DATA TYPES

int 0 ⇒ integers

str "Himanshu", 'Malik', '1' ⇒ string

bool True, False ⇒ Boolean suggest wether its TRUE/FALSE

float 0.23, 8.68 ⇒ represents decimal value

CREATE A VARIABLE

name = 'MALIK' [So, name is a vairable in which string is stored]

print(name)

>> Malik

name = 'Himanshu'

print(name)

>> Malik
Himanshu

* We cannot start a variable using integer

Eg:- 1name = '23' [Invalid]

* Basic Operators & Input :-

```
x = 2
print(x)
```

[These "()" contains argument. For example in this one arguments contains integer value.]

* print ('Hello, What is your name')
  name = input() [input() allows user to type the value]
  print (name)

* Operators (* + - /) ⇒ 4 Basic Operators

```
num1 = 45
num2 = 20
print(num1 + num2)
```

>> 65

So, we can do +, *, -, /

In Python there are more operates which we use in Maths

** ⇒ Exponent

// ⇒ Give whole number

% ⇒ Give us the remainder

*

```
num1 = 5
num2 = 2
num3 = num1 + num2
print(num3)
```

```
Print ("Guess a number: ")
num 1 = input()
print ("Guess one more: ")
num2 = input()
SUM = int(num2) + int(num2)
Print (SUM)
```

```
>>  44
    44
   =88
```

CONDITIONS :-

[< , > , == , !=] ⇒ Comparison Operator

print (2<3)

>> True

print (2<5)

print (2-3+4>5)

print (True == False)

>> True
False
False

[<=, >=]

(Lesser than equal to, greater than equal to)

* Decision IF/ELIF/ELSE:-

```
age = input ('Input your age:')
if age == 20:
   print ("Hey you are 20")
```

>> 12

End nothing gone print

if we put

16, then as well nothing will be printed

because we are putting the value as "string"

Correct, Remember these things:

```
age = input ("Enter your age: ")
if int(age) == 20:
      print("Hey You are 20 years old")
```

>> 20

=20

```
* age = input ("Enter Your age:")
  if int(age) > 20:
     print ("You are teen of ", age, " years old")
```

>> 12

= You are teen of 12 years old

```
age = input("Enter your age: ")
if int(age) >= 18:
    print("You are adult of", age, "years")
else:
    print("You are child", age, "Baby")
```

if we enter 20

>> You are adult of 20 years

if we enter 16

>> You are child 16 Baby

⇒ We can put "elif" as much as we want. But "else" put only once in a decision statement.

```
height = input("Enter UR height: ")
if float(height) >= 2
    Print("You cannot Ride")
elif float(height) <= 1
    Print("Still cannot Ride)
else:
    Print("Welcome to Zumanji")
```

if user enter 2

» You cannot Ride

if user enter 1

» Still cannot Ride

if user enter 1.8

» Welcome to Zumanji

* Chained Condition & Nested Statements:-

Chaine Condition consist of "AND, OR, NOT". Which means that giving more condition in one line.

AND ⇒

True and False ⇒ False
False and True ⇒ False
True and True ⇒ True
False and False ⇒ False

OR ⇒

True or False = True
False or True = True
True or True = True
False or False = False

NOT ⇒

True != False
False != True

it means it will reverse

* x = 2
y = 3

This is the example of Chained condition

```
if x == 2 and y-x == 3:
    print (" :C")
else:
    print (" C:")
```

⇒ C:

x = 2
y = 3

```
if x == 2:
    if y == 3:
        print("x is 2 & y is 3")
    else:
        print(x is not 2)
else:
    print(y is not 2)
```

so if x & y are true then,

» x is 2 & y is 3

~~if any one of the false then,~~

so if y is false then,

» x is not 2

so if x is false then,

» y is not 2

* FOR LOOPS :- [BASICS]

```
for x in range(0, 10, 2):  # (start, stop, step)
    print(x)
```

>>
```
0
2
4
6
8
```

So whats happening is

```
x += 2
 for
x = x+2
```

* While Loops :-

```
While condition == True:
    do this
```

So it means that it will check condition is true & do this. Then again it will check condition is True & do this. It will check unit the condition want be False.

So for that, we will add break

```
While condition == True:
    do this
    break
```

*

```
loop = True
while loop:
    name = input ('Enter Data: ')
    if name == '123':
        break
```

if we put "abcd"

>> Enter Data: abcd
Enter Data:
It will loop until we will break it by putting 123

Enter Data: 123

>>

* LISTS AND TUPLES :-

- fruits = ['apple', 'Banana', 3]

  ∴ The [] square brackets is called "lists" which contain different data types.

* fruits = ['apple', 'Banana', 3]
  print(fruits[1])

  >>
  Banana

* fruits = [1, 2, 3]
  fruits.append(4)
  print(fruits)

  ⇒ So the append will add that value you have given in the last place of Lists.

  >>
  1, 2, 3, 4

* fruits = [1, 2, 3]
  fruits[1] = 9
  print(fruits)

  >>
  [1, 9, 3]

* ITERATION BY ITEM :-

MORE ADVANCED FOR LOOPS BY Iterating through list.

*
```
fruits= ['apple', 'pear', 'Anar']

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

>>
```
apple
pear
anar
```

*
```
name = ['DJ', 'KJ', 'HJ', 'HM']

for naam in name:
    if naam == 'HM':
        print(naam)

else:
    print('nahi hn naam')
```

>>
```
nahi hn naam
nahi hn naam
nahi hn naam
HM
```

*
```
name = ['A', 'B', 'C', 'D']

for x in range(0,4):
    if name[x] == 'C':
        print(name(x))
    else:
        print('not a name')
```

>> not a name
not a name
C
not a name

⇒ We can use len function as well instead of giving value.

Example :-

```
name = ['A','B','C','D']
for x in range(len(name)):
    if name[x] == 'B':
    print(name[x])

    else:
        print("C:")
```

STRING METHODS

.strip()

Remove the spaces.

Eg:-

```
name = input("Enter Your name: ")
print(name.strip())
```

>>

Enter Your name: Himanshu Malik

→ HimanshuMalik

len()

Give the length of a string or a list/item

Eg:-

```
name= input("Enter: ")
Print(len(name))
```

>> Enter: Himanshu

8

.lower()

Convert all the Capital Letters into lower case

Eg:-

```
name = input("Enter: ")
print(name.lower())
```

>> Enter: HIMANSHU

himanshu

.upper()

Convert all the lower case letter into Capital letters.

Eg:-

```
Name = input("Enter: ")
print(name.upper())
```

>> Enter: himanshu

HIMANSHU

.split()

Convert the periods or break down we have given into list.

Eg:-

```
name= input("Enter: ")
print(name.split('.'))
```

>> Enter: Himanshu.Malik.Bye.Hi

['Himanshu','Malik','Bye','Hi']

* SLICE OPERATOR :-

fruits = [`apples', `pear', `orange']
text = `Hello i love Python'

* print (text [1:]) ⇒ So it will by default print at the last. Doesn't matter you know the counter not.

>> I love Python

So basically it is similar to for loop we put in range

(start , stop , step)

* print (text [0:8:1])

>> Hello i lov

Like using appendix we add somewhat in the end. But using Slice Operator we can add wherever we want to add.

Eg:-

```
fruits[0:0] = 'Banana'
print (fruits)
```

>> ['Banana', 'apples', 'pear', 'orange']

* FUNCTIONS

```
def addTwo(x):
    return x+2

def subtractTwo(number):
    return num-2
```

*
```
def addTwo(x):
    return x+2

newNumber = addTwo(7)
print(newNumber)
```

>> 7

*
```
def abcString(string):
    print(string)

abcstring('Hello')
```

>> Hello

*
```
def accel(mass, force):
    a = mass * force
    return a

newX = accel(4,5)
print(newX)
```

>> 20

WRITING TO A TEXT FILE:-

- file = open('file.txt', 'w') | w ⇒ write

⇒ file.close() ⇒ To save the change in the file

- file.write('PYTHON')
- file.close()

To write a file in new line

- file.write('python')
  file.write('\n Hello')

* READING A TEXT FILE

```
file = open('file.txt','r')        r => read
f = file.readlines()
print(f)
```

If we enter this command then, we will see some kind of "\n" in that if we have put every word in new lines.

To solve & ignore that :-

```
file = open('file.txt','r')
f = file.readlines()
newlist = []
for line in f:
    if line[-1] == '\n':
        newlist.append(line[:-1])
    else:
        newlist.append(line)
print(newlist)
```

[OR]

```
newlist = []
for line in f:
    newlist.append(line.strip())
print(newlist)
```

* Using .count() & .find()

*
```
string = 'hello'
print(string.find('l'))
```

>> 2

it will gives us 2 because the first index of l is 2.
Let's Take on more exampl with some value.

*
```
print(string.find('h'))
```

>> 0

⇒ If we put some another value for example in the above case we put '7' then the answer will be "-1" because ~~it is not~~ this value out of the string

*
```
String = 'hello a b z e d q u r s 1'
print(string.count('e'))
```

>> 2

So the .count will count how many same letters are over there.

```
string = input (" Enter Something! ")
if string.count ('_') > 0:
    Print (" Not good")

else:
    print(" hood")
```

>>

Enter Something: PYTHON

⇒ hood

Again

>> Enter Something: PY_THON

⇒ Not hood