We are starting with python today

```
# print("We are going to start with python today.")
# is used to give comment in python
# What are comments - these are non-executable codes/ text

# Input / Output operation in python

print(23)  # print is used to print the output on the screen
```

```
23
```

```
print("My name is - Manvendra")
```

```
My name is - Manvendra
```

```
print( "My name is Manvendra" , 28 )
```

```
My name is Manvendra 28
```

```
print( "My name is Manvendra" , "My age is" , 28 )
```

```
My name is Manvendra My age is 28
```

```
print()
```

```


# Adding multiple print statement will add new lines, each print
statement adds new line
print("My name is Manvendra")
print("My age is - 28")
```

```
My name is Manvendra
My age is - 28
```

```
print("My name is Manvendra")
print("My age is - 28")
print("We are learning python")
```

```
My name is Manvendra
My age is - 28
We are learning python
```

```
print("My name is Manvendra" , "My age is - 28", "We are learning
python" )
```

```
My name is Manvendra My age is - 28 We are learning python
```

```
print("My name is Manvendra" , "My age is - 28", "We are learning
python"  )
```

```
print("My name is Manvendra , My age is - 28 , We are learning python"
)
```

```
My name is Manvendra , My age is - 28 , We are learning python
```

```
print("My name is Manvendra , \nMy age is - 28 , We are learning
python"  )
```

```
My name is Manvendra ,
My age is - 28 , We are learning python
```

```
print("My name is Manvendra , \nMy age is - 28 , \nWe are learning
python"  )
```

```
My name is Manvendra ,
My age is - 28 ,
We are learning python
```

```
print("My name is Manvendra,\tMy age is - 28,\tWe are learning python"
)
```

```
My name is Manvendra, My age is - 28,  We are learning python
```

```
#\n - new line
#\t - tab space
```

```
print("Manvendra\n28")
```

```
Manvendra
28
```

```
print("Manvendra", "28", sep= "\n")
```

```
Manvendra
28
```

```
print("Manvendra", "28", sep= "-")
```

```
Manvendra-28
```

```
print("Manvendra", "28", sep= ":")
```

```
Manvendra:28
```

```
# sep - it is used to speccify the seperator in single print statement
b/w the values
```

```
print("Hello, We are staring with python")
print("We are learning about print statement")
```

```
Hello, We are staring with python
We are learning about print statement
```

```
print("Hello, We are starting with python", end=' ')
print("We are learning about print statement")
print("We learnt about sep argument, now we are learning about end
arg")
```

```
Hello, We are starting with python We are learning about print
statement
We learnt about sep argument, now we are learning about end arg
```

```
print("Hello, We are starting with python", end=',')
print("We are learning about print statement", end = ',')
print("We learnt about sep argument, now we are learning about end
arg")
```

```
Hello, We are starting with python,We are learning about print
statement,We learnt about sep argument, now we are learning about end
arg
```

```
input()  # input() function is used to take input from the user
```

```
Manvendra
```

```
{"type":"string"}
```

```
input("Enter your Name - ")
```

```
Enter your Name - Manvendra
```

```
{"type":"string"}
```

```
# Display your Name, Age and Address in same line using single print.
# Display Name, Age and Address in different line using 3 print
statement.
# Display Name, Age and Address in different line using single print.
# Display Name, Age and Address in same line using 3 print statement.
```

```
# Display your Name, Age and Address in same line using single print.
print("Manvendra", "28", "HSR Bengaluru")
```

```
Manvendra 28 HSR Bengaluru
```

```
# Display your Name, Age and Address in same line using single print.
print("Manvendra", "28", "HSR Bengaluru", sep = "-")
```

```
Manvendra-28-HSR Bengaluru
```

```
# Display Name, Age and Address in different line using 3 print
statement.
print("Manvendra")
print(28)
print("HSR, Bengaluru")
```

```
Manvendra
28
HSR, Bengaluru
```

```
# Display Name, Age and Address in different line using single print.
print("Manvendra\n28\nHsr Bengaluru")
```

```
Manvendra
28
Hsr Bengaluru
```

```
# Display Name, Age and Address in different line using single print.
print("Manvendra",28, "Hsr Bengaluru", sep = "\n")
```

```
Manvendra
28
Hsr Bengaluru
```

```
# Display Name, Age and Address in same line using 3 print statement.
print("Manvendra", end=" ")
print(28, end= " ")
print("HSR, Bengaluru")
```

```
Manvendra 28 HSR, Bengaluru
```

```
input("Enter your name - ")
```

```
Enter your name - Manvendra
```

```
{"type":"string"}
```

```
#Variables
input("Enter your marks in 10th - ")
```

```
Enter your marks in 10th - 96
```

```
{"type":"string"}
```

```
#Variables - named memory addresses
student1_marks = input("Enter your marks in 10th - ")
```

```
Enter your marks in 10th - 96
```

```
student1_marks
```

```
{"type":"string"}
```

```
print(student1_marks)
```

```
96
```

```
marks = input("Enter your marks - ")
name = input("Enter your name - ")
age = input("Enter your age - ")
```

```
Enter your marks - 96
Enter your name - Manvendra
Enter your age - 28
```

```
print("Name is",name)
print("Age is", age)
```

```
Name is Manvendra
Age is 28
```

```
print("Name is",name)
print("Age is", age)
print("Marks is", marks)
```

```
Name is Manvendra
Age is 28
Marks is 96
```

```
# Take 2 inputs
# eng_marks
# maths_marks
# total marks
```

```
eng_marks = input("Enter your english marks - ")
maths_marks = input("Enter your maths mark- ")
print("Total is", int(eng_marks) + int(maths_marks))
```

```
Enter your english marks - 80
Enter your maths mark- 86
Total is 166
```

```
80+86
```

```
166
```

```
# Whenever we use input be defualt the value is string/text
"80" + "86"  # string concatenation
```

```
{"type":"string"}
```

```
# Variables - Named memory addresses
a = 80
print(a)
```

```
80
```

```
# rules for naming variables
# Variable name cannot start with a number
A = 20

A1 = 20

1A = 20
```

```
  File "<ipython-input-68-f53e97da83b2>", line 1
    1A = 20
    ^
SyntaxError: invalid decimal literal
```

```
# rules for naming variables
# Variable name cannot start with a number
# Variable name should not contain any special character other than _

student_marks = 76

student@marks = 76
```

```
  File "<ipython-input-70-296c23e96c7a>", line 1
    student@marks = 76
    ^
SyntaxError: cannot assign to expression here. Maybe you meant '==' instead of '='?
```

```
student$marks = 76
```

```
  File "<ipython-input-71-6f725d32493c>", line 1
    student$marks = 76
            ^
SyntaxError: invalid syntax
```

```
student marks = 76
```

```
  File "<ipython-input-72-938907b40cd4>", line 1
    student marks = 76
            ^
SyntaxError: invalid syntax
```

```
# rules for naming variables
# Variable name cannot start with a number
# Variable name should not contain any special character other than _
# Variable names are case sensitive

C = 30
print(c)
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
<ipython-input-74-52a2417f51e9> in <cell line: 0>()
      1 C = 30
----> 2 print(c)

NameError: name 'c' is not defined
```

```
print(C)
```

```
30
```

```
# rules for naming variables
# Variable name cannot start with a number
# Variable name should not contain any special character other than _
# Variable names are case sensitive
# We should not use python builtin keywords and fucntions as variable
names
```

```
#print = "Manvendra"
```

```
print("Python Programming")
```

```
Python Programming
```

```
"manvendra"("Python Programming")
```

```
<>:1: SyntaxWarning: 'str' object is not callable; perhaps you missed
a comma?
<>:1: SyntaxWarning: 'str' object is not callable; perhaps you missed
a comma?
<ipython-input-81-9c2cdb2b79a4>:1: SyntaxWarning: 'str' object is not
callable; perhaps you missed a comma?
  "manvendra"("Python Programming")

---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-81-9c2cdb2b79a4> in <cell line: 0>()
----> 1 "manvendra"("Python Programming")

TypeError: 'str' object is not callable
```

```
#input = "96"
```

```
input("Enter your marks - ")
```

```
Enter your marks - 96
```

```
{"type":"string"}
```

```
# Data types -
# Int, float, string, boolean, complex

A = 10  # int
type(A)
```

```
int
```

```
# float - fractional/decimal values
X = 12.9
type(X)
```

```
float
```

```
a = 10.
print(a)
```

```
10.0
```

```
type(a)
```

```
float
```

```
s = "python"
type(s)
```

```
str
```

```
s = 'python'
print(s)
```

```
python
```

```
s = "python"
print(s)
```

```
python
```

```
s = 'python'
type(s)
```

```
str
```

```
s = "Mahamata Gandi Said - "An eye for an eye will make whole world
blind" "
```

```
  File "<ipython-input-11-ad8e6dfa4237>", line 1
    s = "Mahamata Gandi Said - "An eye for an eye will make whole
world blind" "
                               ^
SyntaxError: invalid syntax
```

```
s = 'Mahamata Gandi Said - "An eye for an eye will make whole world
blind" '

type(s)

str

print(s)

Mahamata Gandi Said - "An eye for an eye will make whole world blind"

string = 'Student's marks '

  File "<ipython-input-15-7085f3f2dab7>", line 1
    string = 'Student's marks '
                              ^
SyntaxError: unterminated string literal (detected at line 1)


string = "Student's marks "

print(string)

Student's marks

type(string)

str

s = "This is line 1
This is line 2"

  File "<ipython-input-19-ecb9ea92c22e>", line 1
    s = "This is line 1
        ^
SyntaxError: unterminated string literal (detected at line 1)


s = 'This is line 1
This is line 2'

  File "<ipython-input-20-d79d8947b2ea>", line 1
    s = 'This is line 1
        ^
SyntaxError: unterminated string literal (detected at line 1)


s = '''This is line 1
This is line 2'''

print(s)

This is line 1
This is line 2
```

```
type(s)
```

```
str
```

```
# boolean - True and False

start = True
print(start)
```

```
True
```

```
type(start)
```

```
bool
```

```
start = False
print(start)
```

```
False
```

```
type(start)
```

```
bool
```

```
# complex data type
c = 5+6j
type(c)
```

```
complex
```

```
c = 6j
print(c)
```

```
6j
```

```
type(c)
```

```
complex
```

```
c.real
```

```
0.0
```

```
c.imag
```

```
6.0
```

```
# complex data type
c = 5+6j
c.real
```

```
5.0
```

```
c.imag
```

```
6.0
```

```
#Type casting - type conversion
a = 10
type(a)
```

```
int
```

```
#convert int into float
a = float(a)

print(a)
```

```
10.0
```

```
#convert int into str
a = 10
a = str(a)

type(a)
```

```
str
```

```
a = 10  #int
type(a)
```

```
int
```

```
a = str(a)

type(a)
```

```
str
```

```
print(a)
```

```
10
```

```
# integer into boolean
a = 10
bool(a)
```

```
True
```

```
# integer into boolean
a = 0
bool(a)
```

```
False
```

```
# integer into boolean
a = -10
bool(a)
```

```
True
```

```
# integer to complex
a = 10
complex(a)
```

```
(10+0j)
```

```
# int -> float, str, bool, complex

#float -> int, str, bool, complex
A = 10.75
type(A)
```

```
float
```

```
int(A)
```

```
10
```

```
str(A)
```

```
{"type":"string"}
```

```
bool(A)
```

```
True
```

```
A = 0
bool(A)
```

```
False
```

```
A = 10.6
bool(A)
```

```
True
```

```
complex(A)
```

```
(10.6+0j)
```

```
# str - int, float, bool, complex
s = "python"
float(s)
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
```

```
<ipython-input-78-4bd19aafd689> in <cell line: 0>()
      1 # str - int, float, bool, complex
      2 s = "python"
----> 3 float(s)

ValueError: could not convert string to float: 'python'
```

```
s = "10"
int(s)
```

```
10
```

```
# We can onvert a string into int and float only if the string has
digits
```

```
s = "python 10"
int(s)
```

```
---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-80-20353fa79ef9> in <cell line: 0>()
      1 s = "python 10"
----> 2 int(s)

ValueError: invalid literal for int() with base 10: 'python 10'
```

```
s = "python"
bool(s)
```

```
True
```

```
s = ""
bool(s)
```

```
False
```

```
s = "python"
complex(s)
```

```
---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-85-1d36a802013c> in <cell line: 0>()
      1 s = "python"
----> 2 complex(s)

ValueError: complex() arg is a malformed string
```

```
s = "10"
complex(s)
```

```
(10+0j)
```

```
# Str - int/float/complex - if the str only contains numbers/digits
# only empty string will be converted to bool False, everything else
will be True

# bool - int, float, str, complex
s = True
int(s)
```

```
1
```

```
s = False
int(s)
```

```
0
```

```
float(s)
```

```
0.0
```

```
s = True
float(s)
```

```
1.0
```

```
complex(s)
```

```
(1+0j)
```

```
str(s)

# Operators - operator peforms certain actions on operands(variables)
10 + 20
```

```
30
```

```
# Arithmatic Operators -
# +, -, /, *, **

a = 10
b = 20
c = a + b
print(c)
```

```
30
```

```
a = 10
b = 20
```

```
c = a - b
print(c)
```

```
-10
```

```
a = 10
b = 20
c = a / b
print(c)
```

```
0.5
```

```
a = 10
b = 20
c = a * b
print(c)
```

```
200
```

```
10**2
```

```
100
```

```
2**2
```

```
4
```

```
2**3
```

```
8
```

```
2**0.5
```

```
1.4142135623730951
```

```
#PEMDAS
```

```
(10+20)*2/5
```

```
60
```

```
(10+20)*2/5
```

```
12.0
```

```
# The operation takes place from left to the right
```

```
# Assignment Operator
# =, += , -=, *=, /=, **=
```

```
a = 30
```

```
a += 10
a
```

```
40
```

```
a = a+10
a
```

```
50
```

```
a -= 20
a
```

```
30
```

```
a *= 2
a
```

```
60
```

```
a /= 2
a
```

```
30.0
```

```
a **= 2
a
```

```
900.0
```

```
# Comparison/Relational Operator
# == (equality), >, <, >=,<= ,!=
```

```
maths_mark = 24
eng_marks = 30
maths_mark == eng_marks
```

```
False
```

```
maths_mark = 24
eng_marks = 30
maths_mark != eng_marks
```

```
True
```

```
eng_marks > maths_mark
```

```
True
```

```
eng_marks < maths_mark
```

```
False
```

```
eng_marks >= maths_mark
```

```
True
```

```
eng_marks <= maths_mark
```

```
False
```

```
# Membership Operator - True/False
# in and not in
# "python"
```

```
s = "python"
"p" in s
```

```
True
```

```
s = "python"
"z" in s
```

```
False
```

```
s = "python"
"py" in s
```

```
True
```

```
s = "python"
"po" in s
```

```
False
```

```
s = "python"
"po" not in s
```

```
True
```

```
marks = [23,43,56,22]
23 in marks
```

```
True
```

```
# Identity Operator
# is , is not
```

```
a = 890
b = a
```

```
a
```

```
890
```

```
b
```

```
890
```

```
x = 900
y = 900
x
```

```
900
```

```
y
```

```
900
```

```
a is b
```

```
True
```

```
x is y
```

```
False
```

```
id(a)
```

```
137256552359568
```

```
id(b)
```

```
137256552359568
```

```
id(x)
```

```
137256552356784
```

```
id(y)
```

```
137256552364240
```

```
x is not y
```

```
True
```

```
# print - multiple values, sep, end, \n, \t
# input - display message
# Variables - Named memory addresses
# rules for naming variables
# data types - int, float, str, bool, complex
# type casting -
# Operators - Arithmatic, Assignment , Comparison/Relational,
Membership, Identity Operators
```