Type Casting

In [7]:

```
# type casting
x=6.9
print(x)
print(type(x))
print(int(x))  # it truncate(cut) the float value
print(type(int(x)))
```

```
6.9
<class 'float'>
6
<class 'int'>
```

In [4]:

```
y = int(x) # storing value as integer
print(y)
print(type(y))
```

```
6
<class 'int'>
```

In [8]:

```
import math
print(math.floor(x)) # it rounds down the float value
print(type(math.floor(x)))
print(int(x))
print(type(int(x)))
```

```
6
<class 'int'>
6
<class 'int'>
```

In [12]:

```
s= "11"
# print(math.floor(s))  # throws error
# print(type(math.floor(s)))
print(int(s))
print(type(int(s)))
```

```
11
<class 'int'>
```

In [15]:

```
x = -3.4
print(math.floor(x)) # it gives smaller value
print(type(math.floor(x)))
print(int(x))  # it discard float value
print(type(int(x)))
```

```
-4
<class 'int'>
-3
<class 'int'>
```

In [14]:

```
x = 6
print(math.floor(x))
print(type(math.floor(x)))
print(int(x))
print(type(int(x)))
```

```
6
<class 'int'>
6
<class 'int'>
```

In [16]:

```
x="15"  # covertable string to int
print(x)
print(type(x))
print(int(x))
print(type(int(x)))
```

```
15
<class 'str'>
15
<class 'int'>
```

In [18]:

```
x="nikhil"  # non-covertable string to int
print(x)
print(type(x))
# print(int(x))  # throws error
# print(type(int(x)))
```

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

In [19]:

```
x="11.95"  # non-covertable string to int
print(x)
print(type(x))
# print(int(x))  # throws error
# print(type(int(x)))
```

```
11.95
<class 'str'>

---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
Cell In[19], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(int(x))  # throws error
      5 print(type(int(x)))

ValueError: invalid literal for int() with base 10: '11.95'
```

In [20]:

```
x=True  # bool to int
print(x)
print(type(x))
print(int(x))  # convert True -> 1
print(type(int(x)))
```

```
True
<class 'bool'>
1
<class 'int'>
```

In [21]:

```
x=False  # bool to int
print(x)
print(type(x))
print(int(x))  # throws error
print(type(int(x)))
```

```
False
<class 'bool'>
0
<class 'int'>
```

In [23]:

```
x=5+6j  # complex to int
print(x)
print(type(x))
print(int(x))  # throws error
print(type(int(x)))
```

```
(5+6j)
<class 'complex'>

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[23], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(int(x))  # throws error
      5 print(type(int(x)))

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

In [25]:

```
# type casting in float
x=56  # int to float
print(x)
print(type(x))
print(float(x))
print(type(float(x)))
```

```
56
<class 'int'>
56.0
<class 'float'>
```

In [27]:

```
x=True  # bool to float
print(x)
print(type(x))
print(float(x))  # it converts True -> 1.0
print(type(float(x)))
```

```
True
<class 'bool'>
1.0
<class 'float'>
```

In [28]:

```
x=False  # bool to float
print(x)
print(type(x))
print(float(x))  # it converts False -> 0.0
print(type(float(x)))
```

```
False
<class 'bool'>
0.0
<class 'float'>
```

In [29]:

```
x="11.8"  # appropriate string to float
print(x)
print(type(x))
print(float(x))
print(type(float(x)))
```

```
11.8
<class 'str'>
11.8
<class 'float'>
```

In [31]:

```
x="nikhil"  # inappropriate string to float
print(x)
print(type(x))
print(float(x))  # throws error
print(type(float(x)))
```

```
nikhil
<class 'str'>

--------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
Cell In[31], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(float(x))  # throws error
      5 print(type(float(x)))

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

In [32]:

```
x="11"  # appropriate string to float
print(x)
print(type(x))
print(float(x))
print(type(float(x)))
```

```
11
<class 'str'>
11.0
<class 'float'>
```

In [35]:

```
x=11.6+2.9j  # complex to float
print(x)
print(type(x))
print(float(x))  # throws error
print(type(float(x)))
```

```
(11.6+2.9j)
<class 'complex'>

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[35], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(float(x))  # throws error
      5 print(type(float(x)))

TypeError: float() argument must be a string or a real number, not 'comple
x'
```

In [36]:

```
# type casting in complex

x=7  # int to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
7
<class 'int'>
(7+0j)
<class 'complex'>
```

In [37]:

```
x=74.5  # float to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
74.5
<class 'float'>
(74.5+0j)
<class 'complex'>
```

In [38]:

```
x=True  # bool to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
True
<class 'bool'>
(1+0j)
<class 'complex'>
```

In [39]:

```
x=False  # bool to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
False
<class 'bool'>
0j
<class 'complex'>
```

In [40]:

```
x="54"  # approprite string to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
54
<class 'str'>
(54+0j)
<class 'complex'>
```

In [43]:

```
x="54"  # approprite string to complex
print(x)
print(type(x))
y=complex(x)
print(y)
print(type(y))
print(y.real,y.imag)
print(type(y.real),type(y.imag))
```

```
54
<class 'str'>
(54+0j)
<class 'complex'>
54.0 0.0
<class 'float'> <class 'float'>
```

In [41]:

```
x="54.7"  # approprite string to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
54.7
<class 'str'>
(54.7+0j)
<class 'complex'>
```

In [44]:

```
x="vasima"  # inapproprite string to complex
print(x)
print(type(x))
print(complex(x))
print(type(complex(x)))
```

```
vasima
<class 'str'>

--------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
Cell In[44], line 4
      2 print(x)
      3 print(type(x))
----> 4 print(complex(x))
      5 print(type(complex(x)))

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

In [45]:

```
# typr casting in bool
x=1  # int to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
1
<class 'int'>
True
<class 'bool'>
```

In [47]:

```
x=0  # int to bool
print(x)
print(type(x))
print(bool(x))  # only for zero gives false in bool f()
print(type(bool(x)))
```

```
0
<class 'int'>
False
<class 'bool'>
```

In [48]:

```
x=562  # int to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
562
<class 'int'>
True
<class 'bool'>
```

In [49]:

```
x=-213  # int to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
-213
<class 'int'>
True
<class 'bool'>
```

In [50]:

```
x=25.36  # float to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
25.36
<class 'float'>
True
<class 'bool'>
```

In [51]:

```
x=25+56j  # complex to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
(25+56j)
<class 'complex'>
True
<class 'bool'>
```

In [52]:

```
x=25+0j  # complex to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
(25+0j)
<class 'complex'>
True
<class 'bool'>
```

In [53]:

```
x=0+56j  # complex to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
56j
<class 'complex'>
True
<class 'bool'>
```

In [54]:

```
x=0+0j  # complex to bool
print(x)
print(type(x))
print(bool(x))  # only 0+0j gives False
print(type(bool(x)))
```

```
0j
<class 'complex'>
False
<class 'bool'>
```

In [55]:

```
x="oi"  # string to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
oi
<class 'str'>
True
<class 'bool'>
```

In [56]:

```
x="256"  # string to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
256
<class 'str'>
True
<class 'bool'>
```

In [57]:

```
x="0"  # string to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
0
<class 'str'>
True
<class 'bool'>
```

In [58]:

```
x="1"  # string to bool
print(x)
print(type(x))
print(bool(x))
print(type(bool(x)))
```

```
1
<class 'str'>
True
<class 'bool'>
```

In [66]:

```
x=""  # string to bool
print(x)
print(type(x))
print(bool(x))  # it returns False
print(type(bool(x)))
```

```

<class 'str'>
False
<class 'bool'>
```

In [60]:

```
# string type casting string

x = 88  # int to string
print(x)
print(type(x))
print(str(x))
print(type(str(x)))
```

```
88
<class 'int'>
88
<class 'str'>
```

In [61]:

```
x = 88.26  # float to string
print(x)
print(type(x))
print(str(x))
print(type(str(x)))
```

```
88.26
<class 'float'>
88.26
<class 'str'>
```

In [62]:

```
x = 88+6j  # complex to string
print(x)
print(type(x))
print(str(x))
print(type(str(x)))
```

```
(88+6j)
<class 'complex'>
(88+6j)
<class 'str'>
```

In [63]:

```
x = True  # bool to string
print(x)
print(type(x))
print(str(x))
print(type(str(x)))
```

```
True
<class 'bool'>
True
<class 'str'>
```

In [64]:

```
x = False  # bool to string
print(x)
print(type(x))
print(str(x))
print(type(str(x)))
```

```
False
<class 'bool'>
False
<class 'str'>
```

Programs on Type Casting

In [70]:

```
# write a program to calculate Area of circle. Data should be taken from user but the va

import math
# print(math.pi)
r=float(input("enter radius of a circle:"))
area = math.pi*r**2
print("area of circle is:",area)
```

```
3.141592653589793
enter radius of a circle:5
area of circle is: 78.53981633974483
```

In [3]:

```python
# write a program to calculate the area of triangle, length of side should be given by u
# python itself

import math
print(math.sqrt(25))

a=float(input("enter length of side one:"))
b=float(input("enter length of side two:"))
c=float(input("enter length of side three:"))

s = (a+b+c)/2
area =(s*(s-a)*(s-b)*(s-c))
print("area of triangle is:",math.sqrt(area))
```

```
5.0
enter length of side one:10
enter length of side two:12
enter length of side three:9
area of triangle is: 44.039045175843675
```

In [2]:

```python
# write a program to calculate SI, data should be taken from the user

p=float(input("Enter principle amount:"))
r=float(input("Enter rate of interest:"))
t=int(input("enter time:"))
si=(p*r*t)/100
print("your SI is:",si)
```

```
Enter principle amount:10000
Enter rate of interest:15
enter time:2
your SI is: 3000.0
```

In [ ]:

```

```