Variables

```
print("abc")
print("venky")
```

```
abc
venky
```

```
# using semicolons

print("abc");
print("venky")
```

```
abc
venky
```

```
print("abc") print("venky")
```

```
  File "<ipython-input-7-d6150b3eb194>", line 1
    print("abc") print("venky")
                 ^
SyntaxError: invalid syntax
```

Next steps: Fix error

```
print("abc"); print("venky"); a = 10; print(a)
```

```
abc
venky
10
```

```
# Indentation

print("abc")
 print("venky")
```

```
  File "<ipython-input-14-59fb797931c6>", line 4
    print("venky")
    ^
IndentationError: unexpected indent
```

Next steps: Explain error

```
welcome_message = "Welcome to Python class"
print(welcome_message)
```

```
Welcome to Python class
```

```
# space not allowed while declaring

welcome message = "Welcome to Python Class"
print(welcome message)
```

```
  File "<ipython-input-16-1f2892d9eeb1>", line 1
    welcome message = "Welcome to Python Class"
            ^
SyntaxError: invalid syntax
```

Next steps: Fix error

```
_abc = 9
print(_abc)
9abc = 10
print(9abc)
```

```
  File "<ipython-input-18-4c18c323b647>", line 3
    9abc = 10
    ^
SyntaxError: invalid decimal literal
```

Next steps: Fix error

```
_ = 100
print(_)
```

```
100
```

```
import keyword
keyword.kwlist
```

```
['False',
 'None',
 'True',
 'and',
 'as',
 'assert',
 'async',
 'await',
 'break',
 'class',
 'continue',
 'def',
 'del',
 'elif',
 'else',
 'except',
 'finally',
 'for',
 'from',
 'global',
 'if',
 'import',
 'in',
 'is',
 'lambda',
 'nonlocal',
 'not',
 'or',
 'pass',
 'raise',
 'return',
 'try',
 'while',
 'with',
 'yield']
```

```
len(keyword.kwlist) # 35 keywords in python
```

```
35
```

```
False = 10
print(False)
```

```
  File "<ipython-input-22-55e34d1ef718>", line 1
    False = 10
    ^
SyntaxError: cannot assign to False
```

Next steps: Fix error

```
print("ABC")
```

```
ABC
```

```
print = 10
```

```
# assign a name of the person

n = "Venky"

print(n)
```

```
Venky
```

```
name = "Venky"
s_n = "abi"
student_name = "abi"
```

```
length_of_person_name = len(name)
name_length = len(name)
```

```
5
```

```
# try to avoid using Capital letters while assigning variables

NUM = 10
num = 10
print(NUM)
print(num)
```

```
10
10
```

Comments

```
# Single Line Comment

"""
I'm explaining multiple line comments here
in Python class
"""
'''
explain
concepts
of
comments
'''

# explain
# concepts
# of
# comments
print("comments")
```

```
comments
```

Datatypes

Strings

```
# A string is a series of characters

new_string = "This is a string"
new_string_2 = 'This is a string'
new_string_3 = 'This is a string'

print(new_string)
print(new_string_2)
print(type(new_string))
print(type(new_string_2))
```

```
This is a string
This is a string
<class 'str'>
<class 'str'>
```

```
print(id(new_string))
print(id(new_string_2))
print(id(new_string_3))
```

```
132141852564160
132141852563120
132141852570000
```

```
name = "Venky"
name_2 = "Venky"

print(id(name))
print(id(name_2))
```

```
132141854175664
132141854175664
```

```
a = 250
b = 250

print(id(a))
print(id(b))

c = 300
d = 300
print(id(c))
print(id(d))
```

```
132143135612944
132143135612944
132141854396720
132141854401296
```

```
message = 'I have watched "Indian-2" Audio Launch Yesterday.'
print(message)
```

```
I have watched "Indian-2" Audio Launch Yesterday.
```

```
string_1 = "This is a string"
string_2 = "This is a string "
string_1 == string_2
```

```
False
```

```
name = "Venky"
name_2 = "Venky  "
name_3 = "  Venky"
name_4 = " Venky "
```

```
name_2
```

```
'Venky  '
```

```
name_2.rstrip()
```

```
'Venky'
```

```
name_3
```

```
'  Venky'
```

```
name_3.lstrip()
```

```
'Venky'
```

```
name_4
```

```
' Venky '
```

```
name_4.strip()
```

```
'Venky'
```

```
first_name = "Venky"
last_name = "Viky"

first_name+" "+last_name
```

```
'Venky Viky'
```

Numeric

```
value = 10
print(type(value))
```

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

```
f_value = 10.2
print(type(f_value))
```

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

```
a = 10
b = 12
s = a+b
sub = a-b
mul = a*b
div = a/b
print(s)
print(type(s))
print(sub)
print(type(sub))
print(mul)
print(type(mul))
print(div)
print(type(div))
```

```
22
<class 'int'>
-2
<class 'int'>
120
<class 'int'>
0.8333333333333334
<class 'float'>
```

```
a = 10
b = 5
c = a/b

print(c)
print(type(c))

# floor division

d = a//b
print(d)
print(type(d))
```

```
2.0
<class 'float'>
2
<class 'int'>
```

```
a = 12
b = 5

c = a/b
print(c)
print(type(c))
d = a//b
print(d)
print(type(d))

a = 14
b = 5

c = a/b
print(c)
print(type(c))
d = a//b
print(d)
print(type(d))
```

```
2.4
<class 'float'>
2
<class 'int'>
2.8
<class 'float'>
2
```

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

```
a = 10
b = 0
print(a/b)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-65-7d1d879c47c9> in <cell line: 3>()
      1 a = 10
      2 b = 0
----> 3 print(a/b)

ZeroDivisionError: division by zero
```

Next steps: Explain error

```
f_value = 12.2
print(f_value)
print(type(f_value))
a = 12.2
b = 12.0
c = a+b
d = a-b
e = a*b
f = a/b
print(d)
```

```
12.2
<class 'float'>
0.1999999999999993
```

```
from decimal import Decimal
Decimal(0.3)
```

```
Decimal('0.299999999999999988897769753748434595763683319091796875')
```

```
a = 0.1
b = 0.2
c = a+b
print(c)
print(type(c))
```

```
0.30000000000000004
<class 'float'>
```

```
from decimal import Decimal

a = Decimal('10.4')
b = Decimal('12.5')
c = a+b
print(c)
print(type(c))
```

```
22.9
<class 'decimal.Decimal'>
```

```
10.4+12.5
```

```
22.9
```

```
f = 12.0
i = int(f)
print(i)
print(type(i))
```

```
12
<class 'int'>
```

```
a = 12.56442845
round(a,2)
```

```
12.56
```

```
Decimal(0.1)
```

```
Decimal('0.1000000000000000055511151231257827021181583404541015625')
```

```
Decimal(0.1)
```

```
Decimal('0.1000000000000000055511151231257827021181583404541015625')
```

```
a = 10
b = 12.3
c = a+b
print(c)
print(type(c))
```

```
22.3
<class 'float'>
```

```
a = complex(12)
print(a)
print(type(a))
b = complex(12, 2)
print(b)
print(type(b))
a+b
```

```
(12+0j)
<class 'complex'>
(12+2j)
<class 'complex'>
(24+2j)
```

```
a = 0.2
b = 0.1
c = a+b
print(c)
```

```
0.30000000000000004
```

```
a = 4
b = 2

c = a/b
print(c)
print(type(c))
```

```
2.0
<class 'float'>
```

Lists

```
new_list = [2,4,6,10,12.4, 12.8, "Venky", "Gate", "Python", 128, "IPL", "VIRAT"]
```

```
type(new_list)
```

```
list
```

```
new_list[-3]
```

```
128
```

```
len(new_list)
```

```
12
```

```
new_list[0]
```

```
2
```

```
new_list[-12]
```

```
2
```

```
print(type(new_list))
print(id(new_list))

new_list[-1] = "Rohit"
print(new_list)
print(id(new_list))
```

```
<class 'list'>
132141853140608
[2, 4, 6, 10, 12.4, 12.8, 'Venky', 'Gate', 'Python', 128, 'IPL', 'Rohit']
132141853140608
```

```
new_list = [1,2,3]
print(new_list)
print(id(new_list))
print(id(new_list[0]))
new_list[0] = 10
print("Updated list: ", new_list)
print(id(new_list))
print(id(new_list[0]))
```

```
[1, 2, 3]
132141853365952
132143135604976
Updated list:  [10, 2, 3]
132141853365952
132143135605264
```

```
new_list = [2,2,3]
new_list2 = [270, 270, 300, 10000, 10000]
print("First list")
print(id(new_list[0]))
print(id(new_list[1]))
print("Second List")
print(id(new_list2[0]))
print(id(new_list2[1]))
print(id(new_list2[-1]))
print(id(new_list2[-2]))
```

```
First list
132143135605008
132143135605008
Second List
132141853217520
132141853217520
```

```
132141853216976
132141853216976
```

```
a = 10000
b = 10000

print(id(a))
print(id(b))
```

```
132141853219408
132141853219280
```

```
name = "Venky"
print(name)
print(id(name))

name = "Kenky"
print(name)
print(id(name))
```

```
Venky
132141854175664
Kenky
132141853385136
```

```
name[0] = "K"
print(name)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-116-9fcaac50d270> in <cell line: 1>()
----> 1 name[0] = "K"
      2 print(name)

TypeError: 'str' object does not support item assignment
```

Next steps: Explain error

```
print(name[0])
print(name[1])
print(name[2])
print(name[3])
print(name[4])
print(name[-1])
print(name[-2])
print(name[-3])
print(name[-4])
print(name[-5])
```

```
V
e
n
k
y
y
k
n
e
V
```

Tuple

```
new_tuple = (2,5,10,"Venky",12.5,2,5,10,"Viky")
print(new_tuple)
print(type(new_tuple))
print(id(new_tuple))
```

```
new_tuple[-1]
```

```
'Viky'
```

```
new_tuple = (2,4,6,10,12.4, 12.8, "Venky", "Gate", "Python", 128, "IPL", "VIRAT")

print(new_tuple[-1])
```

```
VIRAT
```

```
new_tuple_2 = (2,4,6,(21,4),[24,2],12.5,10,22,22,257,1000,10000,10000,257)
```

Start coding or generate with AI.

```
new_tuple[-1] = "Rohit"
print(new_tuple)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-128-2746a8e48515> in <cell line: 1>()
----> 1 new_tuple[-1] = "Rohit"
      2 print(new_tuple)

TypeError: 'tuple' object does not support item assignment
```

Next steps: Explain error

```
new_list1 = [1,2]
new_tuple1 = tuple(new_list1)
print(id(new_list1))
print(new_tuple1)
print(type(new_tuple1))
print(id(new_tuple1))
```

```
132141853451456
(1, 2)
<class 'tuple'>
132141854145856
```