Python - Functional Programming

BOOLEANS

Contents

- Booleans.
- Associated Truth Values.
- Precedence in bool.
- Short Circuiting.
- Operators
- Logics of Comparison Operators

Boolean

- **bool** class in python has a concrete implementation and usage.
- **bool** is a **subclass** of **int**.

- It posses all the properties and methods of integers, and add some specialized ones such as **and**, **or**,…
- Two constant values are defined : **True** and **False**.
- As we previously discussed python has many **Singleton** object. **True** and **False** are **Singleton** objects for type **bool**.

How to validate ?

issubclass(**bool, int**) → True

"**issubclass()** is a built-in function that returns **True** if a class supplied as the first argument is the subclass of another class supplied as the second argument, else it returns **False**."

isinstance(True, **bool**) → True

isinstance(True, **int**) → True

"**isinstance()** is a built-in function that returns **True** if the object is instance of classinfo argument or instance of classinfo subclass else returns **False**.

Note: Pythons Built-in Functions

is vs == recap

- Because **True** and **False** are **singleton** objects, they will always retain their same memory address throughout the lifetime of your application.
- So, comparisons of any Boolean expression to **True** and False can be performed using either the **is** (identity) operator, or the **==** (equality) operator.

- Lets say c_id = 2.
- c_id == **True** c_id is **True** when c_id is **bool** object

bool as int

- since **bool** objects are also **int** objects, they can also be interpreted as the integers **1** and **0**.
- If we do the below:
  - **int**(**True**) → **1**
  - **int**(**False**) → **0**
- **Note:**
  - **True** and **1** are not the same objects.
  - **False** and **0** are not the same objects.
- We can use **id** function to verify the above note.

Beware!! of bool as int

- **True > False** → True (Sounds like TENET no its not).
- (11 **==** 21) **== False** → True
- We can write above as :
- (11 **==** 21) **== 0** → True

- And all the operations of integer we can apply on bool
- **True + True + True** → **3**
- **-True** → **-1**
- **100 * False** → **0**
- (**True + True + True**) **% 2** → **1**

bool constructor

- The Boolean constructor **bool**(x) returns **True** when x is **True** and **False** when x is **False**.
- What really happens is that many classes contain a definition of how to cast instances of themselves to a Boolean, this is sometimes called the truth value.
- Integers have a truth value defined according to this rule:
  - **bool**(0) → False

Object Truth Values

- All objects in Python have an associated truth value.
- We already saw this with integers (although to be fair, bool is a subclass of int)
- But this works the same for any object
- Every object has a **True** truth value, except:
  - **None**
  - **False**
  - **0** in any numeric type (e.g. 0, 0.0, 0+0j, …)
  - empty sequences (e.g. **list**, **tuple**, **string**, …)
  - empty mapping types (e.g. **dictionary**, **set**, …)
  - custom classes that implement a **__bool__** or **__len__**

Behind the scene #1 False

- Classes define their truth values by defining a special instance method:
- **__bool__**(self) or **__len__**
- Then, when we call **bool**(x) Python will actually executes x.**__bool__**() or **__len__** if **__bool__** is not defined then by default its **True** .
- Examples:
- **bool**([1, 2, 3]) → **True**
- **bool**([]) → **False**

```
if my_list:
# code block
```

code block will execute if and only if my_list is both not None and not empty

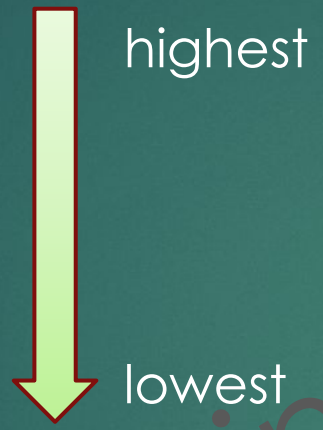
Operators

- The Boolean Operators: not, and, or:

P1	P2	not P1	P1 and P2	P1 or P2
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Precedence

- ( )
- > < >= <= == != in is
- not
- and
- or



True or **True** and **False**

True or **False** → True

(**True** or **True**) and **False**

True and **False** → False

use parentheses to make your code more readable!

Behind the Scene#2 Short-Circuit Evaluation

P1	P2	P1 or P2
0	0	0
0	1	1
1	0	1
1	1	1

if P1 is True, then P1 or P2 will be True no matter the value of P2 So, P1 or P2 will return True without evaluating P2 if P1 is True

P1	P2	P1 and P2
0	0	0
0	1	0
1	0	0
1	1	1

if P1 is False, then P1 or P2 will be False no matter the value of P2 So, P1 or P2 will return False without evaluating P2 if P1 is False

send me your suggestions!

- Email: sundar.muthuraman.offical@gmail.com
- Contact me : +91 9962988838