Python 101

Programming fundamentals.

Contents

Comments

The only “Standard” way (PEP8 suggested) of commenting in python is by using a # for a single line comment like follows-

This is a single line comment

If I need to give multiple line comment I will give multiple single line comments.

A non standard way is by using doc strings.

https://github.com/django/django/blob/master/django/core/signing.py line 126

Comments contd..

Comments are used for telling others who will see your code that what a certain part of your code (code-segment) does.

Comments exist only in the file that you write. They are completely ignored when compiled.

Comments Practical.

Create a file named **comment_test.py**

Write following code in it

```
# This is a single line comment

import this
```

Save the file and compile it using py_compile. On checking the compiled file we can see that “This is a single line comment” doesn’t exist in our compiled bytecode.

Tokens

A token is the smallest part of a program that is significant to compiler/interpreter.

Keywords

Keywords are reserved words in a language.

Eg. in Python there are *def, if, else, None, True etc.*

But who can remember all the keywords right?

There are 33 of them.

Keywords Practical.

Open the Python Interpreter and write following code in it.

```
import keyword
```

```
keyword.kwlist
```

So You don’t need to remember the keywords whenever you need it you can see it by typing two lines.

Identifiers

Identifier is name of literally anything created by user, be it variable, function, class or object. Identifiers can only contain (A-Z a-z 0-9 or _) and must not start with digits.

There are some special Identifiers as well, just like _

_ contains the last value that was evaluated (printed out on terminal) in the interpreter.

Identifiers Practical

Open terminal

type _

It shows some error which tells us that _ is not defined yet.

Ofcourse it is not defined because nothing has been calculated yet.

Now just type any number e.g. 4,5,6,7 (Press enter)

Now type _ and it will print the last value that you evaluated.

Try the same by typing 4+5 on terminal and see value of _ now

Literals

Literals are the values that are meant to be interpreted exactly as they were written. These are notations for values of some types in python.

These are often the values that we assign to variables. We will discuss variables after we have discussed Literals.

There are a few literals used in Python, eg.

{ }, 1, -1, 1.1, None, True, False, “String”, ‘string’, b‘string with byte flag’, r‘string with raw string flag’, u’string with unicode flag’, f’string with format flag’, [ ], (), {1},

Literals Practical

There is a built-in function in Python called type(some_object).

It tells the type of the object you pass to it. As we already know that everything in python is object lets try knowing the types of the literals we just saw in example.

Open the Python interpreter. Enter the code given in next slide.

Try out the following code on the interpreter

```
>>> type(1)
<class ‘int’>
>>> type(1.1)
<class ‘float’>
>>> type(“String”)
<class ‘str’>
>>> type({})
<class ‘dict’>
>>> type([ ])
<class ‘list’>
>>> type(())
<class ‘tuple’>
>>> type({1})
<class ‘set’>
>>> type(None)
<class ‘NoneType’>
>>> type(True)
<class ‘bool’>
```

Variables

Variables are something that hold a value.

They are like containers.

We often assign literals to Variables for using them.

Variable names follow the convention for names of identifiers.

Variables Practical

Open Python Interpreter and type following.

```
>>> some_variable = 123

>>> some_variable

123
```

As per PEP 8 variable names should be all lowercase and words seperated by _ (underscore).

Variables Type and Dynamic Typing

Variables in other languages that are “Statically Typed” have their own defined Type. e.g. in Java String x; can only hold String values.

In Python it’s different, In Python Variables are “Dynamically Typed’ ie. The variable changes its type to whatever it has been assigned.

Dynamic Typing Practical

```
>>> x=1
>>> type(x)
<class ‘int’>
>>> x = 1.1 <- mind that there has been no error at this point. instead x changed its type to float.
<class ‘float’>
```

Operators

Operators are symbols that tell the compiler to perform a certain operation which might be Logical, Arithmetic, Relational, Bitwise, Assignment.

Arithmetic Operators

+, - *, / (float div), // (int div) , %, ** (exponent)

1+2 = 3

1-2 = -1

1*2 = 2

1 / 2 = 0.5

1//2 = 0

1%2 = 1

1**2 = 1 (1*1)

Relational Operators

==, >, < , <=, >=, != . These return values in True or False

1 == 2 False

1 > 2 False 1>=2 False (greater than OR equal to)

1 < 2 True 1 <= 2 True (less than OR equal to)

1 !=2 True (1 not equal to 2)

Bitwise Operators

& (and) , | (or), ^ (xor), ~ (complement), << (left shift), >> (right shift)

1 & 2 = 0

1 | 2 = 3

1 ^ 2 = 3

~1 = -2 (-1 -1)

1<<2 = 4 4>>1 = 1

Assignment Operators

A bit back when we talked about Variables we said Variables are like containers and they hold values, thus there should be some mechanism using which we would be able to assign literals or other values to the variables.

That mechanism is called assignment and in order to assign a value to a variable we use assignment operators.

Assignment Operators

=, +=, -=, *=, %=, **=

x = 1 (assign the literal value 1 that is integer to a variable x)

x += 1 (x= x+1)

x *= 1 (x = x*1) and so on.

For +=, -= , *= , %= , **= to work without any error the variable must be holding some value beforehand as in case it doesn’t hold any value before using these operators it would mean x = not defined + 1 which is impossible to perform.

Logical Operators

and , or , not

True and False = False

True or False = True

not True = False

Character Encoding

Computer only knows numbers. But how can we represent alphabets using numbers?

Character encoding is the method by which computer comes to know how to interpret the raw 0s and 1s into characters.

We are familiar with ASCII. But it has limited support for characters.

So Python uses more extensive support for characters that is Unicode. Unicode has two representational standards that are UTF and UCS. UTF is much more popular due to its flexibility than UCS.

Character Encoding Practical

There are two base functions chr and ord that we will study here, rest of the coding and decoding will be discussed in strings.

chr(integer_value) prints the unicode character corresponding to that integer value. eg. chr(8377) gives out ₹

ord works just opposite to this it gives us corresponding integer to a character.

eg. ord('a') gives out 97

Interview Questions

Are there multiline comments in Python?

No,

But we can implement the functionality using doc strings.

Distinguish Keywords and Literals.

Keywords are reserved Tokens in a Programming language.

Literals are the values which are meant to be interpreted exactly as they are written.

Eg. 1,2,3,4 are literals and True, False, and, or, not are Keywords.

Is Python Dynamic Typed? Can we also called in weakly typed?

Yes Python is Dynamically Typed.

No we can not call it weakly typed. In weakly typed language type coercion happens amongst two non compatible types as well. Javascript is a prime example of that.

e.g. 9+ ‘9’ is ‘99’ in JS whereas in Python it would raise errors.

What is the purpose of _ identifier?

It holds the last interpreted value as a simple cache mechanism. When working in the interactive mode in interpreter.

Is Python Case sensitive? Tell me a few programming languages that are Case-insensitive.

Yes. Python is Case sensitive.

BASIC, FORTRAN, and PASCAL are notably Case insensitive.

What is the difference between / and //

/ is float division ie.1 /2 would result in 0.5

Whereas // is integer division ie. 1//2 would result in 0.

What does type function do in Python?

type(some_object) returns the type of the object we pass to this function.

What do you understand by Character Encoding?

Character encoding is the method by which computer comes to know how to interpret the raw 0s and 1s into characters.