PYTHON

→ Python is a simple and easy to understand language which feels like reading simple english. This pseudo code nature of python makes it easy to learn and understandable by beginners.

→ Features of Python :-

→ Easy to understand → Less development time
→ Free and open source
→ ~~Free and open source~~ High level language.
→ portable → works on linux/windows/mac.

→ Python code :-

→ Print ("helloworld")
↓
Function

→ Modules :-

A module is a file containing code written by somebody else (usually) which can be imported and used in our programs.

PIP:

PIP is The package manager for python you can use pip to install a module on your system.

Ex:-
↳ PIP install flask
flask module.

⇒ Types of modules:-

Two types

1) Built in modules → Pre installed in python
Ex:- os, abc etc.
2) External modules → Need to install using PIP.
Ex:- tensorflow, flask, etc

⇒ Comments:-

comments are used to write something which the programmer does not want to execute.
Ex:-
↳ Can be used to mark author name, date etc.

Types of comments:-

Two types

1) Single line comments :- written using "#"
2) multi line comments :- written using """

use Python as calculator : using windows power shell.

we can use python as a calculator by typing "python" + ↵ Enter on the terminal

↳ This opens REPL
Read evaluate print loop.

Practice Questions

1) write a program to print twinkle twinkle little star poem in python.

Ans print ("'' twinkle Twinkle little star
How I wonder what u are
- - - - - -
- - - - - - ''')

we have to use '''' otherwise it will show error. if we are using more than one line we have to write the content between three inverted comas.

(''')

4) Install an external module and use it to perform an operation of your interest.

→ First we have to install the module by using PIP: Iamdown loading.

Playsound module

→ module

Ex:-1) PIP install flask we will write in the terminal. it will install directory

2) PIP install playsound.

↳ from playsound import playsound

↳ Playsound('C:\\Users\\Kiran\\Documents\\Python\\Johnny Depp Bgm Remix.mp3')

↓

path

5). Write a python program to print the contents of a directory using OS module. Search online for the function which does that.

→ Import os

Print(os.listdir())

↳ Tells how many directory are there in a perticular folder.

Variables and Datatypes

→ A variable is the name given to a memory location in a program

Ex: a = "Kiran"
b = 0.7
c = 0.7

variable → container to store a value.
Keyword → Reserved word in python
↘ def
Identifiers → class / function / variable name.

Datatypes:

→ Integers
→ Floating.
→ Boolean
→ string.
→ None

Ex:

To know the Type of variable

Print (Type (a))

Rules for defining a variable name:

→ A variable name can contain alphabets, digits and underscores.
→ A variable name can only start with an alphabet and underscore
→ A variable can't start with a digit
→ No space
→ Case sensetives.

Ex: Kiran, one7, _seven, seven_ etc.

operators in python

→ Arithmatic operator :→ +, −, *, /, -- etc

Ex:
5+2 = 7
5−2 = 3
5*2 = 10
5/2 = 2.5

→ Assignment operator → =, +=, -= etc

Ex:
a = 20
a+=20
o/p : 40

a = 20
a-=20
o/p : 0

→ comparision operator :→ ==, >, >=, <, <=, != --etc

Ex:
b = (5>2)
print(b)
o/p = True

b = (5<2)
print(b)
o/p = False

b = (5!=2)
print(b)
o/p = True

→ Logical operator :→ and, or, not.

ex:
bool 1 = True
bool 2 = False
print (bool 1 and bool 2)
print (bool 2 or bool 2)
print (bool not bool 2)

o/p :- False
True

type() function and Typecasting :

type function is used to find the data type of a given variable in Python.

Ex:

→ a = 31
type (a) ⇒ class <int>

→ a = "31"
type(a) ⇒ class <str>

A number can be converted into a string and vice versa (if possible)

There are many functions to convert on data type into another.

Ex:
str(31) ⇒ "31" → integer to string conversion
int("32") ⇒ 32 → string to int conversion.
float(32) ⇒ 32.0 → int to float conversion

--- and so on :

Here "31" is string literal and 31 is a numeric literal

⇒ input() function.

This function allows the user to take input from the keyboard as a string.

Ex:

```
a = input("Enter your name:").
```

if a is "Kiran" The user entered "Kiran".

output: Enter your name: Kiran

It is important to note that the output of input is always a string (even if the number is entered)

Practice set

Ex: 2) `a = input("Enter number")`
`a = int(a)`
`print(type(a))`

output: Enter number: 10 <class "int">

1) write a program to add two numbers.

```
a = 5
b = 3
c = a + b.
print("The sum of two numbers a and b is", 5+3)

print(type(c))
```

Output:

The sum of two numbers a and b is 8.

<class 'int'>

2) Write a python program to add two to ~~numbers~~ find remainder when a number is divided by 2.

Ans
```
a = 5
b = 3
c = a % b
print("the sum of two is
      The Reminder is ", 5 % 3)
```

output : The reminder is 2

3) Check The type of The variable assigned using input() function

Ans
```
a = input("Enter number : ")
a = int(a)
print(type(a))
```

output : Enter number : 15

<class 'int'>

4) Use comparision operators to find out whether a given variable a is greater than b or not Take a = 34 and b = 80.

Ans. a = 34
b = 20.

Print (a > b)

output: False.

5) Write a python to find average of two numbers entered by the users.

Ans. a = input ("Enter the first number: ")
b = input ("Enter the second number: ")
a = int (a)
b = int (b)
avg = (a+b)/2
Print ("The average value is :", avg)

output:
enter the first number: 10
enter the second number: 20.
The average value is 15

6) Write a python program to calculate Square of a number entered by the user.

Ans. a = input ("enter the number: ")
a = int (a)
Print (a**a)

output: enter the number: 5 3125

2)

A string is a sequence of characters enclosed in quotes.

We can primarily, write a string in Three ways

Three ways

1) Single quated strings → a = 'kiran'
2) Double quoted strings → b = "kiran"
3) Triple quoted strings → c = """kiran"""

String slicing:-

A string in python can be sliced for getting a part of the string.

Consider the following string :-

name = "kiran"

k	i	r	a	n
0	1	2	3	4
-5	-4	-3	-2	-1

⇒ Length = 5

Ex:- print(name[0:3]) ⇒ o/p → kir

The index in a string start from '0' to (length-1) in python. In order to slice a string, we use the following Syntax.

Syntax:-

a = name[ind_start : ind_end].

first index included — last index is not included.

a[0:3] returns "kir" → characters from 0 to 3

a[1:3] returns "ir" → characters from 1 to 3

→ Negative Indexing:-

negative indices can also be used as shown in the figure above. -1 correspond to the (length-1) index, -2 to (length-2)

Ex:-

a = "kiran kumar"

Print(a[-1])

output:- r.

→ KIRAN KUMAR
0 1 2 3 4 5 6 7 8 9
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1

a[1, 6:2]

I A,

→ slicing with skip value

we can provide a skip value as a part of our slice like is.

name = "kirankumar"

name[1:6:2]

O/p. → "iaK"

other advanced slicing techiques:-

name = "kirankumar"

name[:7] → name[0:7] → "kiranku"

name[0:] → name[0:7] → "kiranku"

=> String functions

1) len() function -> This functions returns the length of the string

len("kiran") -> returns 5

2) String.endswith("ran") -> This function tells whether the variable string ends with the string "ran" (or) not. if string is "kiran", it returns true for "ran" since kiran ends with ran

print(story.endwith("notes"))

3) String.count("c") -> counts the total number of occurence of any character

print(story.count("c"))

4) String.capitalize() -> This function capitalizes the first charater of a given string. print(story.capitalize())

5) String.find(word) -> This fiinction finds a word and returns the index of first occurence of that word in the string.

print(story.find("upon"))

6) String.replace(oldword, newword) -> This function replaces the oldword with newword in the entire string.

print(story.replace("kiran", "code with kiran"))

print (story.replace ("kiran", "Alekya")

⇒ Escape Sequence characters :-

→ Sequence of characters after backslash '\'

Escape seq. characters

→ Escape sequence character comprises of more than one characters but represents one character when used within the strings.

Ex :- \n , \t , \' , \\ etc.

↓ newline | ↓ Tab | ↓ single quote | ↘ backslash.

Practice set :-

1) write a python program to display a user entered name followed by Good After noon using input ( ). function.

Ans. name = input ("enter your name").
print ("good afternoon " + name).

Output :- good afternoon Kiran.

2) write a program to fill in a letter template given below with name and date

```
letter = '''Dear <|NAME|>,
You are selected!
<|DATE|>'''
```

Ans.

```
letter = '''Dear <|NAME|>,
Greetings from ABC coding house. I am happy to tell you are selected!
Date : <|DATE|>
'''
name = input("Enter your name\n")
date = input("Enter Date\n")
letter = letter.replace("<|NAME|>", name)
letter = letter.replace("<|DATE|>", date)
print(letter)
```

O/P :-

3) write a program to detect double spaces in a string.

Ans
```
a = "This is a string with double  string"
doublespace = a.find("  ")
print(doublespace)
```

4) Replace the double spaces from problem 3 with single spaces.

Ans
```
a = "This is a string with double  string"
doubleSpace = a.replace("  "," ")
print(doublespace).
```

5) write a program to format the following letter using escape sequence characters.

```
letter = "Dear Kiran, This python course
    is nice. Thanks!".
```

Ans
```
letter = "Dear Kiran, \n This python course
    is nice! \n\t Thanks.
print(letter).
```

O/P:
```
Dear Kiran,
    This python course is nice!
        Thanks.
```

Lists and Tuples

Python lists are containers to store a set of values of any data type

Ex:-

friends = ["Apple", "Akash", "Rohan", 7, False]
- "Apple", "Akash", "Rohan" → string
- 7 → int
- False → Boolean

⇒ changing "0'3" value with '7'

```
list = [1, 2, 3, 4, 5]
list[0] = 7
print(list)
```

O/P:- [7, 2, 3, 4, 5].

⇒ List Indexing:-

- List can be indexed just like a string

```
L1 = [7, 9, "Harry"]
L1[0] => 7
L1[0:2] => [7, 9] → List slicing.
L1[4] => error
```

List methods

consider the following list:

LI = [1, 8, 7, 2, 21, 15]

1) LI.sort() : updates the list to [1, 2, 7, 8, 15, 21]
2) LI.reverse() : updates the list to [21, 15, 8, 7, 2, 1].
3) LI.append() : ~~updates the list to~~ add 8 at the end of the list.
4) LI.insert(3,8) : This will add 8 at 3 index
5) LI.pop(2) : will remove index "2"
6) LI.remove(21) : will remove 21 number

→ Tuples :

A tuple is a immutable data type in Python
↳ cannot change

a = ( ) → Empty type
a = (1,) → Tuple with only one element needs a coma
a = (1, 3, 7) → Tuple with more than one element.

→ once defined a tuple's elements cant be altered or manipulated.

Methods : (a = (1, 7, 1, 2)].

1) a.count(1) : a.count(1) will returns number of times '1' accurs in 'a'. | O(p : 2
2) a.index(1) : a.index(1) will Return the index of first occurence of 1 in 'a'

Practice Set :-

1) Write a program to store seven fruits in a list entered by the user.

Ans.
```
f1 = input("Enter Fruit number 1:")
f2 = input("Enter fruit number 2:")
f3 = input("Enter fruit number 3:")
f4 = input("Enter fruit number 4:")
f5 = input("Enter fruit number 5:")

mylist = [f1, f2, f3, f4, f5]
print(mylist)
```

o/p :- ['Kiran', 'Kumar', 'venkat', 'Srinu']

2) Write a program to accept marks of 6 students and display them in a sorted manner.

Ans.
```
m1 = int(input("Enter marks 1"))
m2 = int(input("Enter marks 2"))
m3 = int(input("Enter marks 3"))
m4 = int(input("Enter marks 4"))

marks = [m1, m2, m3, m4]
marks.sort()
print(marks)
```

3) check That a tuple cannot be changed in python.

Ans
```
a = (2, 4, 5, 3, 2)
a[0] = 25
```

o/p :- it cannot change in tuple

4) write a program to sum a list with 4 numbers.

Ans
```
a = [2, 4, 56, 2, 60]
print (a[0] + a[1] + a[2] + a[3])
```

(or)

```
print (sum (a))
```

5) write a program to count the number of zeros in the following tuple:-

Ans
```
a = (7, 0, 8, 0, 0, 9)
print (a.count(0))
```

page 9

Dictionary & Sets

Dictionary is a collection of key-value pairs

Syntax

```
a = {"key" : "value",
     "kiran" : "code",
     "marks" : "100",
     "list" : [1, 2, 9]}
```

a["key"] => prints "value"

a["list"] => prints [1, 2, 9]

Properties of Dictionary

- it is unordered
- it is mutable
- It is indexed.
- cannot contain duplicate keys.

Dictionary methods

considering the following dictionary.

```
a = {"name" : "kiran",
     "from" : "India",
     "marks" : [56, 61, 64, 80]}
```

1) a.items() :- returns a list of (key, value) tuples

2) a.keys() :- returns a list containing dictionary keys

3) a.update({"friend":"kumar"}) :- updates the dictionary with supplied.

4) a.get("name") : returns the value of the specified keys (and value is returned (eg. "Kiran" is returned here)

→ Programs :-

→ updatedict = {"Lovish": "Kiran", "Diya": "Friend"}.

a.update(updatedict)

print(a)

→

Sets in python

Set is a collection of non repetitive elements.

s = set() → empty set is written has
⇒ no repetitions allowed

s.add(1)
s.add(2).

(or) set = {1, 2}

Ex: a = {1, 3, 4, 5}

print(a)
Print(type(a)).

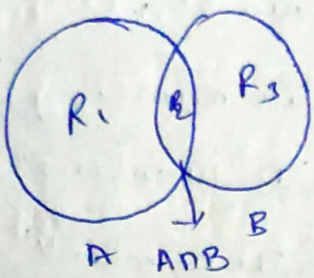
Properties

→ sets are unordered
→ sets are unindexed
→ There is no way to change items in sets.
→ sets cannot duplicate values

Methods : s = {1, 8, 2, 3}.

1) len(s) : Returns 4, The length of the set.
2) s.remove(8) : updates the set s and removes 8 from s.
3) s.pop() : Removes an arbitary element from the set and returns the element removed.
4) s.clear() : empties the sets
5) s.union({4, 11}) : Returns a new set with all items from both

6) s.intersection({8,11,3}): Returns a set which contains only items in both sets {8 4}.



$R_2 \Rightarrow A \cap B$

$R_1 + R_2 + R_3 = A \cup B$

$R_1 + R_3 \Rightarrow A \Delta B$

$R_1 \rightarrow A - B$

$R_3 \rightarrow B - A$.

Practice Set 1

1) Write a program to create a dictionary of Hindi words with values as their english translation. Provide user with an option to look it up.

Ans
```
dict = {"Patakaa": "cracker",
        "Bengan": "Bringal",
        "kalam": "pen"}.

Print("options are", dict.keys())
a = input("Enter the hindi word\n")
Print("The meaning of your word is:
      ", dict[a])
```

b) Write a program to input eight numbers from the user and display all the unique numbers

Ans
```
s1 = int(input("Enter number1"))
s2 = int(input("Enter number2"))
s3 = int(input("Enter number 3"))
s4 = int(input("Enter number 4"))
s5 = int(input("Enter number 5"))

set = {s1, s2, s3, s4, s5}

print(set)
```

c) Can we have a set with '18"(int/and '18"(~~set~~ str) as a values in it?

Ans.
```
Set = {18, "18", 18.0}

print(set)
print(Set[0]) X
```
it print all the values
it doesn't not support indexing.

d) what will be the length of following set S:

```
S = set()
S.add(20)
S.add(20.0)
S.add("20")
```

Length of S after these operations?

Ans:
```
S = {20, 20.0, "20"}
Print(len(S))
```

1) in this 20 and 20.0 has same values. coz it was showing the o/p has 2 values.

o/p len is 2

e) S = { }
what is the type type of S?

Ans
```
S = { }
print(type(S))
```

o/p it is dictionary type.

1). Create an empty dictionary Allow 4 friends to enter their favorite language as values and use keys as their names. Assume that the name are unique.

```
favLang = {}  # (dict declaration)
a = input("Enter your fav lang kiran")
b = input("Enter your fava lang punit")
c = input("Enter your fav lang Teja")
d = input("Enter your fav lang srinu")
```

Enter by the user fav lang.

```
favLang["kiran"] = a
favLang["punit"] = b
favLang["Teja"] = c
favLang["srinu"] = d
```

declaring keys to enter Language

Keys should be unique

```
print(favLang)
o/p: {"kiran": "Python", "punit": "Java", "Teja": "C++", "srinu": "C"}
```

2) if names of 2 friends are same, what will happen to the program in problems?

Ans The code is same as above if 2 friends as same name. only one name will be printed. and Language will be updated which was latestly given. Lang.

4) can you change the values inside a list which is contained in set S.

S = {8, 7, 12, "Harry": [1, 2]}.

Ans: we can not store the list in the set concept.
it will show error.

Conditional Statement

→ if else and elif in python

⇒ if else and elif statements are a multiway decision. taken by our program due to certain conditions in our code

Syntax

```
if(condition1) ——→ if condition 1 is true
   print("yes").
elif(condition2) → if condition 2 is false
   print("NO")
else:                → otherwise
   Print("may be").
```

Ex:-

```
a=22
if(a>9):
   Print("greater")
else:
   Print("lesser")
```

o/p greater

Q
→ Write a program to print yes when the age entered by the user is greater than or equal to 18

Ans
```
a=int(input("Enter the age:"))
if(a>=18):
    print("user is greater")
else:
    print("user is less")
```

Relational operators :

Relational operators are used to evaluate conditions inside the if statements. Some examples of relational operators are :

`==` → equal to
`>=` → greater than equal to
`<=`, etc

Logical operators :

In python logical operators operate on conditional statements.

Ex :

AND → True if both operands are true else false
OR → True if at least one operand is true
NOT → inverts true to false & false to true

Ex1 AND

age = 45

if (age >30 and age < 54) :

print ("you can work with us").

O/p :- you can work with us.

45 is b/w 30 and 54 so (you can work with us will be printed)

OR

age = 59

if (age >= 30 and age < 54) :

print (" you can work with us")

else :

print (" you cannot work with us").

O/P you cannot work with us because both conditions are not satisfying.

elif clause

elif in Python
Statement ca
with a lot
followed by

if (

elif

Importa

→ There co
→ Last e
con

elif Ex
a
if

elif clause

elif in python means [else if]. An if statement can be chained together with a lot of these elif statements followed by an else statement.

```
if (condition 1):
    #code
elif (condition 2):
    #code
elif (condition 3):
    #code
else:
    #code
```

→ this ladder will stop once a condition in an if or elif is met.

Important notes:

→ There can be any number of elif statements.
→ Last else is executed only if all the conditions inside elifs fails.

elif Ex:-

```
a = 45
if (a > 3):
    print("The value of a is greater Than 3")
elif (a > 13):
    Print("The value of a is greater Than 13")
elif (a > 7):
```

Print ("The value of a is greater than 7")

else:

Print ("The value is not greater than 3 or 7").

O/P: The value of a is greater than 3.

if a = 2 Then the O/P will be. print

O/P: The value of is not greater than 3 (or) 7

→ IS and In in python

→ is

a = None ——→ it points the same object in the program.

if (a is none):

Print ("Yes")

else:

Print ("No").

Practice set

1) Write a program to find greatest of four numbers entered by the user.

Ans

```
a = int(input("Enter number 1 : "))
b = int(input("Enter number 2 : "))
c = int(input("Enter number 3 : "))
d = int(input("Enter number 4 : "))

if (a > d):
    f1 = a
else:
    f1 = d
if (b > c):
    f2 = b
else:
    f2 = c
if (f1 > f2):
    Print(str(f1) + " is greater")
else:
    Print(str(f2) + " is greater").
```

Write a program to find out whether a student is pass or fail, if it requires total 40% and at least 33% in each subject to pass. Assume 3 subjects and take marks as an I/P from user

Ans

```
Sub1 = int(input("Enter first sub marks:"))
Sub2 = int(input("Enter second sub marks"))
Sub3 = int(input("Enter Third sub marks"))
if( sub1 <33 or sub<33 or sub<33):
Print(" you are fail because you have
    less than 33% in one of the subjects").
elif ( sub1+sub2+sub3)/3 <40:
Print(" you are fail due to total percentage
    less than 40")
else:
Print(" congratulation ! you passed the exam").
```

3) A spam comment is defined as a text containing following keywords: "make a lot of money", "buynow", "subscribe this", "click this". write a program to detect these spam.

Ans
```
text = input("Enter the text \n")
if("make a lot of money" in text):
    spam = True
elif("buynow" in text):
    spam = True
elif("subscribe this" in text):
    spam = true
elif("click this" in text):
    spam = True
else:
    spam = False.
if(spam):
    print("This text is spam")
else:
    Print("This Text is not spam").
```

4) write a program to find whether a given username contains less than 10 characters or not.

Ans

```
name = input("Enter The name")
if(len(name) < 10):
Print("name has less than 10 characters")
else:
Print("name has more than 10 characters")
```

5) write a program which finds out whether a given name is present in a list or not.

Ans

```
List = ["Kiran", "Kumar", "Pollana"].
name = input("Enter your name :\n")
if name in List:
Print("name is in the list")
else:
Print("name is not in the List").
```

6) Write a program to calculate the grade of a student from his marks from the following scheme:

90 - 100 - Ex
80 - 90 - A
70 - 80 - B
60 - 70 - C
50 - 60 - D
< 50 - F

Ans

```
marks = int(input("Enter the marks"))
if(marks >= 90):
    grade = "Ex"
elif(marks >= 80):
    grade = "A"
elif(marks >= 70):
    grade = "B"
elif(marks >= 60):
    grad = "C"
elif(marks >= 50):
    grade = "D"
else:
    grade = "F"
print("your grade is"+grade)
```

Loops in python

→ Sometimes we want to repeat a set of statements in our program. for instance print 1 to 1000.

→ Loops make it easy for a programmer to tell the computer, which set of instructions to repeat and how!

Types of Loops:

Two types of loops.

1) while loop.
2) For loop.

while loop:

Syntax:

```
while condition:
    # Body of the loop.
```

⇒ This block keeps executing until the condition is true.

→ In while loops, The condition is checked first. if it evaluates to true, The Body of the loop is executed, otherwise not.

while loop :-

```
i=0
while i>10:
  print ("yes")
```

o/p yes
yes
yes
yes
10 times print.

(A)

```
i=0
while i>10:
  print ("yes" + str(i))
```

o/p :-
yes1
yes2
yes3
yes4
yes5
:
yes10.

Q) Print from 1-50 by using while loop.

Ans

```
i=1
while ~~i~~ = ~~50~~ 50:
  i = i+1   (start from 2)
  (print(i)
  → (start from 1)
```

% directly print from 2

~~for loop~~ Ex :- while

```
fruit = ["banana", "watermelon", "strobery",
                                  "mango"].
i=0.
while i >len(fruit):
  print (fruit[i]).
  i=i+1
```

⇒ if the condion never becomes false. the loop repeatedly getting executing.

⇒ For loop:

A for loop is used to iterate through a sequence like list, tuple or string. [iterable].

Syntax:

```
l = [1, 7, 8]
for item in l:
    print(item)
```

print.

Ex:

⇒ Printing the fruits by using ~~for~~ loop.

```
fruit = ["banana", "watermellon", "strobery",
         "mango"].

for item in fruits:
    print(item)
```

o/p:

banana
watermellon
strobery
mango.

⇒ Range function in python

→ The range function in python is used to generate a sequence of numbers.

→ we can also specify the start, stop and step-size as follows: [it is usually not used with range()]

Syntax: range(start, stop, step-size)

Ex: ①

```
for i in range(8):
    Print(i)
```

o/p: 0, 1, 2, 3 - - - - 7

②

```
for i in range(1, 8):
    Print(i)
```

o/p: 1, 2, 3, 4 - - 7

→ For With else:

```
for i in range(10):
    Print(i)
else:
    Print("This is inside else of for")
```

o/p
0
1
2
.
.
.
9
This is inside else of for

→ The break statement

→ "break" is used to come out of the loop when encountered it instructs the program to - Exit the loop now.

Ex:-

```
for i in range(10):
    print(i)
    if i == 5:
        break
    else:
        print(done)
```

break ⟹ it is used to break the loop.

This part will print if the full loop is executed other wise it will not print else part in this program.

The continue statement

"continue is used to stop the current iteration of the loop and continue with the next one it instructs the program to "skip" this iteration"

Ex:-

```
for i in range(10):
    if i == 5:
        continue
    print(i)
```

→ In this 5 value will not print.
→ after 5 it will print 6, 7, 8, 9. like this.

if we write print(i) above the if condition it will print 5 also.

⇒ Pass Statement

Pass is a null statement in python.

it instructs to "Do nothing"

Ex:

```
l=[1,7,8]
for item in l:
    Pass ⟶ without pass, the program
           will Throw an error.
```

Practice Set

1) Write a program to print multiplication table of a given number for loop.

Ans

```
num = int(input("Enter the number"))
for i in range(1,11):
    print(str(num) + " x " + str(i) + " = " +
          str(i*num))
```

(or) using 'f' string

```
Print(f"{num} x {i} = {num*i}")
```

O/P:

```
Enter the number 5
```

5 table will print

2) Write a program to greet all the persons names stored in a list l1 and which starts with 5

l1 = ["Harry", "sambha", "sachin", "srikanth"]

Ans

l1 = ["Harry", "sambha", "sachin", "srikanth"]

For name in l1
if name.startswith("s")
print ("Hello " + name)

o/p

Hello Sambha
Hello Sachin
Hello srikanth.

3). Attempt problem 1 using while loop

sum = int(input("Enter the number :"))
i = 1
while i >= 10:
i = i + 1
Print ("str(sum) + "x" + str(i) + "=" + str(sum x i)

5 x 1 = 5

4) write a program to find whether a given number is prime (or) not.

Ans
```
num = int(input("Enter the number: "))
Prime = True
for i in range(2, num):
    if(num % i == 0):
        print
        Prime = False
        break
if prime:
    Print("This number is prime")
else:
    Print("This number is not prime")
```

5) write a program to calculate the factorial of a given number using for loop.

Ans
```
Num = int(input("Enter the number: "))
factorial = 1
for i in range(1, num+1):
    factorial = factorial * i
Print(f"The factorial of {num} is {factorial}")
```

6) write a program to find the sum of first n natural numbers using while loop.

```
Sum = [2, 3, 4, 5]
i = Sum[0]
while i:
    i = i+1
Print(Sum[0] + Sum[1] + Sum[2] + Sum[3])
```

7) write a program to print the following star pattern

```
  *
 * **
* * * *
```
for n = 3

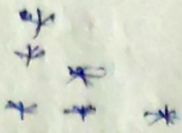
Ans

```
n = 3
for i in rang (3):
    Print(" " * (n-i-1), end = "")
    Print("*" * (2*i+1), end = "")
    Print(" " * (n-i-1))
```

for rows ← it is for print in a pattern.

```
  *
 * **
* * * *
```

8)



Ans n=4

```
for i in range(4):
    Print("*" * (i+1))
```

9) Write a program to print multiplication table of n using for loop in reversed order

Ans

```
num = int(input("Enter the number"))

for i in range(0,10).
    i=10-i
    Print("num","x","=",num*i)
```

Functions & Recursions

→ Functions :-

A function is a group of statements performing a specific task.

When a program gets bigger in size and it's complexity grows, it gets difficult for a programmer to keep track on which piece of code is doing what!

A function can be reused by the programmer in a given program any number of times.

Example and Syntax of a function

```
def funname( ):
    Print("Hello")
```

(space / indent)

→ This function can be called any number of times, any where in the program.

Function call :-

→ Whenever we want to call a function we put the name of the function followed by parentheis as follows!

funName( ) → This is called function call.

function definition :

The part containing the exact set of instructions which are executed during the function call.

function Ex:

```
def percent(marks):
    P = ((marks[0]+marks[1]+marks[2]+marks[3])/400)*100
      (or)
    P = ((sum/400)*100)
    return P
```

(def → keyword; marks → it takes the values of marks, it is optional; return ← keyword)

```
marks1 = [45, 78, 86, 77]
percent1 = percent(marks1)
marks2 = [75, 98, 88, 78]
percent2 = percent(marks2)
Print(percent1, percent2)
```

(OR)

```
def percent(marks):
    return (sum(marks)/400)*100
marks1 = [42, 72, 50, 38]
percent1 = percent(marks1)
marks2 = [32, 78, 90, 75]
Percent2 = percent(marks2)
Print(percent1, percent2)
```

Quick quiz

→ write a program to greet a user with "hood boy" using function

Ans
```
def greet(name):
    print("hood boy, " + name)
greet("kiran")
```
(def greet(name): — user defined function; print("hood boy, " + name) — function defination; greet("kiran") — function call)

→ Types of functions in python

Two types.

1) Built in functions → (len(), print(), range(), etc)
2) user defined function.

↓

The fun() function we defined is an example of user defined function.

→ function with arguments

A function can accept some values it can work with. we can put these values in the parentheris

A function can also return values as shown below:

```
def greet(name):
    gr = "Hello" + name
    return gr
a = greet("Kiran")
```
"Kiran" is passed to greet in name.

↳ a will now contain "Hello Kiran"

More than one value

```
def Sum(num1, num2)
    return num1 + num2
S = Sum(6, 32)
    ↳ calling function
Print(S).
```

⇒ default parameter value

→ we can have a value as default argument in a function.

→ if we specify name = "Stranger" in the line containing def, This value is used when no argument is passed.

Ex:

```
def greet(name = "stranger"):
    # function body
```

greet( ) → name will be "stranger" in function body (default)

greet("kiran") → name will be "kiran" in function body (passed)

Ex:- def greet (name = "Stranger"):
print ("good job" + name)
greet ( )

o/p. Stranger

=> Recursion:-

Recursion is a function which calls itself
It is used to directly use a mathematical formula as a function. (for example:-

Ex:- factorial (n) = n x factorial (n-1)

This function can be defined as follows:

```
def factorial(n):
    if i==0 or i==1 : → Base condition which doesn't call the function any further.
        return 1
    else:
        return n x factorial(n-1) → function calling itself
```

factorial program

```
n = 0
product = 1
for i in range (n):
    product = product * (i+1)
Print (product)
```

0 x 0+1
1 x 2 * 3 x 1

by using function

```
def factorial_iterate (n):
    product = 1
for i in range (n):
```

```
    product = product * (i+1)
  return product
f = factorial_iter(5)
print(f)
```

(or)

```
print(factorial_iter(5))
```

Notes

factorial(4)

↳ [function called]

4 × factorial(3)

4 × [3 × factorial(2)]

4 × 3 × [2 × factorial(1)]

4 × 3 × 2 × [1] [factorial returned]

Practice set !

1) write a program using function to find greatest of three numbers.

Ans

```
def max(num1, num2, num3):
    if (num1 > num2):
        if(num1 > num3):
            return num1
        else:
            return num3
```

```
else:
    if (num2 > num3):
        return num2
    else:
        return num3.

m = max(3,5,2)
print("The value of the max is " + str(m))
```

Calculate the fahrenheit Temperature

```
def farh(cell):
    return (cell * (9/5) + 32

c = 0
f = farh(c)
print("fahrenheit temp is " + str(f)
```

=>

File I/O

→ The random Access memory is volatile and all it's contents are lost once a program terminates. In order to persist the data forever, we use files.

→ A file is data stored in a storage device. A python program can talk to the file by reading content from it and writing content to it.

RAM – volatile memory
HDD – non volatile memory

Types of files

2 types of files

1) Text files (.txt, .c etc)
2) Binary files (.jpg, .dat, etc)

Python has a lot of functions for reading, updating and deleting files.

Opening a file

Python has an open() function for opening files. It takes 2 parameters: filename and -mode

Ex:-
open ("this.txt", "r")
↓ ↓ ↓
open is a builtin function | filename | mode of opening (read mode)

Ex program :-

filename

read method

f = open ("sample.txt", 'r') it is a default function
open the file in 'r' mode
data = f.read() → function
read its content
print (data)
print its content
f.close()
close the file → function

we can also specify the number of characters in read() function; f.read(2)
↳ read first two charaters

Other methods to read the file

we can also use f.readline() function to read on full line at a time.

Ex: f.readline()

modes of opening a file

r → open for reading
w - open for writing
a - open for append
+ - open for updata

"rb" will open for read in binary mode,
'rt' will open for read in text mode.

Writing File in python

In order to write to a file, we first open it in write or append mode after which, we use the python's f.write() method to write to the file!

Ex!

```
f=open("sample.txt",'w')
data=f.write("plear write This to the file")
print(data)
f.close()
f=open("another.txt",'w')
f.write("plear write This to the file")
f.close()
```

write

Add the content at the end of the line

```
f=open('another.txt','a')
f.write("plear write This to the file")
'I am appending')
f.close()
```

With Statement :-

The best way to open and close the file automatically is the with statement

Syntax: with open("this.txt") as f:
    f.read()
(Dont need to write f.close() as it is done automatically)

Ex:
with open("another.txt", 'r') as f:
    a = f.read()
with open("another.txt", 'w') as f:
    a = f.write("no")
print(a)