UNIT-1 Review of Class-11

Python features.

(1) Interpreter based Programming language → Line by line execution of source code.
(2) Free & open source :→ Source is available free of cost. Free to use for commercial purpose.
(3) Portable → same code can be used for different machines.
(4) Object Oriented Support → support both OOPs & Procedural.
(5) Extensible → Python code can be written in other language.
(6) Dynamically typed → Variable datatype can be decided at runtime.
(7) Robust standard library → Extensive standard library available for anyone to use.
(8) Easy to code & Read → Simple syntax, indented blocks make it easy to read & code.

CODING MODES IN PYTHON

Interactive mode

- Used when user want to run one single line or block of code.
- Command typed at IDL prompt. & Execute when ↳ pressed.

Script mode

- Used when bunch of command in file.
- Runs commands sequentially.

COMMENTS

- Statements ignored by interpreter,
- Used to explain code.
- Make code more understandable.

(#) → for single line comment
This is comment.

(ee eeee) → Trinple quotes for multiple lines.

TOKENS

(1) Smallest individual unit.
(2) All statement & instruction built by tokens.

TYPES:

- **Keyword** — Reserve words / cannot used as identifier. 35 keywords in python.
- **Identifier** — Name given to variable, function, class etc
- **Literals** — values stored in program memory & often refered by an identifier.
  - **String Literal** — (', ", ''', """, \, \n, \t, \b) Newline, Tab, Backspace
  - **Numerical Literals** (int, float, complex).
  - **Operators** — Arithmatic, Assignment, Relational, Logical

Flow of control in python

(1) Python if, elif, & else blocked

```
a = int(input())
b = int(input())
c = int(input())
if a>b and a>c:
    print(a)
elif b>c:
    print(b)
else:
    print(c)
```

(2) For loop

```
sum=0
for i in range(1,11):
    sum = sum+i
print(sum)
```

(3) while

```
n = int(input())
for i in range(2, n//2+1):
    if n%i==0
        print("Not Prime")
        break
    else:
        print("Prime")
```

PYTHON STRINS

"Set of character etc enclosed in Single Quote, Double Quote or triple quotes"

Blank

```
S1 = " "
S2 = Str()
```

BASIC

```
S1 = "HELLO"
S2 = "12345"
```

ACCEPTING

```
S = input("Enter Name-")
```

STRING CONCATENATION

Concatination

```
S = "Hello"
t = "world"
n = S + " " + t
print(n)
```

output = Hello world.

Replication

```
S = "New"
t = S * 5
print(t)
```

output = NewNewNewNewNew.

Indexing

0	1	2	3	4	5
P	Y	T	H	O	N
-6	-5	-4	-3	-2	-1

→ Forward Index

→ Backward Index

Slicing

Str [Start : STOP : STEP)

```
S = "METAMORPHOSIS"
print(S[0:5])      → METAM
print(S[2:8:2])    → TMR
print(S[::-1])     → SISOHPROMATEM
print(S[-6:-2])    → PHOS
print(S[-1:7:-2])  → SSH
```

Traversal

```
S = "PYTHON"
for x in S:
    print(x)
```

→ PYTHON.

```
S = "PYTHON"
for i in range(len(S)):
    print(S[i])
```

→ PYTHON

STRING FUNCTIONS

(i)

Name	Desc.	Example.
isalpha().	Returns True if all character are alphabet.	S="PYTHON" Print(s.isalpha()) → True
isalnum()	Returns True if all character are alphabet or Numbers.	S="PYTHON3" Print(S.isalnum()) ⇒ True.
isdigit()	Returns True if all character are digits.	S="1234" print(S.isdigit()) ⇒ True.
isspace()	Returns True if there is space in string.	S="Python Hai" print(S.isspace()) → True.
isupper()	Returns True if all characters in uppercase.	S="PYTHON" print(S.isupper()) ⇒ True.
count()	Count occurence of particular character in string. S="Magic Moments Man" Print(S.count("M")) = 4 Print(S.count("M",2)) = 3 print(S.count("M",2,10)) = 2	S="MagicMoment" =print(S.count("M") 3.
index()	Returns index number of given chacters. print(S.index("M")) ⇒ 0 Print(S.index("M",2)) ⇒ 6.	
split()	Returns a list with sequence of substring, by eliminating all space. S="We love Python" print(S.split()) ⇒ print(S.split("E")) ⇒	['We','Love','Python'] ['W','LOV','Python']

Python List

1. **Empty list**

```
l1 = []
l2 = list()
```

2. **Homogenous list**

```
l1 = [10, 20, 30, 40]
l2 = ['A', 'B', 'C']
l3 = [True, false, True]
```

3. **Hetrogenous list**

```
l1 = [10, True, "Arun"].
```

4. **Nested list**

```
l2 = [[10,20,30],[5,10,15]]
```

5. **Acceepting a list from user →**

```
l1 = eval(input("Enter list numbers"))
```

Enter list numbers 10, 20, 30, 40.

6. **Indexing of list**

```
data = [10,20,30,40]
data[3] = 100
print(data)
```

10, 20, 30, 100

7. **Slicing**

```
Data = [1,2,5,6]
data[2:2] = [3,4]
print(data)
```

⇒ [1,2,3,4,5,6] → Not Removing any data Only Inserting at given Place.

```
data = [1,2,5,6]
data[2:3] = [3,4]
print(data)
```

⇒ [1,2,3,4,6] ⇒ Remove 5 from list & insert New from given Place.

8. **Del Command** used to delete one or more values from list.

```
data = [10,20,30,40,50]
del data[3]
print(data)
```

⇒ [10, 20, 30, 50]

```
data = [10, 20, 30, 40, 50]
del data[2:4]
print(data)
```

⇒ [10, 20, 50]

PYTHON LIST METHOD

Function	Use	Example.
len()	Returns number of element in list	Print (len(S))
max()	Returns maximum value of list	Print (max(S))
min()	Returns minimum value of list	Print (min(S))
Sorted()	Returns ordered list of values.	Print (Sorted(S)) for Ascending Order Print (Sorted, S, reverse= True)) for Descending Order.
clear()	Remove all element of list	S. clear() Print (S)
copy()	Create copy of existing list.	P = S.copy() Print (P)
append()	Add element at end of list	S. append (100)
extend()	append list of to another list.	S. extend [60, 70]
insert()	insert element at given index.	S. insert (4, 100) insert 100 @ 4th index.
Pop()	Remove last element from list.	X = S. pop ()
Pop (index)	Remove given index from list.	X = S. pop (3)
Remove (element)	Remove given value from list.	S. remove (40)
count()	Counts, given value no. of element in list.	S = [1, 1, 2, 2, 2, 3, 2, 4, 2] Print (S. count (2)) ⇒ 4.

PYTHON DICTONARY

(1) Collection of key values pair enclosed in {}
(2) un-ordered.
(3) Keys are immutable (number, string, tuple)
(4) Values are mutable.

```
d = {}      # Empty Dictionary
e = dict()  # Empty Dictionary
f = {'A':10, 'B':20, 'C':30}
g = {100:'A', 200:'B', 300:'C'}      } Dictionary with Records
h = {'Rollno':1001, 'Name':'Ravi', 'Marks':[10,20,30]}
        # Dictionary as tuple.
i = dict([('A',10),('B',20),('C',30)])
    print(i)
{'A':10, 'B':20, 'C':30}
```

How to display value ⇒ Dictname[key]

```
f = {'A':10, 'B':20, 'C':30}
print(f['B'])
⇒ 20
```

Add values to Dict ⇒ Dictname[key] = value.

```
f['D'] = 100
print(f)
{'A':10, 'B':20, 'C':30, 'D':100}.
f['A'] = 200   # changes value of A to 100.
```

Example	Output.
f = {'A':10, 'B':20, 'C':30}	
print(f.get('B'))	⇒ 20
print(f.keys())	⇒ dict_keys(['A','B','C'])
print(f.values())	⇒ dict_values([10,20,30])
print(f.items())	⇒ dict_items([('A':10),('B':20),('C':30)])
f.update({'D':40,'E':50})	⇒ Add two items at end of list.
X = f.pop('B')	⇒ Remove B from list.
X = f.popitem()	⇒ Remove last value from list.

Python Tuples

1. Collection of objects enclosed in ().
2. Python tuples are immutable & ordered.
3. Indexed like list & strings.
4. May contain Heterogenous elements.
5. Tuples can be nested.

```
t = ()          # empty tuple
u = tuple()     # empty tuple.
                # tuple may not have ()
m = 10,
n = (10,20)
x = ((10,20,30),(40,50,60))   # Nested tuples.
```

Python Functions:

- Built in function. min, max, len etc
- User defined functions.

Advantage → Reduce the size of program / Improve reusability of code.

Syntax

```
def << name of the >> (formal Parameter):
    tab indentation
        code
```

eg:-

```
def findbig(a,b):
    if(a>b):
        return a
    else:
        return b
x,y = 5,10
bigval = findbig(x,y)
```

TEXT FILE HANDLING

file is a sequence of bytes on the disk/permanent storage where a group of related data is stored. File Handling enables us to create, update, read, & delete.

STEPS

1. open a file.
2. perform operation (Read / write) or processing.
3. close file.

Type of file
- Text file
- Binary file
- CSV file.

Opening file file_object = open("filename.txt", mode)

mode r — Read r+ — Read/write w — write w+ — write/Read
a — append a+ — Append + Read

Closing file file object = file_object.close().

```
f = open("focus.txt",'r')
S = f.read(11)
print(S)
f.close()
```

⇒ Mindfulness

Note Text file contains → Mindfulness, cognitive training, and a healthy lifestyle may help sharpen your focus.

```
f = open("focust.txt",'r')
S = f.readline()
print(S)
f.close()
```

O/P Mindfulness, cognitive training, & a healthy - - - - -

Binary File Handling Python

Binary files store data in the binary format which is understandable by the machine. So when we open binary file in our machine, it decodes the data & display it in human readable format.

File Modes

- rb → Read only
- wb → write only
- ab → append
- rbt → Read & write
- wbt → write & Read
- abt → write & Read.

Writing data to a Binary file

Pickle is a special python Package that is used to generate data in binary format.

Methods of Pickle

load() , dump()
- load() → Read
- dump() → Write.

Pickling is the process whereby a python object hierarchy is converted into a byte stream.

Unpickling A byte stream is converted into object hierarchy.

```
import pickle
pickle.dump(list_name, file-object)
pickle.load(file-object)
```

eg:- Write in Binary file

```
import pickle
list = []          #empty list
while True:
    roll = Input("Enter Roll No -")
    name = input('Enter Name -')
student = {'roll':roll, "name":name}   #create dictionary
    list.append(student)   #add record.
    choice = Input("Want to add More (Y/N)?")
        if (choice=='n'):
            break
file=open("student.dat", "wb").
    pickle.dump(list, file)
        file.close()
```

Read data from binary file

```
import pickle
file = open("student.dat", "rb")
list = pickle.load(file)
print(list)
file.close()
```

UPDATING RECORD IN BINARY FILE:

```
import pickle
roll = Input("Enter Rollno. for updatl =")
file = open("student.dat", "rb+")
list = pickle.load(file)
found = 0
lst = []
for x in list:
    if roll in x['roll']:
        found = 1
        x['name'] = input("Enter New Name -")
    lst.append(x)
if found == 1
    file.seek(0)
    pickle.dump(lst, file)
    print("Record updated")
else:
    print("Rollno not exist")
file.close()
```

Searching of record in binary file.

```
import pickle.
roll = input ("Enter roll no to search — ")
  file = open('student.dat', "rb")
  list = pickle.load(file)
    file.close()
  for x in list:
        if roll in x['roll']:
            Print ("Name of student : ", x['name'])
            break
        else:
        print ("Record Not found")
```

CSV FILES

Comma Seperated

CSV file operations in python file in the CSV format can be imported & exported from program that store data in tables, eg:- Excel.

Advantages of CSV files:

(1) faster to Handle.
(2) Smaller in size.
(3) Easy to generate & import onto spreadsheet / Dbase.
(4) Human readable & easy to edit.
(5) Simple to implement & Parse.
(6) Processed by almost all existing applications.

CSV file operation in Python:

(1) For working with CSV files in python, there is an inbuilt module called CSV.
used to Read / write tabular data in CSV format.

(2) To perform Read / write operation in python we just import CSV module.
It handle CSV files correctly regardless of OS.

```
import CSV
file = open("Data.CSV", "w", newline='')
file.write("Name, Age, City \n")
file.write("Ram, 23, Delhi \n")
file.write("Hari, 56, Haryana \n")
file.close()
```

Reading from CSV file.

* Reading from a CSV file is done using the reader object.

The CSV file is opened in As text file in Python Buit-in open() which return file object.

* The reader object is an iterable that gives us access to each line of the CSV fileas a list of fields.

Next() to read next line.

```
import CSV
f = open("c:\data\student.csv","r")
csv_reader = csv.reader(f)
print("Content of Student file are-")

for row in csv_reader:
    print(row)

f.close()
```

Write in CSV file

csvwriter.writerow(row) → for single row
csvwriter.writerows(rows) → for multiple rows.

```
import CSV

datas = ['Name', 'Age', 'Grade']
datamul = [['Ram', "23", 'A'],['Shyam', 46, 'B']]
with open('Student.CSV', 'w', newline='') as file:

    writer.CSV write(file)
    writer.writerow(datas)
    writer.writerows(datamul)

file.close()
```

Q1 Write a python Program that performs the following task:-

(a) Open a CSV file named "inventory.csv" in read mode. using csv.reader() object.

(b) Iterates through each row in CSV file.

(c) Check if Quantity of each item is less than 10.of so, appends the itemname to list named low_stock_items.

(d) finally, print the list of low_stock_item.

```
import csv
low_stock_item = []
with open("inventory.csv", "r") as file:
    reader = csv.reader(file)
    next(reader)
    for row in reader:
        if int(row[1]) < 10:
            low_stock_item.append(row[0])
print("List of stock item low: ", low_stock_item)
file.close()
```

Q2 Open a CSV file named "sales.csv" in read mode using csv.reader() object & Calculate total revenue by sum. up the value in the third column (Price) of each row.

```
import csv
total_rev = 0
with open("sales.csv", "r") as file:
    reader = csv.reader(file)  next(reader)
    for row in reader:
        total_rev = total_rev + float(row[2])
    print("Total Revenue=", total_rev)
    file.close().
```

38 Write a python program that opens a CSV file named "Student.csv" in write mode using csv.writer() object. The program prompt student name & their grade & then write data into CSV file. Ensure error handling?

```
import csv
def write_student_data(file_name):
try:
with open("file_name", 'w', newline='') as file:
writer = csv.writer(file)
writer.writerow("Name", "Grade")
while True:
name = input("Enter Name of student -")
if name.lower() == 'exit':
break
grade = input("Enter grade =")
writer.writerow([name, grade])
except Exception as e:
print("Error:", e)
write_student_data("student.csv")
file.close()
```

38 Consider a list named nums which contains Random integers. write user defined () to perform:

(1) Pushbig() : Every no. more than 5 Push in Bigstack.

(2) Popbig() : Pop number from stack at end Display "Stack empty".

```
def Pushbig(Nums, Bignums):
    for N in Nums:
        if len(str(N)) >=5:
            Bignums.append(N)
def Popbig(Bignums)
while bignums:
    Print(Bignums())
else:
Print("Stack is Empty")
```