Codes Snippet

March 4, 2025

[23]:
```
print("Hello, World")
```

```
Hello, World
```

[5]:
```
a=10
b=20
print("The Sum:",(a+b))
```

```
The Sum: 30
```

[7]:
```
# bool,int, float, str
a = 'Data Science'
print(a)
type(a)
```

```
Data Science
```

[7]:
```
str
```

[9]:
```
type(a)
```

[9]:
```
str
```

[13]:
```
z = True
type(z)
```

[13]:
```
bool
```

[16]:
```
# bool -> int -> float -> str
True + 6 + 7.5
# 1+6+7.5
```

[16]:
```
14.5
```

[18]:
```
int(7.5) + 3
```

[18]:
```
10
```

[20]:
```
bool(0)
```

[20]:
```
False
```

[29]:
```
#Auto typecasting
True + 3 + int(4.5)
```

[29]:
```
8
```

[31]:
```
a=3
b=4.5
print(type(a))
print(type(b))
```

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

[33]:
```
a + int(b)
```

[33]:
```
7
```

[35]:
```
3 + int('4')
```

[35]:
```
7
```

[37]:
```
#True -> 1, False -> 0
False + 4
```

[37]:
```
4
```

[48]:
```
int(123.987)
```

[48]:
```
123
```

[57]:
```
int(10+5j)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[57], line 1
----> 1 int(10+5j)

TypeError: int() argument must be a string, a bytes-like object or a real␣
 ↪number, not 'complex'
```

[59]:
```
int(True)
```

[59]:
```
1
```

[61]:
```
int(False)
```

[61]:
```
0
```

[63]:
```
int("10")
```

[63]:
```
10
```

[65]:
```
int("10.5")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[65], line 1
----> 1 int("10.5")

ValueError: invalid literal for int() with base 10: '10.5'
```

[67]:
```
int("ten")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[67], line 1
----> 1 int("ten")

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

[71]:
```
int("0B1111")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[71], line 1
----> 1 int("0B1111")

ValueError: invalid literal for int() with base 10: '0B1111'
```

[81]:
```
bool(0)
```

[81]:
```
False
```

[83]:
```
bool(1)
```

[83]:
```
True
```

[85]:
```
bool(10)
```

[85]:
```
True
```

[87]:
```
bool(10.5)
```

[87]:
```
True
```

[89]:
```
bool(0.178)
```

[89]:
```
True
```

[91]:
```
bool(0.0)
```

[91]:
```
False
```

[93]:
```
bool(10-2j)
```

[93]:
```
True
```

[95]:
```
bool(0+1.5j)
```

[95]:
```
True
```

[97]:
```
bool(0+0j)
```

[97]:
```
False
```

[99]:
```
bool("True")
```

[99]:
```
True
```

[101]:
```
bool("False")
```

[101]:
```
True
```

[103]:
```
bool("")
```

[103]:
```
False
```

[111]:
```
a = "I am a Data Scientist" #indexing starts from 0 from 1 & we count space as␣
 ↪well so, an index of a is 2. # "1=0", "space="1", "a=2", "m=3 & so on.␣
 ↪#"t=-1", "s=-2" & so on in case of reverse indexing
a
```

[111]:
```
'I am a Data Scientist'
```

[113]:
```
a[2:4] #it will print a letter which is on index 2 & 3 excluding index 4
```

[113]:
```
'am'
```

[115]:
```
a[-9:] #if we see the index in the reverse direction, it will start from -1␣
↪#so, [-9:] from index -9 it will print all letters including a word at index␣
↪9
```

[115]:
```
'Scientist'
```

[117]:
```
# : -> slicing operator
a[7:] #starting from index 7 (including) it will print till end
```

[117]:
```
'Data Scientist'
```

[119]:
```
a[:7] # print everything excluding the letters starting from index 7
```

[119]:
```
'I am a '
```

[2]:
```
#Slicing a list
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

#Get a slice of the list from index 2 to index 6 (exclusive)
slice_1 = my_list[2:6]
print(slice_1)
```

```
[3, 4, 5, 6]
```

[4]:
```
# Get a slice of the list from index 1 to the end
slice_2 = my_list[1:]
print(slice_2)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10]
```

[6]:
```
# Get a slice of the list from the beginning to index 5 (exclusive)
slice_3 = my_list[:5]
print(slice_3)
```

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

[12]:
```
# Get a slice of the list with a step size of 2
slice_4 = my_list[::2]
print(slice_4)
```

```
[1, 3, 5, 7, 9]
```

[14]:
```
# Comparison  operators
x = 5
y = 10
print(x == y)
```

```
False
```

[16]:
```
print(x != y)
```

```
True
```

[18]:
```
print (x > y)
```

```
False
```

[20]:
```
print(x < y)
```

```
True
```

[22]:
```
print(x >= y)
```

```
False
```

[24]:
```
print(x <= y)
```

```
True
```

[ ]:
```

```