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

```
Hello
```

```
import sys
print(sys.version)
```

```
3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]
```

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

```
Hello
```

```
# simple python program
# display my product detatils

print("Product ID:101")
print("Product Name:pA")
print('Product pA Cost is:1234.55')

'''
using print function
we can display
our user defined message
'''
```

```
Product ID:101
Product Name:pA
Product pA Cost is:1234.55
'\nusing print function\nwe can display\nour user defined message\n'
```

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

```
Hello
```

```
print(Hello)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-430155136.py in <cell line: 0>()
----> 1 print(Hello)

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

```
code = 45
print(code)
```

```
45
```

```
print("code")
```

```
code
```

```
print(-13321312)
print(334423)
print(0)
print(0.0)
print(-34242.343)
print(34423.323)
```

```
-13321312
334423
0
0.0
-34242.343
34423.323
```

```
print(type(10))
```

```
<class 'int'>
```

```
print(type(10.0))
```

```
<class 'float'>
```

```
print(type(0),type(0.0),type(''))
```

```
<class 'int'> <class 'float'> <class 'str'>
```

```
print(type('45'),type('45.0'))
```

```
<class 'str'> <class 'str'>
```

```
print(type("45"),type("45.0"))
```

```
<class 'str'> <class 'str'>
```

```
print('A')
print("B")
```

```
A
B
```

```
code = 45  # initialization
print(code)
print(type(code))
```

```
45
<class 'int'>
```

```
print(code,type(code))
```

```
45 <class 'int'>
```

```
s = "Hello"
print(len(s))
```

```
5
```

```
s= ""
print(len(s))
```

```
0
```

```
pid = 101
print("product id is:pid")
```

```
product id is:pid
```

```
# product id is: 101
# Combine user defined sentence with named variable
```

```
pid = 101
print("product id is:",pid)
```

```
product id is: 101
```

```
pid = 102
pname = 'pB'
pcost = 3234.55

print("Product ID:",pid)
print("Product Name:",pname)
print('Product',pname,'Cost is:',pcost)
```

```
Product ID: 102
Product Name: pB
Product pB Cost is: 3234.55
```

```
n = 56
print("n value is:%d"%(n))
```

```
n value is:56
```

```
n="56"
print("n value is:%d"%(n))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-2449231586.py in <cell line: 0>()
      1 n="56"
----> 2 print("n value is:%d"%(n))

TypeError: %d format: a real number is required, not str
```

```
n="56"
print("n value is:{}".format(n))
```

```
n value is:56
```

```
n = 34
print("n value is:{}".format(n))
```

```
n value is:34
```

```
n = 3425.234
print("n value is:{}".format(n))
```

```
n value is:3425.234
```

```
n = 'A'
print("n value is:{}".format(n))
```

```
n value is:A
```

```
pid = 102
pname = 'pB'
pcost = 3234.55

print("Product ID:{}".format(pid))
print("Product Name:{}".format(pname))
print('Product:{} Cost is:{}'.format(pname,pcost))
```

```
Product ID:102
Product Name:pB
Product:pB Cost is:3234.55
```

```
## Multi line string
s="data1\ndata2\ndata3\ndata4"
print(type(s))
```

```
<class 'str'>
```

```
print(s)
```

```
data1
data2
data3
data4
```

```
s='''data1
data2
data3
data4'''
print(s)
```

```
data1
data2
data3
data4
```

```
s="""data1
data2
data3
data4"""
print(s)
```

```
data1
data2
data3
data4
```

```
print("""About product pA features
this product pA supports
vision and mission
```

```
for enterprise system
using this pA product
we can measure any components""")
```

```
About product pA features
this product pA supports
vision and mission
for enterprise system
using this pA product
we can measure any components
```

```
pid = 102
pname = 'pB'
pcost = 3234.55

print("""Product ID:{}
---------------------------
Product Name:{}
-------------------------
Product:{} Cost is:{}
-------------------------""".format(pid,pname,pname,pcost))
```

```
Product ID:102
---------------------------
Product Name:pB
-------------------------
Product:pB Cost is:3234.55
-------------------------
```

```
n = "45"
print(type(n))

print(int(n)) # runtime casting
print(type(n))
```

```
<class 'str'>
45
<class 'str'>
```

```
n
```

```
'45'
```

```
n = "56"
n = int(n) ### 1.type cast str ->int 2. then initialize value to variable n
print(type(n))
n
```

```
<class 'int'>
56
```

```
print(10+20)
```

```
30
```

```
10+20.0
```

```
30.0
```

```
10/5
```

```
2.0
```

```
10 // 5
```

```
2
```

```
2 ** 3
```

```
8
```

```
2 ** 5
```

```
32
```

```
3 ** 2
```

```
9
```

```
3 ** 2.5
```

```
15.588457268119896
```

```
r = 10+20.30
print(r)
```

```
30.3
```

```
int(r) # convert to int - type
```

```
30
```

```
"python"+"programming"
```

```
'pythonprogramming'
```

```
+
|--> arithmetic addition
|--> string Concatenate

*
|-->arithmetic multiplication
|--> string repeation
```

```
va = 10
vb = 20
print(va + vb)

s1 = "python"
s2 = "programming"
print(s1+s2)

print(va * 3) # multiplcation
print(s1 * 3) # string repeatation - like loop
```

```
30
pythonprogramming
30
pythonpythonpython
```

```
s = "python\n"
print(s*3)
```

```
python
python
python
```

```
s1 = "45"
s2 = "abc"
print(type(s1),type(s2))
```

```
<class 'str'> <class 'str'>
```

```
int(s1)
```

```
45
```

```
int(s2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-2598563064.py in <cell line: 0>()
----> 1 int(s2)

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

```
len(s2)
```

```
3
```

```
s1 * len(s2)
```

```
'454545'
```

```
a=10
b=20
s1="python"
s2="programming"

print(a+b)
print(s1+s2)
print(a*4)
print(s1*3)
print(s1*len(s2)) # string * int ->string
```

```
30
pythonprogramming
40
pythonpythonpython
pythonpythonpythonpythonpythonpythonpythonpythonpythonpythonpythonpython
```

```
a = 15
b = 25
c = "50"
d = "60"

print(a+b)   # 40

print(c+d)   # 5060

print(str(a)+c) # 1550

print(int(d)+a)  # 75

print(a+d) # TypeError
```

```
40
5060
1550
75
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-2717637702.py in <cell line: 0>()
     12 print(int(d)+a)  # 75
     13
---> 14 print(a+d) # TypeError

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
a = 15
b = 25
c = "50"
d = "60"
print(a+d) # TypeError -- Exit state

# we won't get any result
print(a+b)
print(c+d)

print(str(a)+c)

print(int(d)+a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-766790279.py in <cell line: 0>()
      3 c = "50"
      4 d = "60"
----> 5 print(a+d) # TypeError
      6
      7 print(a+b)   # 40

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
n = input('Enter n value:')
print(type(n))
print(n)
n
```

```
Enter n value:23
<class 'str'>
23
'23'
```

```
n = input('Enter n value:')
print(type(n))
print(n)
n
```

```
Enter n value:45.78
<class 'str'>
45.78
'45.78'
```

```
n = input('Enter n value:')
print(type(n))
print(n)
n
```

```
Enter n value:data
<class 'str'>
data
'data'
```

```
# read a product cost from <STDIN>
# calculate 18% GST
# initialize GST value to another variable
# Calculate GST + product cost

pcost = input('Enter a product Cost:')
GST = 0.18 * pcost
total = GST + pcost
print('''Product Cost is:{}
---------------------------
GST : {}
---------------------------
Total Cost including GST:{}'''.format(pcost,GST,total))
```

```
Enter a product Cost:1278.45
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-2874799472.py in <cell line: 0>()
      5
      6 pcost = input('Enter a product Cost:')
----> 7 GST = 0.18 * pcost
      8 total = GST + pcost
      9 print('''Product Cost is:{}

TypeError: can't multiply sequence by non-int of type 'float'
```

```
# read a product cost from <STDIN>
# calculate 18% GST
# initialize GST value to another variable
# Calculate GST + product cost

pcost = input('Enter a product Cost:')
GST = 0.18 * float(pcost)
total = GST + float(pcost)
print('''Product Cost is:{}
---------------------------
GST : {}
---------------------------
Total Cost including GST:{}
---------------------------'''.format(pcost,GST,total))
```

```
Enter a product Cost:1343.23
Product Cost is:1343.23
---------------------------
GST : 241.7814
---------------------------
Total Cost including GST:1585.0114
---------------------------
```

```
print(type(True),type(False))
```

```
<class 'bool'> <class 'bool'>
```

```
size = 450
size > 400
```

```
True
```

```
size < 400
```

```
False
```

```
LB = 1.34
LB >0.34
```

```
True
```

```
LB < 0.34
```

```
False
```

```
login_name = "root"
login_name == "root"
```

```
True
```

```
login_name == "ROOT"
```

```
False
```

```
file_size = 456
file_size > 400 and file_size < 500
```

```
True
```

```
file_size = 678
file_size > 400 and file_size < 500
```

```
False
```

```
login_shell = 'ksh'

login_shell == 'bash' or login_shell == 'ksh'
```

```
True
```

```
s=""
len(s) == 0
```

```
True
```

```
# operators
# arithmetic inputs (int,float) -> result: int,float
# string     inputs (int,str) --> result : str
# relational inputs (int,float,str) -->result: bool
# logical    inputs (int,float,str) -->result: bool

In Python Any Expression --> bool (or) any method/function -->bool //use conditional statement
```

```
shell_name = input('Enter a shell name:')

if(shell_name == "bash"):
  fname = "/etc/bashrc"
elif(shell_name == "ksh"):
  fname = "/etc/kshrc"
elif(shell_name == "psh"):
  fname = "C:\\window_profile"
else:
  shell_name = "/bin/nologin"
  fname = "/etc/profile"

print(f'Shell name is:{shell_name} Config file name is:{fname}')
```

```
Enter a shell name:csh
Shell name is:/bin/nologin Config file name is:/etc/profile
```

```
# Write a python program
# read a login name from <STDIN>
# test - input login name is empty --- Usage: Input is empty   -- len(login_name) == 0
# test - login name is root (or) admin <-- any one name is matched - OK
```

```
#          |
#          |-> login is not matched
#

login_name = input('Enter a login name:')

if(len(login_name) == 0):
  print("Usage: Sorry your input is empty")
elif (login_name == "root" or login_name == "admin"):
  print("Yes - Login is Matched")
else:
  print("Sorry - Login is Not matched")
```

```
Enter a login name:
Usage: Sorry your input is empty
```

```
login_name = input('Enter a login name:')

if(len(login_name) == 0):
  print("Usage: Sorry your input is empty")
else:
  if (login_name == "root" or login_name == "admin"):
    print("Yes - Login is Matched")
  else:
    print("Sorry - Login is Not matched")
```

```
Enter a login name:
Usage: Sorry your input is empty
```

```
login_name = input('Enter a login name:')

if(len(login_name) == 0):
  print("Usage: Sorry your input is empty")
else:
  if (login_name == "root" or login_name == "admin"):
    print("Yes - Login is Matched")
  else:
    print("Sorry - Login is Not matched")
```

```
Enter a login name:root
Yes - Login is Matched
```

```
login_name = input('Enter a login name:')

if(len(login_name) == 0):
  print("Usage: Sorry your input is empty")
else:
  if (login_name == "root" or login_name == "admin"):
    print("Yes - Login is Matched")
  else:
    print("Sorry - Login is Not matched")
```

```
Enter a login name:usrA
Sorry - Login is Not matched
```

```
i = 0
while(i < 5):
  print(f'i value : {i}')
  i = i + 1
```

```
i value : 0
i value : 1
i value : 2
i value : 3
i value : 4
```

```
5 < 5
```

```
False
```

```
while(False):
  print("Hello")
```

```
i = 15
while(i > 10):
  print(f'i value is:{i}')
  i = i - 1
```

```
i value is:15
i value is:14
i value is:13
i value is:12
i value is:11
```

```
10 > 10
```

```
False
```

```
i=0
while(i < 5):
  login_name = input('Enter a login name:')
  i=i+1
  if(login_name == "root" or login_name == "admin"):
    print("OK")
    break
  else:
    print("Try-Again")
```

```
Enter a login name:userA
Try-Again
Enter a login name:userB
Try-Again
Enter a login name:userC
Try-Again
Enter a login name:userD
Try-Again
Enter a login name:userE
Try-Again
```

```
i=0
while(i < 5):
  login_name = input('Enter a login name:')
  i=i+1
  if(login_name == "root" or login_name == "admin"):
    print("OK")
    break
  else:
    print("Try-Again")
```

```
Enter a login name:userA
Try-Again
Enter a login name:root
OK
```

```
pin = 1234
count = 0
max = 3
while(count < max):
  p = input('Enter a pinNumber:')
  count = count + 1
  if(int(p) == pin):
    print(f'Success pin is matched - {count}')
    break

if(int(p) != pin):
  print('Sorry your pin is blocked')
```

```
Enter a pinNumber:12321
Enter a pinNumber:123123
Enter a pinNumber:12321
Sorry your pin is blocked
```

```
for var in "python":
  print("var value is:",var)
  print("-"*15)
```

```
var value is: p
---------------
var value is: y
---------------
var value is: t
---------------
var value is: h
---------------
var value is: o
---------------
var value is: n
---------------
```

```
for var in "ab":
  print("Hello")
```

```
Hello
Hello
```

```
for var in 'abcde':
  if(var == 'c'):
    break
  else:
    print(var)
```

```
a
b
```

```
for var in 'abcde':
  if(var == 'c'):
    continue
  else:
    print(var)
```

```
a
b
d
e
```

```
for var in range(5):
  print("Hello",var)
```

```
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
```

```
range(5) ->[0,1,2,3,4]
range(15) ->[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]

range(3,15) -> [3,4,5,6,7,8,9,10,11,12,13,14]

range(3,15,2) ->[3,5,7,9,11,13]
```

```
for var in range(5):
  print(var)

print("") # empty line

var = 0
while(var < 5):
  print(var)
  var=var+1
```

```
0
1
2
3
4

0
1
2
3
4
```

```
for var in range(3,15):
  print(var)
```

```
3
4
5
6
7
8
9
10
11
12
13
14
```

```
for var in range(3,15,2):
  print(var)
```

```
3
5
7
9
11
13
```

```
# Operators + conditional + Looping
```

```
s = "python"
print(s[0])
print(s[1])
print(s[-1])
print(s[-2])
```

```
p
y
n
o
```

```
## Slicing - group of chars/range of chars
msg="python is a general purpose programming language"
print(len(msg))
```

```
48
```

```
for var in msg:
  ....//48 times

slicing
|-> string_variablename[n:m]
                        | |___ m-1 value
                        |
                        starting n value

|-> string_Variablename[n:] # from nth index to ALL

|-> string_Variablename[:m] # from 0th index to m-1
```

```
print(msg[7:21]) # 7th index to 20th index (21-1)
```

```
is a general p
```

```
print(msg[7:]) # 7th index to ALL
```

```
is a general purpose programming language
```

```
print(msg[:5]) # 0th index to 4th index (or) 1st 5 chars
```

```
pytho
```

```
print(msg[-3:])
```

```
age
```

```
print(msg[-5:])
```

```
guage
```

```
my_var = msg[-5:]
```

```
print(my_var)
```

```
guage
```

```
s="python"
s.upper()
```

```
'PYTHON'
```

```
v="data\n"
print(v)
```

```
data
```

```
v.strip()
```

```
'data'
```

```
"AB".lower()
```

```
'ab'
```

```
"abcd".title()
```

```
'Abcd'
```

```
print()
len()
type()
........//directCall

"string_Value".functionCall()
 <or>
string_Variable.functionCall()
|
object.methodCall()
```

```
#help(str)
help(str.lower)
```

```
Help on method_descriptor:

lower(self, /) unbound builtins.str method
    Return a copy of the string converted to lowercase.
```

```
s.strip()
|
|
type(s) <--- 1st - determine the type
  |
  |-> <class 'str'>
              |
            help(str.strip) // will get strip method docs

# like linux command man command
# python help(type/obj/method)
```

```
s="python"
s.upper()
```

```
'PYTHON'
```

```
print(s)
```

```
python
```

```
s = s.upper()
print(s)
```

```
PYTHON
```

```
complex(1)
```

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

```
r = complex(1)
print(type(r))
```

```
<class 'complex'>
```

```
################ End of the Day1 ####################
```