Lecture - 01 Variables & Data Types.

What is Python?

- Python is simple and easy
- Free and open source
- High Level language
- Developed by Guido van Rossum
- Portable

Our First program:

```
print ("Hello world")
```

↓
Function.

Python character set :-

- Letters - A to Z , a to z
- Digits - 0 to 9
- Special symbols - + = * / etc..
- White spaces - Blank space, tab, carriage return, newline, formfeed.
- Other characters - Python can process all ASCII and Unicode characters as part of data or literals

Variables:

- A variable is a name given to a memory location in a program.

var value

name = "chavan"

age = 20

price = 81.11

* Rules for Identifiers:

1. Identifier can be combitation of uppercase and lowercase letter, digits or an underscore(_). So
2. An identifier can not start with digit, so while variable1 is valid, 1variable is not valid.
3. We can't use special symbols like !, #, @ ;/, & etc. in our identifier.
4. Identifier can be of any length.

Data Types :-

- Integers
- String
- Float
- Boolean
- None (it represents non-values)
- List

NOTE :- To find which data type is ?

Eg:- print (type (variable_name))

Types of Operators:

An Operator is a symbol that performs a certain operation bwp between operands.

- Arithmetic Operators (+, -, *, /, %, **)
- Relational / comparsion Operators (==, !=, >, <, >=, <=)
- Assignment (=, +=, -=, *=, /=, %=, **=)
- Logical (not, and, or)
- Bitwise (&, |, ^, ~, <<, >>)
- Identity operator (is, is not).
- Membership operator (In, not in).

Comments in Python.

Single line Comment

```
"""
Multi Line Comment
"""
```

Type Conversion → Conversion (Automatically)
→ Casting (manually)

type Conversion

```
a = 2
b = 4.25
sum = a+b
print (sum)
```

o/p :- 6.25

type Casting

```
a, b = 1, "2"
c = int(b)
sum = a + c
print (sum)
```

o/p :- 3

Type casting is used to convert the datatype from one to another

Input() in python

Input() statement is used to accept values (using keyboard) from user.

```
input() # result for input() is always a str
int(input()) #int
float(input()) #float
```

Practice Questions.

program to input 2 nums & print their sum

```
num1 = int (input ("enter first num:"))
num2 = int (input ("enter second num :"))

sum = num 1 + num2
print ("The total is:", sum)
```

```
print ("sum =", num1+num2)
```

WAP to input side of a square & print it's area.

```
a = int (input ("Enter side sqr value :"))
print ("Area of a sqr is :", a*a)
```

WAP to input 2 floating point numbers & print their Average

```
num1 = float (input ("Enta first num :"))
num2 = float (input ("Enter second num :"))

print (" Avg = ", (num1 + num2) /2)
```

WAP to input 2 int numbers, a and b

Print "True" if a is greater than ~~b~~ or equal to b

If not print "False"

```
a = int (input ("Enter value : "))
b = int (input ("Enter value : "))
print (a>=b)
```

Lecture - 02 Strings & Conditional Statements

Strings - String is data type that stores a sequence of characters.

Basic Operations:

- Concatenation

  "hello" + "world" → "helloworld"

- Length of str

  len(str) (used to find the length of a string)

→ We use Escape Sequence character (some spl character) for formatting.

Eg:- Str1 = "This is a string. In we are creating it in python".

print (str1)

o/p:- This is a string
We are creating it in python

likewise, "\twe" for long tab ~~Sep~~ Space.

Indexing - To access characters of a string

A p n a - C o l l e g e

0 1 2 3 4 5 6 7 8 9 10 11

str = " Apna_ college "

str [0] is 'A', str [1] is 'p'....

str [0] = 'B' # Not allowed.

Eg:- str = "apna_ college"

ch = str [0]

print (ch)

o/p:- a

→ By using Indexing we can only access the characters, ~~no~~ can't modify or manipulate.

Slicing - Accessing parts of a string.

str [starting_idx : ending_idx] # ending idx is not ~~allowed~~ included.

str = "Apna College"

str[1:4] is "pna"

str [:4] is same as str [0:4]

str [1:] is same as str [1 : len(str)]

Slicing (Negative index)

A p p l e

-5 -4 -3 -2 -1

str = "Apple"

str [-3 : -1] is "pl".

String Functions :-

str = "I am a coder"

1) str. ends with ("er,") # returns true if string ends with substring.

Eg:- str = " I am a coder"

print (str. endswith ("er")) # True.

2) str. capitalize () # capitalizes the 1st char.

Eg:- str = " i am coder"

print (str. capitalize ()) # I am coder

3) str. replace (old, new) # replaces all occurences of old

Eg:- str = " I am coder"

print (str. replace ("e", "a"))

o/p:- I am codar

4) str. find (word) # returns 1st index of 1st occurrence

Eg:- str = " I am coder"

print (str. replace ("o"))

o/p:- 6 (It returns the index)

5) str.count ("am") # counts the occurance of the substring

Eg:- str = " i am Charan and studying python from youtube"

print (str.count ("studying"))

o/p:- 1.

Practice Questions.

WAP to input user's first name & print it's length.

```
name = str (input ("Enter name: "))
print (len (name))
```

WAP to find the Occurance of "$" in a string

```
str = " Hi, my name is charan Vamshi "
print (str.find ("$"))
```

Conditional statements.

if - elif - else (SYNTAX)

```
if (condition):
        statement 1
elif (condition):
        statement 2
else:
        statement n
```

Eg:-

```
light = "green"

if (light = "red"):
    print ("stop")
elif (light == "green"):
    print ("go")
else :
    print ("look")
```

o/p:- go.

NOTE* - Proper spacing in between the code is called Indentation.

practice Question

Grade students based on marks

marks >= 90 , grade = "A"

90 > marks >= 80 , grade = "B"

80 > marks >= 70 , grade = "C"

70 > marks , grade = "D".

```
int
marks = input ("Enter marks : "))

    if (marks >= 90)
        grade = "A"
    elif (marks >= 80 and marks < 90):
        grade = "B"
    elif (marks >= 70 and marks < 80):
        grade = "C"
    else :
        grade = "D"

print ("Grade of student :", grade).
```

Nesting - if

```
if (condition):
    if (condition 2):
        print ( )
```

Practice Questions

WAP to check if a number entered by the user is odd or even.

```
num = int (input ("Enter number : "))
if (num % 2 == 0):
    print ("even")
else :
    print ("odd")
```

WAP to find the greatest of 3 numbers entered by the user.

```
a = int (input ("Enter number : "))
b = int (input ("Enter number : "))
c = int (input ("Enter number : "))
```

```
if (a>=b and a>=c):
    print ("a")
elif (b>=c):
    print ("b")
else:
    print ("c").
```

WAP to check if a number is a multiple of 7 or NOT.

```
num = int (input ("Enter number : "))
if (num % 7 == 0):
    print ("Yes, it's multiple of 7")
else:
    print ("No, it's not multiple of 7")
```

Lecture - 03 Lists & Tuples.

Lists in Python.

A built in data type that stores set of values.

→ It can store elements of different types (integer, float, string, etc ...)

```
marks = [87, 64, 33, 95] # marks[0], marks[1]...

student = ["karan", 85, "Delhi"] # student[0], student[1]...

student[0] = "Arjun" # allowed in python

len(student) # returns length.
```

Example:-

```
marks = [94.4, 87.5, 95.2, 66.4, 44.1]

print(marks) # [94.4, 87.5, 95.2, 66.4, 44.1]

print(type(marks)) # list

print(marks[0]) # 94.4

print(marks[1]) # 87.5
```

NOTE* String → Immutable (cannot change index values)

list → Mutable (can change index values)

List Slicing

Similar to string slicing.

list_name [starting_idx : ending_idx] # ending idx is not inclusive

marks = [87, 64, 33, 95, 76]

marks [1:4] # [64, 33, 95]

marks [:4] is same as marks [0:4]

marks [1:] is same as marks [1: len (marks)]

marks [-3 : -1] is [33, 95]

List Methods / Functions

list = [2, 1, 3]

1) list.append (4) # adds one element at the end

[2, 1, 3, 4]

Eg:- list = [2, 1, 3]

list.append (4)

print (list)

o/p:- [2, 1, 3, 4]

2) list . sort() # sorts in ascending order [1,2,3]

Eg:- list = [2, 1, 3]

print (list. sort()) # None

print (list) # [1, 2, 3]

3) list. sort (reverse = True) #sort in descending order [3, 2,

Eg:- list = [2, 1, 3]

~~print~~

list . sort (reverse = True)

print (list). # [3, 2, 1]

4) list. reverse () #reverse list [3, 1, 2]

Eg:- list = ['a', 'd', 'e', 'f', 'c', 'b']

~~print~~

list. reverse ()

print (list)

o/p:- ['b', 'c', 'f', 'e', 'd', 'a']

5) list.insert (idx, el) # insert element at index

Eg:- list = [2,1,3]

list.insert (1,5)

print (list)

o/p: [2, 5, 1, 3]

6) list.remove (1) # removes first occurance of elem [2,3,1]

Eg: list = [2,1,3,1]

list.remove (1)

print (list)

7) list.pop (idx) # removes element at idx.

Eg:- list = [2, 1, 3, 1]

list.pop (2)

print (list)

o/p: [2, 1, 1]

Tuples in Python

- A built in data type that let' us create immutable sequence of values.

→ We use paranthesis () instead of [] in tuples.

```
tup = (87, 64, 33, 95, 76)   # tup[0], tup[1], ....
tup[0] = 43                  # NOT allowed in python.

tup1 = ()
tup2 = (1,)
tup3 = (1, 2, 3)
```

Eg:- tup = (1, 2, 3, 4) → main thing

print (tup)

O/p:- (1, 2, 3, 4)

Tuple Method.

tup = (2, 1, 3, 1)
0 1 2 3

1) tup.index (el) # returns index of 1st Occurance

tup.index (1) is 1

Eg:- tup = (1, 2, 3, 4)

print (tup.index (2))

o/p:- 1

2) tup.count (el) # counts total occurences

tup.count (1) is 2.

Eg:- tup = (1, 2, 3, 4)

print (tup.count (2))

o/p:- 1

Practice Questions

WAP to ask the user to enter names of their 3 favorite movies & store them in a list.

```
movies = []
movie1 = input ("enter movie1 : ")
movie2 = input ("Enter movie2 : ")
movie3 = input ("Enter movie3 : ")

movies.append (movie1)
movies.append (movie2)
movies.append (movie3)

print (movies)
```

(or)

```
movie1 = input ("Enter movie1 : ")
movie2 = input ("Enter movie2 : ")
movie3 = input ("Enter movie3 : ")

list = [movie1, movie2, movie3]
print (list).
```

WAP to check if a list contains a palindrome of elements. (Hint use copy() method).

```
list = ["m", "a", "d", "a", "m"]

copyofliist = list.copy()
copyoflist.reverse()

if (list == copyoflist):
    print ("Yes, it is a palindrome)
else:
    print ("No, it is not a palindrome)
```

WAP to count the number of students with the "A" grade in the following tuple.

["C", "D", "A", "A", "B", "B", "A"]

```
tup = ("c", "d", "a", "a", "b", "b", "a")
print (tup.count ("a"))
```

Store the above values in a list & sort them from "A" to "D".

```
Grade = ["C", "D", "A", "A", "B", "B", "A"]
Grade.sort()
print(Grade).
```

Lecture - 04 Dictionary & Set.

Dictionary in Python.

Dictionaries are used to store data values in key:value pairs.

They are unordered, mutable (changeable) & don't allow duplicate keys.

"key" : value

```
dict = {
    "name" : "shradha",
    "cgpa" : 9.6,
    "marks" : [98, 97, 96],
}
```

* dict["key"] = "value" # To assign or add new

Nested Dictionaries:

```
student = {
    "name": "charan",
    "Score": {
        "Chem": 98,
        "phy": 97,
        "math": 95
    }
}
```

print (student) # to print whole dictionary

print (student ["subjects"]) # to print specific dictionary

print (student ["subjects"] ["chem"]) # 98

Dictionary Methods.

1) myDict.keys () # returns all keys

2) myDict.values () # returns all values

3) myDict.items () # returns all (key, val) pairs as tuples

4) myDict.get ("key") # return the key according to value

5) myDict.update (newDict) # insert the specified items to the dictionary

Set in Python

- Set is the collection of the unordered items.
- Each element in the set must be unique & immutable.

nums = {1, 2, 3, 4}

set2 = {1, 2, 2, 2} # repeated element stored only one so it resolved to {1, 2}

null_set = set() # empty set syntax.

Set Methods

1) set.add(ele) # adds an element
2) set.remove(ele) # removes the element
3) set.clear() # empties the set
4) set.pop() # removes a random value.

NOTE*: Set → mutable

Set elements → immutable.

5) set.union(set2) # combing both set values & returns new.

Eg:- set1 = {1, 2, 3}

set2 = {2, 3, 4}

print (set1.union(set2))

o/p:- {1, 2, 3, 4}

6) set.intersection(set2) # combing common values & returns new.

Eg:- Set 1 = {1, 2, 3}

Set 2 = {2, 3, 4}

print (set1.intersection(set2))

o/p:- {2, 3}

Practice Questions

- Store following word meanings in a python dictionary :

  tabe : "a piece of furniture", "list of facts & figures"

  cat : "a small animal"

```
dict = {
    "table": ("a piece of furniture", List of facts & f
    "cat": "a small animal"
}
print (dict)
```

You are given a list of subjects for students. Assu one classroom is required for 1 subject. How many classrooms are needed by all students.

"python", java, c++, python, javascript, java

python, java, c++, c

```
subjects = {
    "python", "java", "c++", "python", "java script",
    "java", "python", "java", "c++", "c"
}

print (len (subjects))
```

WAP to enter marks of 3 subjects from the user and store them in a dictionary. Start with an empty dictionary & add one by one. Use subject name as key & marks as values

```
marks = {}

x = int (input ("enter phy : "))
marks.update ({"phy" : x})

x = int (input ("enter math : "))
marks.update ({"math" : x})

x = int (input ("enter chem : "))
marks.update ({"chem : x})

print (marks)
```

Figure out a way to store 9 & 9.0 as separate values in the set.

(you can take help of built-in data type).

```
values = {
        ("float", 9.0),
        ("int", 9)
}
print (values)
```

Lecture - 05 Loops

Loops in Python.

Loops are used to repeat instructions.

while loops

```
while condition:
    # some work
```

Eg:-

```
count = 1
while count <= 5:
    print ("hello")
    count += 1
```

Practice Questions:-

WAP to print numbers from 1 to 100

```
num = 1
while num <= 100:
    print (num)
    num + = 1
```

WAP to print numbers from 100 to 1

```
num = 100
while num >= 1 :
    print (num)
    num -= 1
```

WAP to print the multiplication table of a number.

```
i = 1
n = int (input ("Enter number : "))
while i <= 10 :
    print (n*i)
    i += 1
```

WAP to print elements of the following list using a loop

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

```
nums = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
idx = 0
while idx < len (nums):
    print (idx)
    idx += 1
```

WAP to Search for a number x in this tuple using a loop.

(1, 4, 9, 16, 25, 36, 49, 64, 81, 100)

```
nums = (1, 4, 9, 16, 25, 36, 49, 64, 81, 100)

x = 36

i = 0
while i < len (nums):
    if (nums [i] == x):
        print ("FOUND at idx", i)
    i += 1
```

Break & Continue.

Break: Used to terminate the loop when encountered.

Continue: Terminates execution in the current iteration & continues execution of the loop with the next iteration.

Eg:-

```
i = 0
while i <= 5:
    if (i == 3):
        i += 1
        continue # skip
    print (i)
    i += 1
```

o/p:-

```
0
1
2
4
5
```

For loop.

for loops are used for sequential traversal. For traversing list, string, tuples etc...

Syntax:-

```
for el in list:
    #some work.
```

1. Eg:-

```
nums = [1, 2, 3, 4, 5]

for val in nums:
    print (val)
```

2. Eg:-

```
str = "apna college"

for char in str:
    print (char)
```

o/p:-

```
a
p
n
a
c
o
l
l
e
g
e
```

for loop with else (optional)

Syntax:-

```
for el in list:
    # some work
else:
    # work when loop ends
```

Eg:-
```
for el in list:
    print (el)
else:
    print ("END")
```

Practice Questions

Print the elements of the following LiA using a loop:

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

```
list = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

for val in list:
    print (val)
```

Search for a number x in this tuple using loop

(1, 4, 9, 16, 25, 36, 49, 64, 81, 100)

```
tup = (1, 4, 9, 16, 25, 36, 49, 64, 81, 100)

x = 36

for val in tup :
    if (val == x):
        print ("found")
        break
    else:
        print ("Not found")
```

* range() Function

Range function returns a sequence of numbers, starting from 0 by defaulut, and increments by 1 (by default), and stops befor.

~~Syntax~~ range (start?, stop, step?)

Eg:- print (range (5))

O/p:- range (0, 5)

Eg:- for i in range(5):
    print(i)

o/p:- 0
1
2
3
4

- for i in range(10): # range (stop)
  print (i) o/p:- 0 1 2 3 4 5 6 7 8 9

- for i in range (2,10): # range (start, stop)
  print (i) o/p:- 2 3 4 5 6 7 8 9

- for i in range (2, 10, 2): # range (start, stop, step)
  print(i)
  o/p:- 2 4 6 8

Practice Questions:- (using for loop)

Print the elements of the following list using a loop:

[1,4,9,16,25,36,49,64,81,100]

```
list = [1,4,9,16,25,36,49,64,81,100]

for i in list:
    print(i)
```

WAP to search for a number x in this tuple using loop:

(1, 4, 9, 16, 25, 36, 49, 64, 81, 100)

```
tup = (1,4,9,16,25,36, 49, 64, 81, 100)
x= int (input ("Enter number : "))
for i in tup:
    if (x == i):
        print ("Found")
        break
    else:
        print ("Not found")
```

Practice Questions (using for & range())

Print numbers from 1 to 100.

```
for i in range (1,100):
    print (i)
```

Print numbers from 100 to 1

```
for i in range (100, 0, -1)
    print (i)
```

print the multiplication table of a number n.

```
n = int(input("Enter number :"))
for i in range (1,11):
    print (n*i)
```

Pass Statement :-

Pass is a null statement that does nothing. It is used as a placeholder for future code

eg:-
```
for i in range (5):
    pass
print ("some useful work")
```

o/p:- some useful work.

Practice Questions:-

WAP to find the sum of first n number. (using loop ~~while~~)

using for loop

```
n = 5
sum = 0
for i in range (1, n+1):
    sum += i
print (sum)
```

using while loop.

```
n = 7
sum = 0
i = 1
while i <= n:
    sum += i
    i += 1
print (sum)
```

WAP to find the factorial of first n numbers

```
n = 3
fact = 1
i = 1
while i <= n:
    fact *= i
    i += 1
print ("factorial =", fact)
```

Lecture - 06 Functions & Recursion

Functions in Python

Block of statements that perform a specific task.

Syntax :-

```
def func_name (param1, param2...):
    # Some work
    return val.

function_name (arg1, arg2..) #function call.
```

Eg:-

```
def cal_sum (a, b):
    sum = a + b
    print (sum)
    return sum

cal_sum (5, 10)
```

o/p:- 15

or

```
# function defination
def calc_sum (a, b): # parameters
    return a + b

sum = calc_sum (1,2) # function call ; arguments.
print (sum)
```

Types of Functions.

Built in Functions

- print()
- len()
- type()
- range()

User defined functi

→ Default Parameters:- Assigning a default
to parameter, which is used when no arg
is passed.

Practice Questions:-

WAF to print the length of a list. (list is
parameter)

```
cities = ["delhi", "noida", "pune", "hyd", "chenn
```

```
def cities_list()
    print(len(cities))

cities_list()
```

Write a Function to print the elements of a list in a single line (list is the parameter)

```
cities = ["hyd", "chennai", "banglore", "noida", "pune"]

def cities_list ():
    for item in cilites:
        print (item, end = " ")

cities_list ()
```

WAF to find the factorial of n. (n is the parameter)

```
def cal_fact (n):
    fact = 1
    for i in range (1, n+1):
        fact *= 1
    print (fact)

cal_fact (5)
```

O/p:- 120

WAF to convert USD to INR.

```
def converter (usd_val):
    inr_val = usd_val * 83
    print (usd_val, "USD =", inr_val, "INR")

converter (1)
```

O/p:- 1 USD = 83 INR.

Recursion :-

When a function calls itself repeatedly.

eg:-
```
def show (n):
    if (n == 0):
        return
    print(n)
    show (n-1)
show (5)
```

O/p:-
```
5
4
3
2
1
```

Practice Questions:-

Write a recursive function to calculate the sum of first n natural numbers.

```
def calc_sum (n):
    if (n == 0):
        return 0
    return calc_sum (n-1) + n

sum = calc_sum (10)
print (sum)
```

Lecture - 07 : File Input/Output

1

File I/O in Python

Python can be used to perform operations on a file (read & write data).

Types of all files.

1. Text Files : .txt , .docx , .log etc...
2. Binary Files : .mp4 , .mov , .png , .jpeg etc..

Open , read & close File

We have to open a file before reading or wri

```
f = open ("file-name", "mode")
```

sample.txt
demo.docx.

r: read
w: write ....

```
data = f.read()
f.close()
```

Eg:- f = open ("demo.txt", "r")

```
data = f.read()
print (data)
print (type(data))
f.close ()
```

Types of Modes:-

Character		Meaning.
'r'	→	open for reading (default)
'w'	→	open for writing, truncating the find file
'x'	→	Create a new file and open it for writing
'a'	→	Open for writing, appending to the end of the file if it exists
'b'	→	binary mode
't'	→	text mode (default)
'+'	→	open a disk file for updating (reading and writing)

Reading a file :-

```
data = f.read()          # reads entire file

data = f.readline()      # reads one line at a time
```

Writing to a file :-

```
f = open("demo.txt", "w")
f.write("this is a new line")   # overwrite the

f = open("demo.txt", "a")
f.write("this is a new line")   # adds to the file
```