.g. **"abcd", "$@&%", '???', "1234", "apy"**

   >>>s="hello"

```
>>>type(s)
<class 'str'>
>>>type("hello")
<class 'str'>
```

3. List data type- **is an ordered group of comma-separated values of any datatype enclosed in *square brackets []***

e.g. [1,23,36,48,5], ['teena', 101, 90.5], ['a', 'e', 'i', 'o', 'u']

```
>>>a=[1,23,36,48,5]
>>>type(a)
<class 'list'>
>>>b=["teena","meena","sheena"]
>>>type(b)
<class 'list'>
```

Elements in a List can be individually accessed using its index (positive or negative)					
Positive index value	0	1	2	3	4
List	1	23	36	48	5
Negative index value	-5	-4	-3	-2	-1

Positive index value	0	1	2
List	'teena'	101	90.5
Negative index value	-3	-2	-1

Positive index value	0	1	2	3	4
List	'a'	'e'	'I'	'o'	'u'
Negative index value	-5	-4	-3	-2	-1

4. Tuple Data Type- **is an ordered group of comma-separated values of any datatype enclosed within *parentheses ()*.**

e.g. (1,23,36,48,5), 'teena', 101, 90.5, ('a', 'e', 'i', 'o', 'u')

```
>>>a=(1,23,36,48,5)
>>>type(a)
<class 'tuple'>
>>>b='teena', 101, 90.5
>>>type(b)
<class 'tuple'>
```

Elements in a tuples can be individually accessed using its index (positive or negative)					
Positive index value	0	1	2	3	4
tuple	1	23	36	48	5
Negative index value	-5	-4	-3	-2	-1

Positive index value	0	1	2
tuple	'teena'	101	90.5
Negative index value	-3	-2	-1

Positive index value	0	1	2	3	4
tuple	'a'	'e'	'I'	'o'	'u'
Negative index value	-5	-4	-3	-2	-1

5. Dictionary Data type-**is an unordered set of comma-separated key:value pair enclosed within *curly braces {}*.**

e.g. vowels ={'a':1, 'b':2, 'c':3, 'd':4, 'e':5}

here, 'a', 'b', 'c', 'd', 'e' are the keys of dictionary vowels & 1,2,3,4,5 are the values for these keys respectively.

```
>>>a={1:'teena',2:'heena',3:'sheena'}
>>>type(a)
<class 'dict'>
```

Q What are mutable and immutable data types?
Ans
A **mutable data type** can change its state or contents
e.g.
list, dict, byte array

Immutable data type cannot change its state or contents .
e.g.
int, float, complex, string, tuple, bytes , set

Q How we can find the address of any identifier or variable ?
Ans By using id() we can find the address or memory location of any variable.

```
>>>x=10
>>>id(x)
```

Q What is Type conversion? Explain Implicit and Explicit type Conversion.
Ans
The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion.
Python has two types of type conversion.

- Implicit Type Conversion
- Explicit Type Conversion

Implicit type conversion- Python automatically converts one data type to another data type. This process **doesn't need any user** involvement.

```
>>>a=3
>>>b=3.4
>>>d=a+b
>>>d
6.4
```

Explicit Type conversion- users convert the data type of an object to required data type. We use the predefined functions like **int(),float(),str()** etc.

```
>>>a=3
>>>c=3.4
>>>str(a) #str() function converts interger to string.

'3'
>>>str(c) #str() function converts float number to string
'3.4'
>>>b=3.4
>>>d='3'
>>>int(b) # int() function converts floating number to integer
3
>>>int(d) # int() function converts string to integer
3
>>>d=input("enter any number") #input() takes value in the string form
4
>>>d
'4'
>>>d=int(input("enter any number")) #int()function converts string to integer
4
>>>d
4
```

input() function always enter string value in python .
There is need of int(),float() function can be used for data conversion.

Q What are Escape Sequences or Backslash Character Constants?
Ans

1. These are some non-printable or non-graphic characters which are mainly for formatting(display purpose) and used only with print().
2. All escape sequences occupies **one byte** in computer memory .

3. An escape sequence **always starts with backslash** followed by one or more special characters.
4. **Escape Sequences** must be enclosed in single quotes or in double quotes.
e.g.

```
print('\n')
print("\n")
```

or

```
print("\n hello \n world")
```

few Escape sequences are:

Escape Sequence	Description
\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	New line or ASCII Linefeed (LF)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)
\v	ASCII Vertical Tab (VT)
\ooo	Character with octal value ooo
\xhh	Character with hex value hh

Q What are comments in Python ?

Ans

- A **comment** is text that doesn't affect the outcome of a code.
- it is just a piece of text to let someone know what you have done in a program or what is being done in a block of code.
- It is readable for programmer(a person who is writing the code) but ignored by python interpreter.

TYPES of comments :

i. **Single line comment**: Which begins with # (hash)sign.

ii. **Multi line comment (docstring):** either write multiple line beginning with # sign or use triple quoted multiple line.

e..g.

```
'''this is to check the concept of

python multiline comment '''
```

Q Explain the concept of variable in Python.

Ans

- Variable is a name given to a memory location.
- A variable can consider as a container which holds value.
- Python is a type infer language that means you don't need to specify the datatype of variable.
- Python automatically get variable datatype depending upon the value assigned to the variable.

Variable name follows the same rule as we are using for identifiers

Variable name= Identifier name

Q How we can declare a variable?

Ans

Syntax

Assigning single Value to Variable

variable_name= value

e.g.

```
name = 'python'          # String Data Type
num = 2                  #integer data type
roll_no=1                #integer data type
```

#ASSIGNING DIFFERENT VALUES TO MULTIPLE VARIABLES

variable_name1, variable_name2= value_of_variable1, value_of_variable2

e.g.

```
>>>a,b=3,4
>>>a
3
>>>b
4
```

#ASSIGNING SAME VALUES TO MULTIPLE VARIABLES

variable_name1, variable_name2= value_of_variable1, value_of_variable2

e.g.

```
>>>a,b=0,0
```

or

```
>>>a=b=0
```

```
>>>a
0
>>>b
0
```

CONDITIONAL STATEMENTS
(DECISION MAKING)

- The basic decision statements in computer is selection structure.
- The decision is described to computer as conditional statement that can be answered True or False
- Python language provide the following conditional (decision making) statements.
    - If statement

It is used to control the flow of execution of the statements and also used to test logically whether the condition is true or false.	
Syntax: if test_expression **:** Statement	e.g. n=int(input("enter number")) if(n<=10): print("condition is true")

    - If….else statement

The if…else statement is called alternative execution , in which there are two possibilities and the condition determines which one gets executed	

Syntax:
if test_expression :
Statement of if
else:
Statement of else
	e.g.
	n=int(input("enter number"))
	if(n<=10):
	print("condition is true")
	else:
	print("condition is false")

    - If …..elif…..else statement-

elif- is a keyword used in Python in replacement of else if to place another condition in the program. This is called chained conditional. Chained conditions allows than two possibilities and need more than two branches	

Syntax:
if test_expression :
Statement of if
elif expression:
Statement of elif
else:
Statement of else
	e.g.
	n=int(input("enter number"))
	if(n<=5):
	print("condition is less than 5 ")
	elif (n<=10 and (n>5):
	print("condition is between 10 and 5")
	else:
	print("condition is more than 10")

- Nested if…else statement

We can write an entire **if..else statement in another if** ….else statement called nesting, and the statement is called **nested if**. In a nested if construct, you can have an if…elif…else construct inside an if…elif…else construct	

Syntax:
if test_expression :
Statement(s)
if test_expression :
Statement(s)
elif expression:
Statement(s)
else:
Statement(s)
	e.g.
	n=int(input(“enter number”))
	if n<=15:
	if n==10:
	print(“ok”)
	else:
	print(“use another option”)
	else:
	print(“more than 15”)

CONTROL STATEMENT
(LOOPING STATEMENT)

- Program statement are executed sequentially one after another. In some situations, a block of code needs of times
- These are repetitive program codes, the computers have to perform to complete tasks.
- The following are the loop structures available in python
    - while statement
    - for ….loop statement
    - Nested loop statement

while loop

A while loop statement in python programming language repeatedly executes a target statement as **long as a given condition is true.**	
Syntax: while expression: statement(s)	e.g. n=int(input(“enter no”)) s=0 while(n>0): s=s+n n=n-1 print(“the sum is”,s)

while loop- infinite loop

while 1: print(“*”) OR while 1: print(“*”)	while True: print(“*”) OR while True: print(“*”)

else statement with while loops

• Python supports have an else statement associated with a loop statement
• If the else statement is used with a while loop, the else statement is executed when the condition false.
	e.g.
	c=0
	while c<3:
	print("inside loop")
	c=c+1
	else:
	print("outside loop")

For loop

• The for loop is another repetitive control structure, and is used to execute a set of instructions repeatedly, until the condition becomes false. • The for loop in python is used to iterate over a sequence(list,tuple,string) or other iterable objects. Iterating over a sequence is called traversal.	

Syntax:
for val in expression:
Body of the for loop
 	
expression -> tuple|string|list|dictionary|range()
	e.g.
	for i in [1,2,3]: #list usage
	print(i)
	for i in (1,2,3): #tuple usage
	print(i)
	for i in "hello": #string usage
	print(i)
	for i in {1:'a',2:'b'}: #dictionary usage
	print(i)

range(start, end-1, step_value)
note:
if only one value is specified then it takes only end-1 and will take 0 as starting value
	e.g.
	for i in range(5): #take values 0,1,2,3,4
	print(i)
	for i in range(1,5): #take values 1,2,3,4
	print(i)
	for i in range(1,5,2): #take values 1,3
	print(i)

JUMP STATEMENTS

- Jump statements are used to transfer the program's control from one location to another.
- Means these are use d to alter the flow of a loop like - to skip a part of a loop or terminate a loop.

There are three types of jump statements used in python.

1.break

2.continue

3.pass

break	**continue**	**pass**
It is used to terminate the loop	It is used to skip all the remaining statements in the loop and move controls back to the top of the loop.	• This statement does nothing. • It can be used when a statement is required

		syntactically but the program requires no action.

for val in "string":
if val == "i":
break
print(val)
print("The end")
	for val in "string":
	if val == "i":
	continue
	print(val)
	print("The end")
		for val in "string":
		if val == "i":
		pass
		print(val)
		print("The end")

Output: s t r The end	Output s t r n g The end	Output s t r i n g The end

---

STRING DATA TYPE

Elements in a string can be individually accessed using its index (positive or negative)					
Positive index value	**0**	**1**	**2**	**3**	**4**
string	**H**	**E**	**L**	**L**	**O**
Negative index value	**-5**	**-4**	**-3**	**-2**	**-1**

	STRING (store text type of data)	LIST	Tuple
similarities	**1.Slicing-** extract limited information or access a range of characters		
	2. Elements can be individually accessed using its **index (positive or negative)**		
	3. Iterating/Traversing - Each character can be accessed sequentially **using for loop.**		

for i in "hello": #string usage
print(i)
	for i in [1,2,3]:
	print(i)
		for i in (1,2,3):
		print(i)

s="hello" for i in s: print(i)	t=[1,2,3,4] for i in t: print(i)	t=(1,2,3,4) for i in t: print(i)
s="hello" for i in range(len(s)): print(i,s[i])	t=[1,2,3,4] for i in range(len(t)): print(i,t[i])	t=[1,2,3,4] for i in range(len(t)): print(i,t[i])

4. Common functions

+ (concatenation (combine)), * (replicate),		
s="hello" print(s+"world") print(s*2)	t=[1,2,3] print(t+[4,5,6]) print(t*2)	t=[1,2,3] print(t+[4,5,6]) print(t*2)
len(), in (check for availability, not in count(element/string)--- index(value)		
s="hello" print(s.count('l')) print(len(s)) print(s.index('l'))	t=[1,2,3,4,2] print(t.count(2)) print(len(t)) print(s.index(2))	t=(1,2,3,4,2) print(t.count(2)) print(len(t)) print(s.index(2))

	if 'l' in s: print("ok")	if 3 in t: print("ok")	if 3 in t: print("ok")
Differences			
	Strings are immutable(no change of value in place)	**list are mutable(change of value in place)**	**tuples are immutable(no change of value in place)**
	declare empty string: variable _name= quotation marks e.g.: str=' ' OR str=" "	**declare empty list:** variable _name= [] e.g.: t1=[]	
	str.capitalize()-capitalise first letter of a string str.title()-captialise first letter of each word str.upper()-converts all characters into uppercase str.lower()-converts all characters into lowercase str.count('char')-count a particular character or word str.find(sub)-find the position of searching word or character str.replace(old string,new string)-replace a string with another **str.index()-find the position of a character** **str.isalnum()-check availability of bothalphabets and numbers** **str.isalpha()-check only alphabets** **str.islower()-check whether string is lowercase or not** **str.isnumeric()-check only numbers** **str.isspace()-check only blanks** **str.istitle()-check only first character is in capital or not** **str.isupper()-check the string is in uppercase or not**	**append()** -is used to add an Item to a List. e.g. list.append(3) **append() is also used to create nested list** also e.g. list.append([3,4]) --- **extend()**- can be used to add multiple item at a time in list e.g.list.extend([3,4]) Delete Item From A List- del list[0:2] # delete first two items del list # delete entire list list.insert() insert an Item at a defined index list.remove() remove an Item from a list del list.clear() empty all the list list.pop() Remove an Item at a defined index list.sort() Sort the items of a list in	**tuple(seg) Converts a list into tuple.** **min(tuple) Returns item from the tuple with min value.can be done via list.** **max(tuple) Returns item from the tuple with max value.can be done via list.** sum()- sum of tuple elements can be done via list x = (1,4,6) r= sum(list(x)) print('sum of elements in tuple', r) OUTPUT->11 sorted()- returns the sorted elements list x = (1,4,6) r= sorted(x) print(sorted elements in tuple', r)

	str.lstrip(char) –remove spaces from left side of string **str.rstrip(char)-remove spaces from right side of string** **str.strip(char)-remove spaces from both left and right side** **str.split()-divide the sentence according to spaces or particular characters** **endswith(s1)-check whether string ends with particular character**	ascending or descending order list.reverse() Reverse the items of a list max(list) Return item with maximum value in the list. min(list) Return item with min value in the list. list(seq) Converts a tuple, string, set, dictionary into list.	

DICTIONARY

- Dictionaries are mutable - (can modify its contents but Key must be unique and immutable)
- Unordered collection of elements (each item consist of a key and a value.)
- In the form of Key-Value Pairs which are enclosed in curly braces{}.
- In dictionary keys are unique but values can be duplicate.
- Keys are immutable but values are mutable.
- In dictionary, values can be retrieved only through keys

create an empty dictionary which are as follows

- A = { } # A is an empty dictionary
- A = dict( ) # dict( ) method will create an empty dictionary.

delete the elements of dictionary:

- 1.By using del statement :
- 2. By using pop () function #here pop() function takes keys instead of position

d={1:'a',2:'b',3:'c'}

get()-**return the value of the required Key. e.g. d.get(1)**

clear()-removes all items from the dictionary. e.g. d.clear()

items()-all the key value pair of dictionary in the form of list of tuples. d.items()

keys()-all the keys of dictionary in the form of List. e.g. d.keys()

values()-all the values of dictionary in the form of List. e.g. d.values()

update()-merge the key value air of two dictionary into one. e.g. d.update({4:'e',5:'f'})

setdefault() method returns the value of the item with the specified key. If the key does not exist, insert the key, with the specified value.

e.g.

d.setdefault(6,'g')

max() – returns key having maximum value. e.g. max(d)

sorted- sort by key or value

e.g.

sorted(d.items(),reverse=True)

sorted(d)

Function

A function is a subprogram that acts on data and often returns a value

Advantages

1. **Program development made easy and fast :** Work can be divided among project members thus implementation can be completed fast.

2. Program testing becomes easy

3. Code sharing becomes possible

4.Code re-usability increases

5. Increases program readability

Types of Functions

- Built –in functions (function using Libraries)- **Pre-defined functions like int(),type(),float(),str(),print(),input(),ord(),hex(),oct() , len() etc**
- Functions defined in the modules(function using Libraries)- **functions that are in particular modules like sin(),floor(),ceil(),dump(),load() etc**
- User defined functions- **defined by the programmer**

PARTS OF USER DEFINED FUNCTIONS

- Function definition
- Arguments(those variables which are use in **function calling**)
- Parameters(those variables which are use in **function definition**)
- Function Calling-
    - **void functions**- those functions which are **not returning values** to the calling function
    - **Non void functions-** those functions which are **returning values** to the calling function.

Value return can be literal, variable , expression

Q What are Arguments?

Ans Arguments- passed values in function call.

Passed values can be of three types 1. Literals 2. Variables 3. Expressions	e.g. 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

Q What are Parameters?

Ans <u>Parametes-</u> received values in function definition.

It should be of variable types.	e.g.
	def fun(a,b): #parameters
	c=a+b
	print(c)

Q What are actual and formal arguments /parameters?

Ans

A formal parameter, i.e. a parameter, is in the function definition.	**An actual parameter, i.e. an argument, is in a function call.**
def sum(x,y): #x, y are formal arguments **z=x+y** **return z #return the result** **x,y=4,5** **r=sum(x,y) #x, y are actual arguments** **print(r)**	

Q What is the use of return statement?

Ans It is used to return either a single value or multiple values from a function.

Q Can Python return Multiple values and in what forms?

Ans Python can return Multiple Values from Functions

It can be in the form of **tuples** :

1. Received values as tuple	e.g. def fun(a,b): return a+b,a-b x=2 y=4 z=fun(x,y) print(z)
2.	e.g.

Unpack received values as tuple	def fun(a,b): return a+b,a-b x=2 y=4 d,z=fun(x,y) print(d,z)

Q Explain different types of arguments in detail

Ans

1. Positional parameters(Required Arguments) - these arguments must be provided for all parameters	2. **Default Arguments-** if right parameter have default value then left parameters can also have default value. this argument can be skipped at the time of function calling	3. **Keyword Arguments(Named Arguments)-** We can write arguments in any order but we must give values according to their name

e.g.
def fun(a,b):
c=a+b
print(c)
x=10
y=3
fun(x,y)
	e.g.
	def fun(a,b,c=3):
	d=a+b+c
	print(d)
	x=10
	y=3
	fun(x,y) #here c parameter value is 3
	z=5
	fun(x,y,z) #here it will be take parameter c value as 5
		e.g.
		def fun(a,b,c):
		print(a,b,c)
		
		fun(b=3,c=4,a=2)

Q What do you mean by scope of variables?

Ans Scope means –to which extent a code or data would be known or accessed.

There are two types of variables with the view of scope.

1. **Global variable/Scope**- Name declared in main program . It is usable inside the whole program.	1. **Local variable/Scope**- Name declared in function body. It is usable within the function.
e.g. def fun(a,b): s = a+b print(s) X=int(input("enter no1")) Y=int(input("enter no2")) fun(X,Y) #here X and Y are global variable	e.g. def fun(): A=10 B=20 return(A+B) C=fun() print(C) print(A) #here A and B are local variables and C is global variable

Naming Resolution- LEGB-(LOCAL, ENCLOSED, GLOBAL and BUILT-IN)

1. **Variable in global scope not in local scope**	1. **Variable neither in in local scope nor in global scope**
e.g def fun1(x,y):	e.g. def fun():

s=x+y print(num1) return s num1=100 num2=200 sm=fun1(num1,num2) print(sm)	print("hello",n) fun() # Output: Name error: name 'n' is not defined
3. Variable name in local scope as well as in global scope e.g. def fun(): a=10 print(a) a=5 print(a) fun() print(a) # output 5 10 5	**4. Using global variable inside local scope (this case is discouraged in programming)** e.g. def fun(): global a a=10 print(a) a=5 print(a) fun() print(a) # output 5 10 10

Q How do we create modules in Python?

Ans

Modules in Python are simply Python files with a .py extension. The name of the module will be the name of the file.

Q What are the different ways of importing modules in Python?

Ans

1. **Importing entire module**
2. **Importing selected function/object from a module**
3. **Importing all function/objects of a module**

importing entire module	**importing selected function/object from a module**	**importing all function/objects of a module**
syntax import module_name	syntax from module_name import function_name	syntax from module_name import *
e.g. import math print(math.sqrt(4))	e.g. from math import sqrt print(sqrt(4)) from math import sqrt,pow print(sqrt(4)) print(pow(3,2))	e.g. from math import * print(sqrt(4)) print(pow(3,2))

Q What is the use of _init.py file?

Ans

1. Each package in Python MUST contain a special file called __init__.py.
2. This file can be empty.
3. it specifies that the directory it contains is a Python package.
4. it can be imported the same way a module can be imported.

---

random module-

1. **random()** a) it returns a random number in range(0.0 - 1.0). b) In this lower range limit is inclusive. c) Returns floating point number.	e.g. import random print(random.random())
2. randint() a) it returns a random number in given range. b) both range limit are inclusive. **c)** returns integer number.	import random print(random.randint(4,20))
3. randrange() a) it returns a random number in given range. b) ending range not including. c) returns integer number	import random print(random.randrange(4,20)) print(random.randrange(4,20,3))

File Handling

Q what is the usage of file?

Ans File is created for permanent storage of data or that stores data in an application.

Q How many types of files supported by Python?

Ans 3 (text file, binary file and CSV file)

Q Why is it necessary to close a file?

Ans

1. **Closing a file releases valuable system resources.**
2. ***if our program is large and we are reading or writing multiple files that can take significant amount of resource on the system. If we keep opening new files carelessly, we could run out of resources.***
3. **close() breaks the link of file object**
4. **After using this method, an opened file will be closed and a closed file cannot be read or written any more.**
5. **Closing file is important**
6. **In case we forgot to close the file , Files are automatically closed at the end of the program,**
7. **it is a good practice to close file explicitly.**
8. **This can boost performance.**

Q Write the different ways to open a file

Ans

open()	**with statement**

file_object/file_handler = open(<file_name>, <access_mode>).	**with statement**- in this mode , no need to call close() function syntax: with open**(<file_name>, <access_mode>) as file_object/file_handler**
file_name = name of the file ,enclosed in double quotes. access_mode= It is also called file mode. Determines the what kind of operations can be performed with file,like read,write etc **If no mode is specified then the file will open in read mode.**	
e.g **f=open("abc.txt",'r')** **f.write("hello")** **f.close()**	**e.g.** **with open("abc.txt") as f:** **f.write("hello")**

Q purpose of read(n) method?

This method reads a string of size (here n) from the specified file and returns it. If size parameter is not given or a negative value is specified as size, it reads and returns up to the end of the file. At the end of the file, it returns an empty string

Q Name two important functions of CSV module which are used for reading and writing.

csv.reader() returns a reader object which iterates over lines of a CSV file

csv.writer() returns a writer object that converts the user's data into a delimited string. This string can later be used to write into CSV files using the writerow() or the writerows() function.

MySQL

Data types of SQL- Following are the most common data types of SQL.

1) NUMBER / INTEGER

2) CHAR

3) VARCHAR

4) DATE

5) DECIMAL

DDL	DML
Data definition language	Data manipulation language
Create Drop Alter	Insert Update Delete Select
Creating a Database-To create a database in RDBMS, create command is used. *Syntax,*	INSERT Statement To insert a new tuple(row or record) into a table is to use the insert statement

create database database-name;
Example
create database Test;

CREATE TABLE Command: Create table command is used to create a table in SQL.
Syntax :
CREATE TABLE tablename
(column_name data_type(size),
column_name2 data_type(size)….
);

e.g. create table student (rollno integer(2), name char(20), dob date);

Alter command is used for alteration of table structures. Various uses of alter command, such as,

- to add a column to existing table
- to rename any existing column
- to change datatype of any column or to modify its size.
- alter is also used to drop a column.

Example:
ALTER command- Add Column to existing Table
Using alter command we can add a column to an existing table.

Syntax,
alter table table-name
add(column-name datatype);
e.g.
alter table Student add(address char);

ALTER command-To Modify an existing Column
alter command is used to modify data type of an existing column .
Syntax:-
alter table table-name modify(column-name datatype);
e.g.
alter table Student modify(address varchar(30));

ALTER command- To Rename a column
Using alter command you can rename an existing column.

Syntax:-

(i) To insert records into specific columns
Syntax:
insert into table_name(column_name1, column_name2…)values (value1,value2….);

e.g. INSERT INTO student (rollno,name)VALUES(101,'Rohan');

(ii) **insert records in all the columns**
insert into table_name
values(value1,value2……);

e.g.INSERT INTO student
(VALUES(101,'Rohan','XI',400,'Jammu');

Update command –it is used to update a row of a table. syntax,

UPDATE table-name set column-name = value where condition;
e.g.
UPDATE Student set s_name='Abhi',age=17 where s_id=103;

Delete command

It is used to delete data(record) from a table.It can also be used with condition to delete a particular row.

(i) syntax:- **to Delete all Records from a Table**

DELETE from table-name;

Example

DELETE from Student;

(ii) syntax: **to Delete a particular Record from a Table**

DELETE from Student where s_id=103;

SELECT command

Select query is used to retrieve data from a tables. It is the most used SQL query. We can retrieve complete tables, or partial by mentioning conditions using WHERE clause.
Syntax :
(i) DISPLAY SPECIFIC COLUMNS
SELECT column-name1, column-name2, column-name3, column-name from table-name;

Example
SELECT s_id, s_name, age from Student;

(ii) to Select all Records from Table- A special character asterisk * is used to

alter table table-name change old-column-name new_ column-name;
e.g.
alter table Student change address Location;

The above command will rename address column to Location.

ALTER command -To Drop a Column
alter command is also used to drop columns also. Syntax:-

alter table table-name drop(column-name)

e.g.
alter table Student drop column (address);

DDL - Drop command

This command completely removes a table from database. This will also destroy the table structure. Syntax,

drop table table-name

Example

drop table Student;

To drop a database,

drop database Test;

address all the data(belonging to all columns) in a query. SELECT statement uses * character to retrieve all records from a table.

Example: SELECT * from student;

CONSTRAINTS-

Constraints: Constraints are the conditions that can be enforced on the attributes of a relation. The constraints come in play whenever we try to insert, delete or update a record in a relation. They are **used to ensure integrity of a relation**, hence named as integrity constraints.

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK
6. DEFAULT

Example:

Create table Fee
(RollNo integer(2) Foreign key (Rollno) references Student (Rollno),
Name char(20) Not null,
Amount integer(4),

i. **Not Null constraint** : It ensures that the column cannot contain a NULL value.

ii. **Unique constraint** : A candidate key is a combination of one or more columns, the value of which uniquely identifies each row of a table.

iii. **Primary Key** : It ensures two things :
(i) Unique identification of each row in the table.
(ii) No column that is part of the Primary Key constraint can contain a NULL value.

iv. **Foreign Key** : The foreign key designates a column or combination of columns as a foreign key and establishes its relationship with a primary key in different table.

Fee_Date date); Example: create table Employee (EmpNo integer(4) Primary Key, Name char(20) Not Null, Salary integer(6,2) check (salary > 0), DeptNo integer(3)); example: create table Employee (EmpNo integer(4) Primary Key, Name char(20) Not Null, Salary integer(6,2) check (salary > 0), DeptNo integer(3) default 0);	v. **Check Constraint** : Sometimes we may require that values in some of the columns of our table are to be within a certain range or they must satisfy certain conditions. vi. **Default Constraint** : The **DEFAULT constraint** is used to set a **default** value for a column. The **default** value will be added to all new records, if no other value is specified.

WHERE clause

Where clause is used to specify condition while retriving data from table. Where clause is used mostly with Select, Update and Delete query. If condition specified by where clause is true then only the result from table is returned.

Syntax

SELECT column-name1, column-name2, column-name3, column-nameN
from table-name
WHERE [condition];

Logical operator- AND,OR,NOT **AND operator-** AND to show true value if all the conditions are true EXAMPLE TO return records where salary is less than 10000 and age greater than 25. SELECT * from Emp WHERE salary < 10000 AND age > 25; --- **OR operator-** In this , atleast one condition from the conditions specified must be satisfied by any record to be in the result. Example To return records where either salary is greater than 10000 or age greater than 25. SELECT * from Emp WHERE salary > 10000 OR age > 25;	**Like clause- pattern matches** **Wildcard operators** - used in like clause. (i) Percent sign % : represents zero, one or more than one character. (ii) Underscore sign _ : represents only one character. Example of LIKE clause **To display all records where s_name starts with character 'A'.** SELECT * from Student where s_name like 'A%'; Example **To display all records from Student table where s_name contain 'd' as second character.** SELECT * from Student where s_name like '_d%';
Relational Operator (comparison) **>, <, >=, <=, <> (not equal to) =(equal to)**	**IN- used to show the records from a LIST**

	Display all records of those employees whose belong to mumbai,delhi,jaipur only Select * from emp where city in ('mumbai','delhi','jaipur');
BETWEEN- show records within range **Display records whose salary between 2000 to 3000** select * from emp where sal between 2000 and 3000;	

Aggregrate Functions-These functions return a single value after calculating from a group of values. **frequently used Aggregrate functions.** Avg(), Sum(), max(), min(), count(column_name),count(distinct) **count(column name**)- Count returns the number of rows present in the table either based on some condition or without condition. **COUNT(distinct)** SELECT COUNT(distinct salary) from emp;	**Distinct keyword- it** is used with Select statement to retrieve unique values from the table. Distinct removes all the duplicate records while retrieving from database. Syntax : SELECT distinct column-name from table-name; Example **To display only the unique salary from Emp table** select distinct salary from Emp;

HAVING Clause

It is used to give more precise condition for a statement. It is used to mention condition in Group based SQL functions, just like WHERE clause.

Syntax:

```
select column_name, function(column_name)
FROM table_name
WHERE column_name condition
GROUP BY column_name
HAVING function(column_name) condition;
```

Consider the following Sale table.

Oid	order_name	previous_balance	customer

To find the customer whose previous_balance sum is more than 3000.

```
SELECT *
from sale
group by customer
having sum(previous_balance) > 3000;
```

Order By Clause- arrange or sort data To sort data in descending order DESC keyword	**Group By Clause-** it is used to group the results of a SELECT query based on one or more columns

Syntax : SELECT column-list|* from table-name order by asc / desc; **To display all records in ascending order of the salary.** SELECT * from Emp order by salary; **To display all records in descending order of the salary.** SELECT * from Emp order by salary DESC;	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE condition GROUP BY column_name; **To find name and age of employees grouped by their salaries** Example SELECT name, age from Emp group by salary; --- **Group by in a Statement with WHERE clause** select name, max(salary) from Emp where age > 25 group by salary;

MySQL Connectivity

```
import mysql.connector as m
# Open database connection
db = m.connect(host="localhost",user="root",passwd="1234")

# prepare a cursor object using cursor() method
cursor = db.cursor()

# execute SQL query using execute() method.
cursor.execute("show databases")            # write any sql related command in execute function

# Fetch a first three rows using fetchmany() method.
data = cursor.fetchmany(3)
for i in data:
        print (i)

# disconnect from server
db.close()
```

To fetch some useful information from the database, can use either
fetchone() method to fetch single record
fetchall() method to fetch multiple values from a database table.
fetchmany()- to fetch limited no of records

rowcount – it returns the number of rows using execute() method

Once a database connection is established, we are ready to create tables or records into the database tables using **execute** method of the created cursor.

Click on the below link to know:
DIFFERENCE BETWEEN VARIOUS TOPICS/CONCEPTS