Simple Numbers

4

In [3]:
```
4+4 #addition
```

Out[3]: 8

In [4]:
```
4-3 #substraction
```

Out[4]: 1

In [5]:
```
4*3 #multiplication
```

Out[5]: 12

In [6]:
```
4/3 #division
```

Out[6]: 1.3333333333333333

In [7]:
```
4//3 #floor division
```

Out[7]: 1

In [8]:
```
4%5 #modulous
```

Out[8]: 4

Number are operatnds and + _ * / are operators

what are the different operators

- arthmetic operator
- assignment operator
- rational operator
- logical operator
- logical operator
- unary operator

In [9]:
```
 10+5
```

Out[9]: 15

In [10]:
```
10-5
```

Out[10]: 5

In [11]:
```
10*5
```

Out[11]: 50

In [12]:
```
10/5
```

Out[12]: 2.0

In [15]:
```
10**5
```

Out[15]: 100000

Assignment operator

In [36]:
```
x=10
x
```

Out[36]: 10

In [37]:
```
x+2
```

Out[37]: 12

In [39]:
```
x+=2
x
```

Out[39]: 14

In [40]:
```
x-=2
x
```

Out[40]: 12

In [41]:
```
x*=2
x
```

Out[41]: 24

In [42]:
```
x/=2
x
```

Out[42]: 12.0

unary operator

In [43]:
```
a=5
a
```

Out[43]: 5

In [44]:
```
b=-a
b
```

Out[44]: -5

In [45]:
```
a=123
a
```

Out[45]:
```
123
```

In [46]:
```
type(a)
```

Out[46]:
```
int
```

In [47]:
```
import keyword
keyword.kwlist
```

Out[47]:
```
['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']
```

In [49]:
```
len(keyword.kwlist)
```

Out[49]:
```
35
```

In [50]:
```
import sys
sys.version
```

Out[50]:
```
'3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.192
9 64 bit (AMD64)]'
```

In [51]:
```
a=10
print(a)
```

```
10
```

In [52]:
```
a='work'
print(a)
```

```
work
```

Python Data Types

- integer
- float
- complex
- string
- boolean

In [64]:
```
a=25
print(a)
type(a)
```

```
25
```

Out[64]:
```
int
```

In [56]:
```
type(a)
```

Out[56]:
```
float
```

In [59]:
```
a=1+2j
a
```

Out[59]:
```
(1+2j)
```

In [60]:
```
type(a)
```

Out[60]:
```
complex
```

In [65]:
```
a='nit'
print(a)
```

```
nit
```

In [62]:
```
type(a)
```

Out[62]:
```
str
```

float

In [2]:
```
temp=98.6
temp
```

Out[2]:
```
98.6
```

String

In [3]:
```
s=hyderabad
s
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 s=hyderabad
      2 s

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

In [4]:
```
s='hyderabad' #single quote declaration
s
```

Out[4]: 'hyderabad'

In [7]:
```
type(s)
```

Out[7]: str

In [8]:
```
s1="nit"
s1
```

Out[8]: 'nit'

In [9]:
```
s2='''the cat eats a fish''' # multiline string
s2
```

Out[9]: 'the cat eats a fish'

In [10]:
```
type(s2)
```

Out[10]: str

In [11]:
```
print(type(s1))
```

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

In [13]:
```
s2[6]
```

Out[13]: 't'

In [14]:
```
s2[-1]
```

Out[14]: 'h'

In [16]:
```
s2[2:3]
```

Out[16]: 'e'

Boolean

In [1]:
```
true  #boolean is a case sensitive
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 1
----> 1 true

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

In [2]:
```
True
```

Out[2]: True

In [3]:
```
False
```

Out[3]: False

In [4]:
```
b= True
b
```

Out[4]: True

Complex Data type

In [5]:
```
a=1+2j
```

In [6]:
```
a
```

Out[6]: (1+2j)

In [7]:
```
type(a)
```

Out[7]: complex

In [8]:
```
print(type(a))
```

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

In [9]:
```
print(a.real)
```

```
1.0
```

In [11]:
```
print(a.imag)
```

```
2.0
```

type casting

In [12]:
```
print(int(20))
print(int(1.2))
print(int(25))
print(int(True))
print(int('20'))
```

```
20
1
25
1
20
```

In [13]:
```
print(bool(1+2j))
print(bool(1.2))
print(bool(25))
print(bool(True==False))
print(bool('20'))
```

```
True
True
True
False
True
```

In [ ]: