text type : str

In [8]:
```
s1="arman"
s2="arman's home" #if there is use of aphosthophe, then use double qoutes
s3='''ugignvtiughekighytgkfgt
bfrbhfkrg'''
'''hnth
nghgdghd'''#without variable assigment a triple qoute represents a multilie comment
s3
```

Out[8]:
```
'ugignvtiughekighytgkfgt\nbfrbhfkrg'
```

In [6]:
```
print(s3)
```

```
ugignvtiughekighytgkfgt
bfrbhfkrg
```

numeric type: (int,float)

In [10]:
```
x=1
y=3635465
z=-9743874
print(type(x))
print(type(y))
print(type(z))
```

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

In [11]:
```
x1=1.98
y1=3635465.774
z1=-9743874.5433
print(type(x1))
print(type(y1))
print(type(z1))
```

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

In [13]:
```
a=36e3
b=456E4
print(type(a))
print(type(b))
```

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

In [16]:
```
#decimal form
a1=1111
print(a1)

#binary form
a=0b111
b=0B111
print(a)
print(b)


#octal form
c=0o1111
```

```
d=0O1111
print(c)
print(d)


#hexa decimal
e=0x1111
f=0X1111
print(e)
print(f)

#the final answers will be given in the decimal form only, when we covert them then
```

```
1111
7
7
585
585
4369
4369
```

base conversion

binary conversion

In [19]:
```
#1. bin()

bin(15)
```

Out[19]:
```
'0b1111'
```

In [20]:
```
bin(0o11)
```

Out[20]:
```
'0b1001'
```

In [21]:
```
bin(0x111)
```

Out[21]:
```
'0b100010001'
```

octal conversion

In [22]:
```
#oct()
oct(11010)
```

Out[22]:
```
'0o25402'
```

In [23]:
```
oct(0x112A4)
```

Out[23]:
```
'0o211244'
```

In [25]:
```
oct(0b10111)
```

Out[25]:
```
'0o27'
```

hexadecimal conversion

In [26]:
```
hex(0b10111)
```

```
Out[26]: '0x17'

In [27]: hex(12)

Out[27]: '0xc'

In [29]: hex(0o1101)

Out[29]: '0x241'

In [30]: hex(0x000101)

Out[30]: '0x101'

In [31]: hex(11.32)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-31-17d3b93e9156> in <module>
----> 1 hex(11.32)

TypeError: 'float' object cannot be interpreted as an integer
```

note to self:only integers are allowed in all of the above conversions

sequence type:list,tuple,range

```
In [32]: list1=["apple","banana","cherry"]
         print(list1)

['apple', 'banana', 'cherry']

In [34]: list2=["apples",1,2,False]
         print(list2)

['apples', 1, 2, False]

In [35]: print(type(list1))

<class 'list'>

In [36]: print(type(list2))

<class 'list'>
```

note to self:differnt datatypes are allowed in a list

```
In [47]: list1[0]="strawberry"
         print(list1)

         #ordered, changable
         #allow duplicates indexed

['strawberry', 'banana', 'cherry']

In [39]: #tuple

         my_tuple=("apple","banana","cherry")
         print(my_tuple)

('apple', 'banana', 'cherry')

In [40]: tuple2=("abcd",3,True,False)
         print(tuple2)
```

```
('abcd', 3, True, False)
```

In [41]:
```
print(type(my_tuple))
print(type(tuple2))
```

```
<class 'tuple'>
<class 'tuple'>
```

In [43]:
```
my_tuple[0]="Strawberry"
print(my_tuple)
#ordered,unchangable
#allow duplicates indexed
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-43-b9a2ccbf31fb> in <module>
----> 1 my_tuple[0]="Strawberry"
      2 print(my_tuple)
      3 #ordered unchangable allow duplicates indexed

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

Mapping type: dict

In [44]:
```
d={10:'lucky',20:'arman',30:'dhairya'}
print(d[10])
```

```
lucky
```

In [45]:
```
print(d[20])
```

```
arman
```

In [46]:
```
print(d[30])
```

```
dhairya
```

In [48]:
```
#ordered,changable
#allow duplicates indexed
```

In [51]:
```
d={10:'lucky',10:'arman',30:'dhairya'}
print(d[10])

#there will be no error but the duplicte will get over written
```

```
arman
```

In [52]:
```
d1={10:'lucky',20:'lucky',30:'dhairya'}
print(d1[10])
```

```
lucky
```

Set type

In [8]:
```
x={1,2,3,4,56,7,8} #curly brackets but no key value pairs
print(x)
#kaya order ma answer aave e fix nathi.
#every time run krya pachi ek different output aavse
#unordered
```

```
{1, 2, 3, 4, 7, 8, 56}
```

In [7]:
```
y={'apple','banana',"apple","banana"}
print(y)
#duplicates are not allowed, i.e. they get removed
```

```
{'apple', 'banana'}
```

```
In [10]: list=["apple","banana","cherry","apple"]
         y=frozenset(list)
         print(y)
```

```
frozenset({'apple', 'banana', 'cherry'})
```

Boolean Type: bool

```
In [11]: print(bool(0))
```

```
False
```

```
In [12]: print(bool(1))
```

```
True
```

```
In [13]: print(bool("apple"))
```

```
True
```

```
In [14]: print(bool(""))
```

```
False
```

```
In [15]: print(bool(20>6))
```

```
True
```

```
In [16]: print(bool(20<5))
```

```
False
```

```
In [17]: print(bool(20==8))
```

```
False
```

variables(typecast)

```
In [18]: x=7
         y=str(7)
         z=float(7)
         print(x)
         print(y)
         print(z)
```

```
7
7
7.0
```

```
In [21]: a=1,2,3,4,5
         print(type(a))
         print(a)
```

```
<class 'tuple'>
(1, 2, 3, 4, 5)
```

```
In [24]: a,b,c,d=1,2,3,4
         print(type(a))
         print(a)
         print(type(b))
         print(b)
```

```
<class 'int'>
1
```

```
<class 'int'>
2
```

In [26]:
```
a=b=c=d=2
print(type(a))
print(a)
print(b)
print(c)
```

```
<class 'int'>
2
2
2
```

In [27]:
```
a="10"
b=20
print(a+b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-7d74b4776bd7> in <module>
      1 a="10"
      2 b=20
----> 3 print(a+b)

TypeError: can only concatenate str (not "int") to str
```

In [28]:
```
a=10
b=20
print(a+b)
```

```
30
```

In [29]:
```
a="10"
b="20"
print(a+b)
```

```
1020
```

In [30]:
```
a=1234
print("num=",a)
```

```
num= 1234
```

In [32]:
```
a="1234"
print("num="+a)
```

```
num=1234
```

In [33]:
```
#multiplication repetition
```

In [34]:
```
a="10"
b=20.5
a*b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-34-be2199b35809> in <module>
      1 a="10"
      2 b=20.5
----> 3 a*b

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

In [35]:
```
a="10"
b=7
a*b
```

```
Out[35]: '10101010101010'
```

```
In [36]: #while using * there should be one variable as integer for it to do multiple times
         # * means repetition
```

Global variables vs Local variables

```
In [39]: a="python" #global var
         def test():
             print(a)
             a="java"  #local val
             print(a)
         test()
         print(a)
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-39-570f9c8f9108> in <module>
      4     a="java"  #local val
      5     print(a)
----> 6 test()
      7 print(a)

<ipython-input-39-570f9c8f9108> in test()
      1 a="python" #global var
      2 def test():
----> 3     print(a)
      4     a="java"  #local val
      5     print(a)

UnboundLocalError: local variable 'a' referenced before assignment
```

```
In [41]: a="python" #global var
         def test():
             global a
             a="java"  #local val
             print(a)
         test()
         print(a)
```

```
java
java
```

```
In [42]: a="python" #global var
         def test():
             a="java"  #local val
             print(a)
         test()
         print(a)
```

```
java
python
```

```
In [43]: a="python" #global var
         def test():
             global a
             print(a)
             a="java"  #local val
             print(a)
         test()
         print(a)

         #here the global value of a is changed
```

```
python
java
java
```

Reading user input

In [47]:
```
x=(input("Enter data:"))
#by default it takes in string
```

```
Enter data:fgdgt13.87
```

In [45]:
```
x=int(input("Enter data:"))
```

```
Enter data:trt
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-45-81a5892e4dec> in <module>
----> 1 x=int(input("Enter data:"))

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

In [46]:
```
x=int(input("Enter data:"))
```

```
Enter data:45767
```

In [ ]:
```
#
```