Topics:

1. how to check your python version
2. variable declaration
3. data types
4. Data type conversation

In [2]:
```
#  single line comment in python
'''
multiline
comment
'''
# in multiline comment each line is seperated by \n.
```

Out[2]:
```
'\nmultiline \ncomment\n'
```

sys ==> is a module

In [3]:
```
# to get the vesion of python
import sys
sys.version
```

Out[3]:
```
'3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]'
```

variable declaration

A python variable is a reserved memory location to store the values

In [5]:
```
a = 300
id(a) # id()===> it returns address of the variable.
```

Out[5]:
```
2267138262224
```

In [7]:
```
# we can't mention the values at very begining of the variable
name_student = 29
```

the following identifiers are used as rederved words , or keywords of the language.

In [8]:
```
import keyword
keyword.kwlist
```

Out[8]:
```
['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']
```

data types in python

- int
- float
- string
- boolean

In [9]:
```
value1 = 14233
value2 = 245.897
name = 'acer'
value3 = True
```

Few inbult functions

- print()
- input()
- len()
- help()
- int()
- float()
- type()
- id()
- bool()

In [10]:
```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

In [11]:
```
print(value1)
```

```
14233
```

In [13]:
```
value2
value3
```

Out[13]:
```
True
```

In [16]:
```
print(name, value3)
```

```
acer True
```

In [17]:
```
print(value1, value2, name, value3)
```

```
14233 245.897 acer True
```

sep ===> value seperater

In [18]:
```
print(a, value1, value2, name, value3, sep = '*****')
```

```
300*****14233*****245.897*****acer*****True
```

In [19]:
```
print(value1, value2, name, value3, sep = '----')
```

```
14233----245.897----acer----True
```

In [20]:
```
print(value1, end = '\n') # default
print(value2, end = '***')
print(value3)
```

```
14233
245.897***True
```

In [21]:
```
print('values are: ',value1, name)
```

```
values are:  14233 acer
```

In [22]:
```
help(input)
```

```
Help on method raw_input in module ipykernel.kernelbase:

raw_input(prompt='') method of ipykernel.ipkernel.IPythonKernel instance
    Forward raw_input to frontends

    Raises
    ------
    StdinNotImplentedError if active frontend doesn't support stdin.
```

In [23]:
```
b = input('enter a value: ')
print('value of b: ',b )
```

```
enter a value: 35
value of b:  35
```

In [24]:
```
# to identify type of a variable we can use type() function.
print(type(b))
print(type(a))
print(type(name))
```

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

In [25]:
```
print(type(value1))
print(type(value2))
```

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

type convertions

In [27]:
```
x = 567
```

In [28]:
```
x
```

Out[28]:
```
567
```

In [29]:
```
print(type(x))
```

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

In [30]:
```
x1 = int(b)
print(x1, type(x1))
```

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

In [31]:
```
x2 = input("enter any one number: ")
print(x2, type(x2))
```

```
enter any one number: 67.8
67.8 <class 'str'>
```

In [33]:
```
x3 = float(input("enter a number: "))
print(x3, type(x3))
```

```
enter a number: 67.8
67.8 <class 'float'>
```

In [35]:
```
print('length of a name: ',name, len(name))
```

```
length of a name:  acer 4
```

In [36]:
```
single = 'python'
double = "python"
triple = '''python'''
triple_double = """python"""
print(single, double, triple, triple_double)
```

```
python python python python
```

In [37]:
```
# bool()===> it returns True
e = bool(25)
print(e, type(e))
f = bool(0)
print(f)
g = bool(98)
print(g)
```

```
True <class 'bool'>
False
True
```

operators:

- arithimatic
- relational
- assignment
- logical
- bitwise
- identity
- membership

Arithimatic operators:

- +
- -
- *
- /
- //
- %
- **

In [38]:
```
# arithimatic
a = 10
b = 50
print('a+b: ',a+b)
print('a-b: ',a-b)
print('a * b: ',a * b)
print('a / b: ',a/b)
print('a // b: ',a//b)
print('a % b: ',a % b)
print('a ** b: ',a ** b)
```

```
a+b:  60
a-b:  -40
a * b:  500
a / b:  0.2
a // b:  0
a % b:  10
a ** b:  100000000000000000000000000000000000000000000000000
```

In [39]:
```
print(2 ** 5)
```

```
32
```

In [40]:
```
print(pow(2,5))
```

```
32
```

In [42]:
```
print(20*'welcome to the earth\n')
```

```
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
welcome to the earth
```

In [43]:
```
z = 'sai'; z1 = 'dragoon'
print('merge: ',z + z1)
```

```
merge:  saidragoon
```

In [44]:
```
# arithimatic on boolean.
print('True + True: ',True + True)
print('True + False: ',True + False)
print('False + True: ',False + True)
print('False + False: ',False + False)
```

```
True + True:  2
True + False:  1
False + True:  1
False + False:  0
```

In [45]:
```
print('True - True: ',True - True)
print('True - False: ',True - False)
print('False - True: ',False - True)
print('False - False: ',False - False)
```

```
True - True:  0
True - False:  1
False - True:  -1
False - False:  0
```

In [46]:
```
print('True * True: ',True * True)
print('True * False: ',True * False)
print('False * True: ',False * True)
print('False * False: ',False * False)
```

```
True * True:  1
True * False:  0
False * True:  0
False * False:  0
```

Relational operators:

- ==
- >
- <
- >=
- <=
- !=

In [47]:
```
a1 = 10
a2 = 20
print(a1 == a2)
print(a1 > a2)
print(a1 < a2)
print(a1 >= a2)
print(a1 <= a2)
print(a1 != a2)
```

```
False
False
True
False
True
True
```

Assignment operators:

- +=
- -=
- *=
- /=
- //=
- %=
- **=

In [66]:
```
a=5
a += 1
print(a)
```

```
6
```

In [69]:
```
a1 = 10
a1 -= 5
print(a1,type(a1))
```

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

In [70]:
```
b4 = 30
b4 *= 2
print(b4, id(b4))
```

```
60 140716834172560
```

In [71]:
```
r = 45
r /= 2
print(r, type(r))
```

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

In [72]:
```
r2 = 345
r2 //= 2
print(r2, type(r2))
```

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

In [73]:
```
c = 7
c %= 2
print(c, type(c))
```

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

In [74]:
```
y = 4
y **= 4
print(y, type(y))
```

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

Bitwise operators:

- bitwise &===> bitwise and
- bitwise |===> bitwise or
- bitwise ^===> bitwise xor
- bitwise >> ===> bitwise right shift
- bitwise << ===> bitwise left shift.

In [75]:
```
f1 = 6
f2 = 3
print('f1 & f2: ',f1 & f2)
print('f1 | f2: ',f1 | f2)
print('f1 ^ 2: ',f1 ^ 2)
print('f1 >> 2: ',f1 >> 2)
print('f1 << 2: ',f1 << 2)
```

```
f1 & f2:  2
f1 | f2:  7
f1 ^ 2:  4
f1 >> 2:  1
f1 << 2:  24
```

In [ ]:
```

```