Fundamentals01

February 8, 2024

1 Fundamentals of Python Class 01

In Python Everything is an object. Python is a multi-paradigm language, which means it supports multiple programming paradigms, not just OOP.

Pros - 1. Simple and easy to learn 2. High level 3. Freeware and open source 4. Platform Independent 5. Dynamically typed 6. Procedure oriented, OOPS, Scripting language 7. Interpreted 8. Fuctional Programming (lambda functions)

Cons - 1. Performance

1.1 Python is dynamically typed

Dynamically typed languages are often contrasted with statically typed languages, which require the programmer to declare the type of each variable at compile time, and perform type checking before the program is executed.

A dynamically typed language is a programming language that does not require the programmer to specify the data type of a variable when declaring it. Instead, the type of a variable is determined by the interpreter at runtime, based on the value assigned to it. This allows the programmer to write code faster and more flexibly, but it also increases the risk of type errors that are only detected during execution.

[1]:
```
a = 10
print(type(a))
```

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

[2]:
```
a = 12.3
print(type(a))
```

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

[3]:
```
a = True
print(type(a))
```

```
<class 'bool'>
```

[4]:
```
a = 'hello world'
print(type(a))
```

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

1.2 Python is Case sensitive

[5]:
```
abc = 10
Abc = 20
ABC = 30

print(abc)
print(Abc)
print(ABC)
```

```
10
20
30
```

1.3 Python Reserve words

[8]:
```
import keyword
print(keyword.kwlist)
print(len(keyword.kwlist))
```

```
['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']
35
```

Only 'False', 'None' and 'True' starts with capital letter, others start with small

1.4 Python Fundamental Data Types

1. int
2. float
3. string (str)
4. boolean (bool)
5. complex

Note: By default everything is in decimal (base 10) form in python and will print output in decimal (base 10) form regardless of input.

1.4.1 int

[9]:
```
a = 10
print(a)
```

```
10
```

[11]:
```
# 0B / 0b for binary
b = 0B1000
print(b)

c = 0b1001
print(c)
```

```
8
9
```

[12]:
```
# 0O / 0o for octal
d = 0O0634
print(d)

e = 0o4501
print(e)
```

```
412
2369
```

[15]:
```
# 0X / 0x for hexadecimal case insensitive
f = 0Xae89
print(f)

g = 0xBC34
print(g)
```

```
44681
48180
```

1.4.2 float

[16]:
```
a = 1200.00
print(type(a))
print(a)
```

```
<class 'float'>
1200.0
```

[19]:
```
b = 1.2e3
c = 1.4E4
print(b)
print(c)
```

```
1200.0
14000.0
```

1.4.3 complex

[21]:
```
a = 2+3j
print(type(a))
print(a)
```

```
<class 'complex'>
(2+3j)
```

[22]:
```
a = 2+4j
b = 4+9j
c = a+b
print(c)
```

```
(6+13j)
```

[23]:
```
c = a*b
print(c)
```

```
(-28+34j)
```

[24]:
```
c = a/b
print(c)
```

```
(0.4536082474226804-0.020618556701030934j)
```

[26]:
```
print(a.real)
print(type(a.real))
print(a.imag)
print(type(a.imag))
```

```
2.0
<class 'float'>
4.0
<class 'float'>
```

1.4.4 bool

[27]:
```
a = 10<20
print(a)
print(type(a))
```

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

[28]:
```
b = 20<10
print(b)
```

```
False
```

[29]:
```
print(a+b)
```

```
1
```

[31]:
```
print(a-b)
print(b-a)
```

```
1
-1
```

[32]:
```
print(True+True)
```

```
2
```

1.4.5 string

[34]:
```
a = "Sparsh"
print(a[0])
print(a[1])
print(a[2])
```

```
S
p
a
```

[35]:
```
print(a[-1])
print(a[-2])
print(a[-3])
```

```
h
s
r
```

[36]:
```
print(type(a))
print(type(a[-1]))
```

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

[37]:
```
b = "Sparsh's world of Python"
print(b)
```

```
Sparsh's world of Python
```

[38]:
```
c = """This is universal "Hey" , 'Hi'"""
print(c)
```

```
This is universal "Hey" , 'Hi'
```

[39]:
```
d = 'Sparsh\'s world of python'
print(d)
```

```
Sparsh's world of python
```

1.5 Multiline comment

[40]:
```
"""
hello this is a multi line comment
"""
```

[40]:
```
'\nhello this is a multi line comment\n'
```