Problem Solving with Python

Priyansh Singh

Development Environment?

- IDLE (Integrated Development Language Environment)
- Python Shell
- Any Text Editor like vim, Sublime Text, Visual Studio Code, Notepad

Identifiers

- Identify a variable, function, class, module or other object.

- An identifier starts with a letter A to Z or a to z or an underscore (_) followed by letters, digits (0 to 9) or underscores.

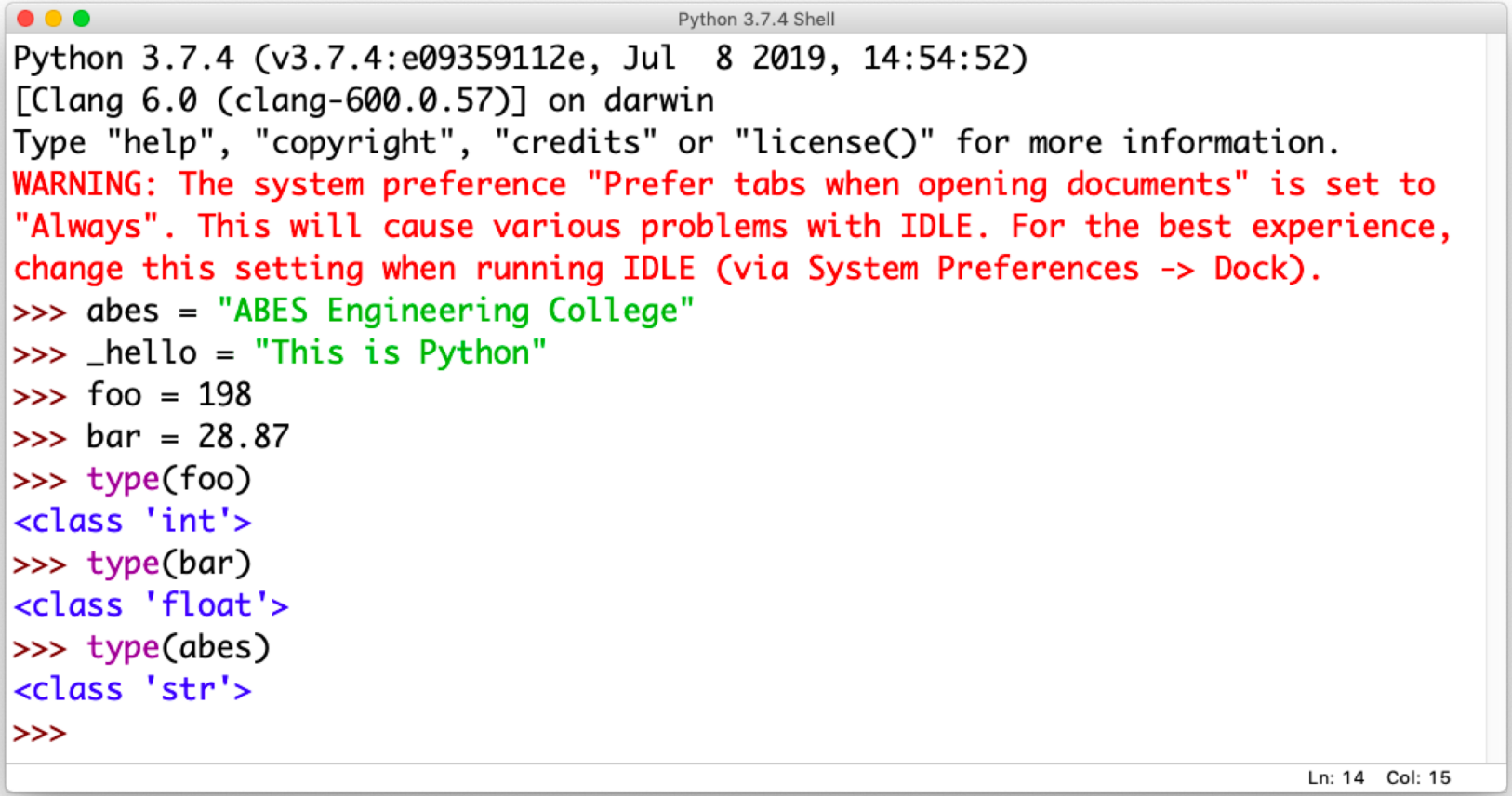
1. The identifiers are case sensitive.

2. Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

1. Starting an identifier with a single leading underscore indicates that the identifier is private.

1. Starting an identifier with two leading underscores indicates a strongly private identifier.

1. If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.


Python 3.7.4 Shell
Python 3.7.4 (v3.7.4:e09359112e, Jul 8 2019, 14:54:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The system preference "Prefer tabs when opening documents" is set to "Always". This will cause various problems with IDLE. For the best experience, change this setting when running IDLE (via System Preferences -> Dock).
>>> abes = "ABES Engineering College"
>>> _hello = "This is Python"
>>> foo = 198
>>> bar = 28.87
>>> type(foo)
<class 'int'>
>>> type(bar)
<class 'float'>
>>> type(abes)
<class 'str'>
>>>
Ln: 14 Col: 15

Blocks of Code and Indentation

No braces to indicate blocks of code for class and function definitions or flow control.

Blocks of code are denoted by line indentation, which is rigidly enforced.

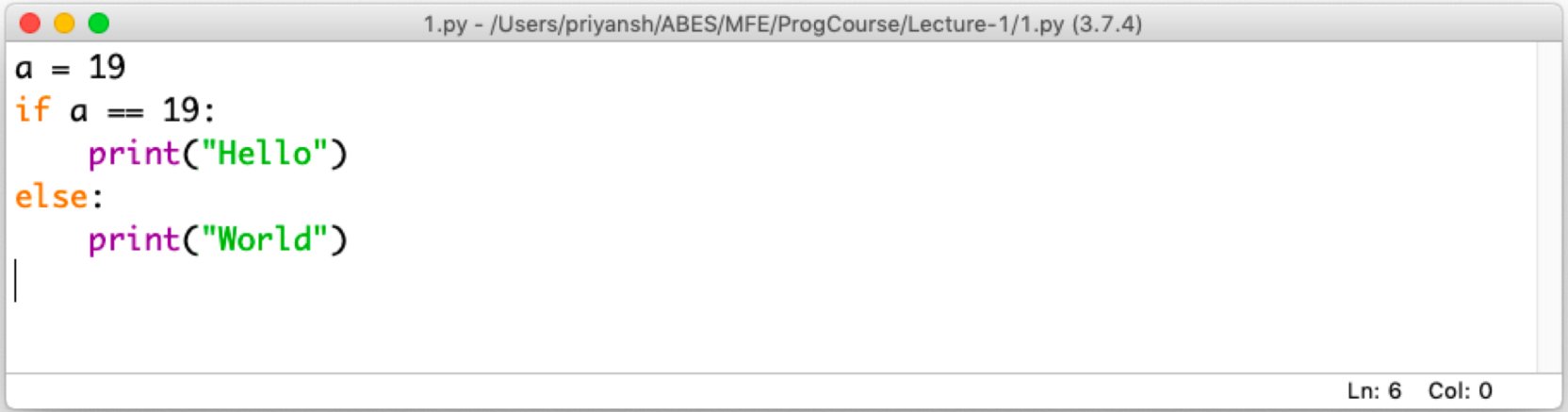
Spaces in the indentation is variable, but all statements within the block must be indented the same amount.

Statement Suites

- A group of individual statements, which make a single code block are called suites in Python.

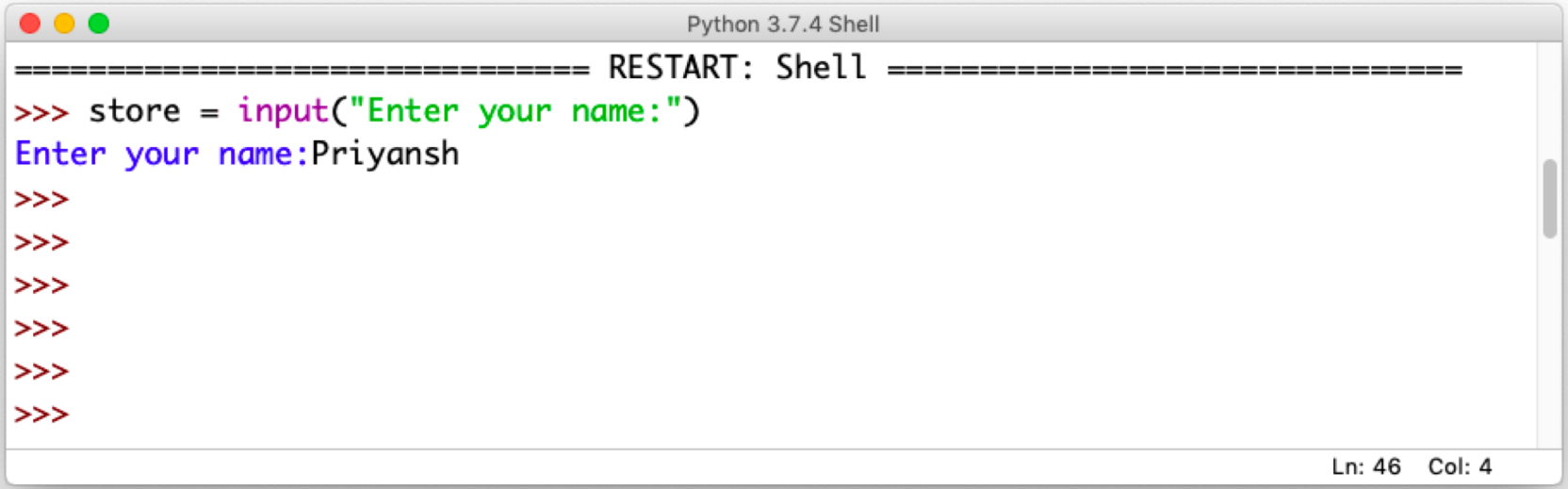
- Compound or complex statements, such as if, while, def, and class require a header line and a suite.

- Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite.


1.py - /Users/priyansh/ABES/MFE/ProgCourse/Lecture-1/1.py (3.7.4)
a = 19
if a == 19:
print("Hello")
else:
print("World")
Ln: 6 Col: 0

Input and Output

The function input() reads and stores input from STDIN. The input() function can also be called with a text message. The message prompts the user for input.


Python 3.7.4 Shell
========== RESTART: Shell ==========
>>> store = input("Enter your name:")
Enter your name:Priyansh
>>>
>>>
>>>
>>>
>>>
>>>
Ln: 46 Col: 4

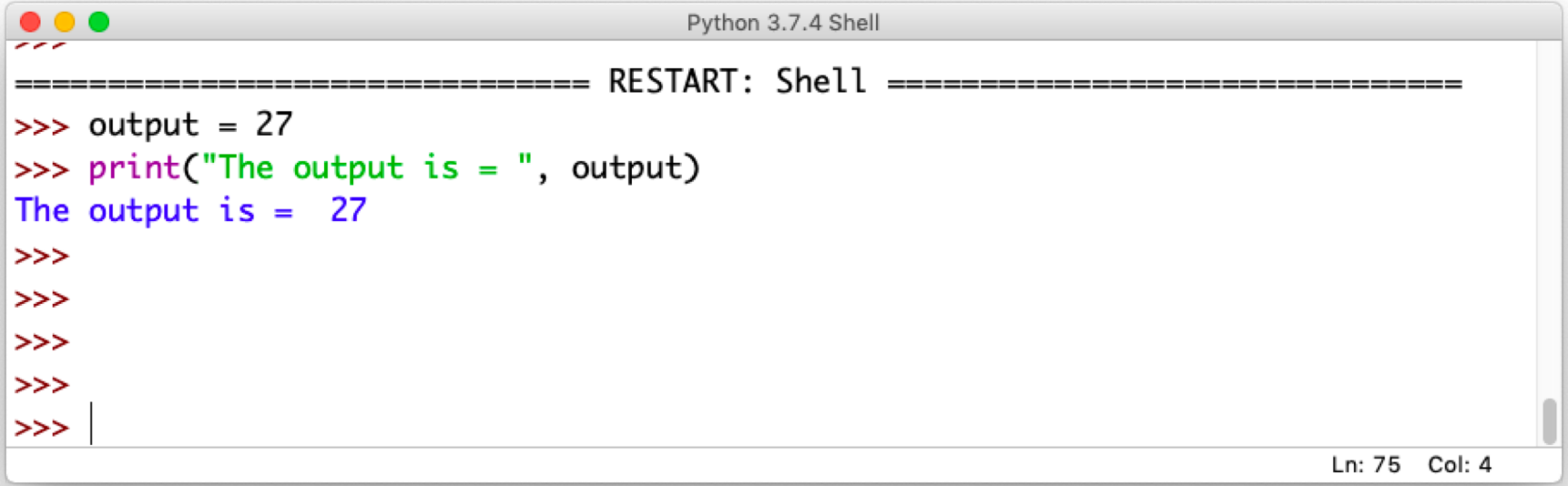
Input and Output

The input is always read as a string.

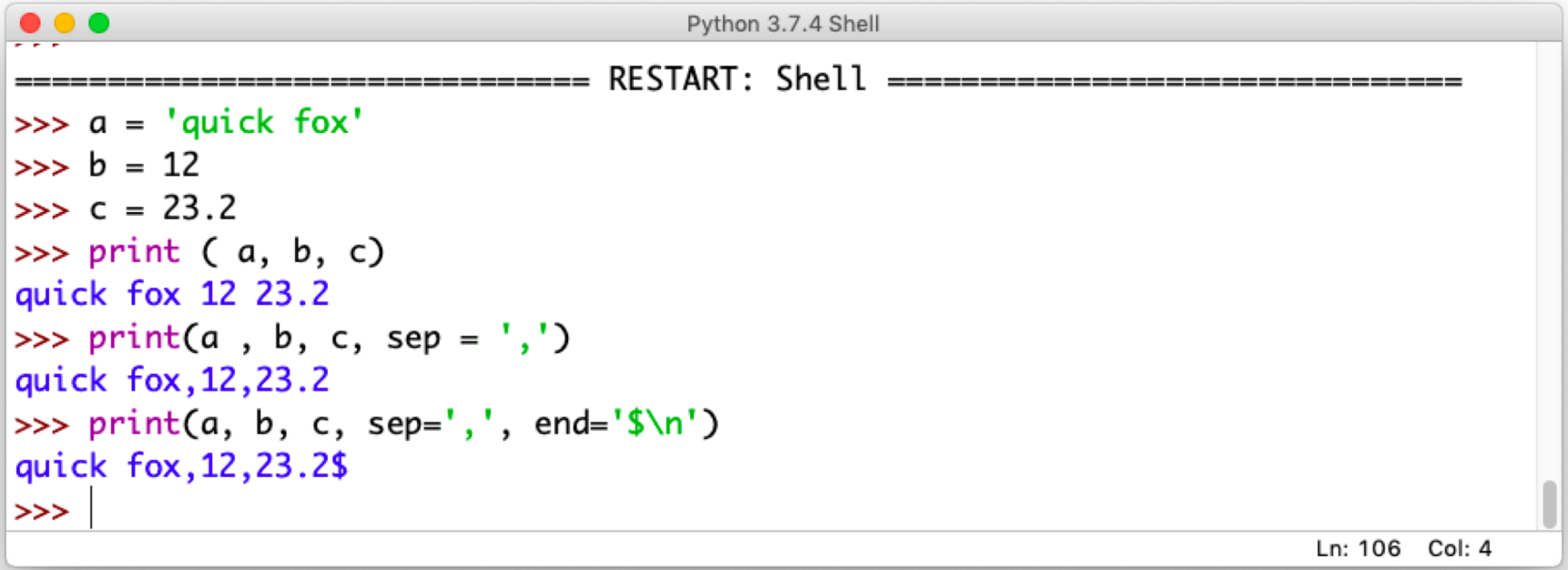
The input string can be converted to an an integer or float using the int() or float() function respectively.

Input and Output

The print() function prints data to STDOUT.



Input and Output



Operators in Python

+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
%	Modulus. Divides left hand operand by right hand operand and returns remainder	a % b
**	Exponent	a ** b
//	Floor division. Divides left hand operand with right hand operand, rounds off the result to the closest but lower integer	9//4 will yield 2 and -9/4 will yield -3.

Arithmetic Operators

Operators in Python

==	Returns True if both operands are equal.	a == b
!=	Returns True if both operands are not equal.	a != b
>	Returns true if left operand is more than right operand.	a > b
<	Return true if right operand is more than left operand.	a < b
>=	Return true of left operand I more than or equal to right operand.	a > = b
<=	Returns true if left operand is less than or equal to right operand.	a <= b
is	Object comparison. Evaluates to true if the names on either side of the operator point to the same object and false otherwise.	x is y essentially checks the identity (as returned by the id() function) of the two objects – x and y.
is not	Opposite of above.	Opposite of above.

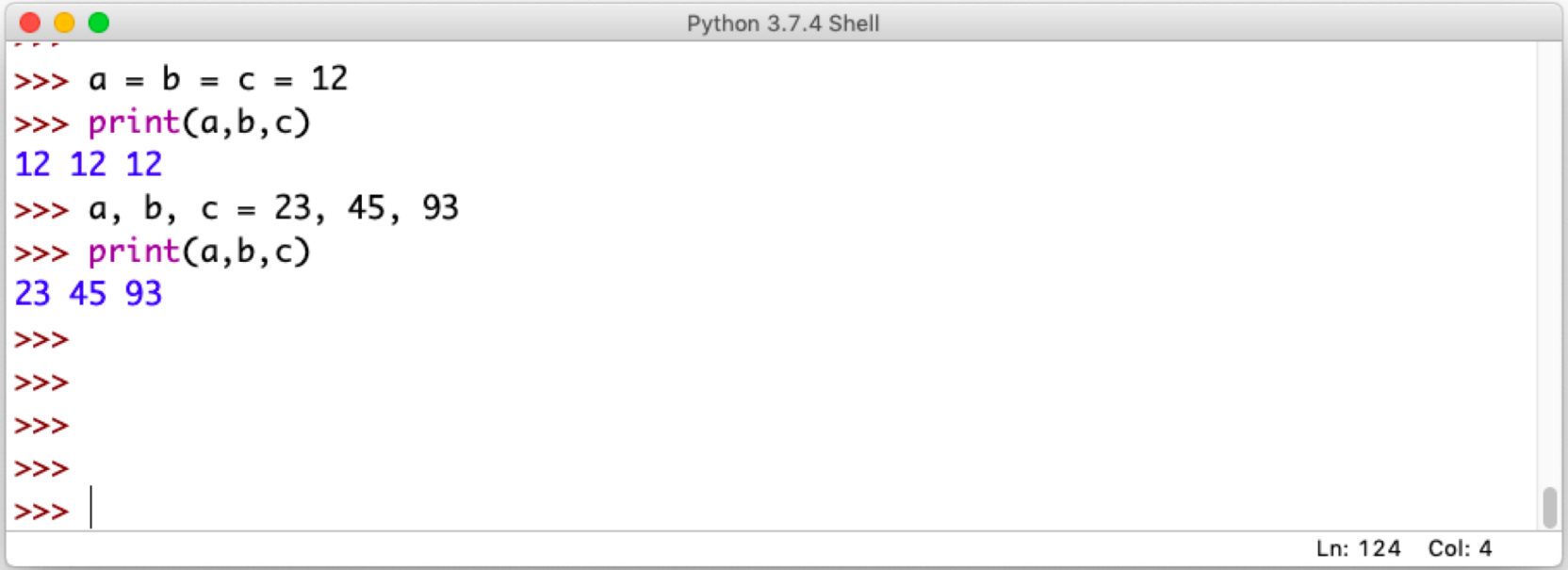
Comparison Operators

Operators in Python

=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+=	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-=	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*=	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/=	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c/a
%=	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**=	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//=	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Assignment Operators

Multiple Assignments



Operators in Python

& Binary AND	Operator copies a bit to the result if it exists in both operands	a & b = 12 (0b0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (0b0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (0b0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (-0b11110100)
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (0b1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (0b0000 1111)
& Binary AND	Operator copies a bit to the result if it exists in both operands	a & b = 12 (0b0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (0b0011 1101)

Bitwise Operators

Operators in Python

and Logical AND	If both operands are true then condition becomes true.	(a and b) is False.
or Logical OR	If any of the two operands are true then condition becomes true.	(a or b) is True.
not Logical NOT	Used to reverse the logical state of its operand.	not (a and b) is True.

Logical Operators

Membership Operators

Membership operators test for membership in a sequence, such as strings, lists, tuples, dictionaries and sets.

If applied over a dictionary it checks for the existence of the given element in the keys of the dictionary.

in	Evaluates to True if it finds a variable in the specified sequence and False otherwise.
not in	Evaluates to True if it does not find a variable in the specified sequence and False otherwise.

Truth value of Objects

In Python any object can be tested to be True or False for use in an if or while or Boolean operations.

The following objects are considered False.

- None
- False.
- Zero of any numeric type – 0, 0.0 and 0j.
- Any empty sequence – '' (empty string), () (empty tuple), [] (empty list).
- Empty mappings – {} (empty dictionary).

All other objects are considered to be True.

Python Built in Data Types

The built in data types can be categorized in the following categories –

1. Numeric Types (integers, float and complex)
2. Sequence Types (String, List, Tuple andRange)
3. Mapping Types (Dictionary) and
4. Set Types (Sets and Frozensets)

Range

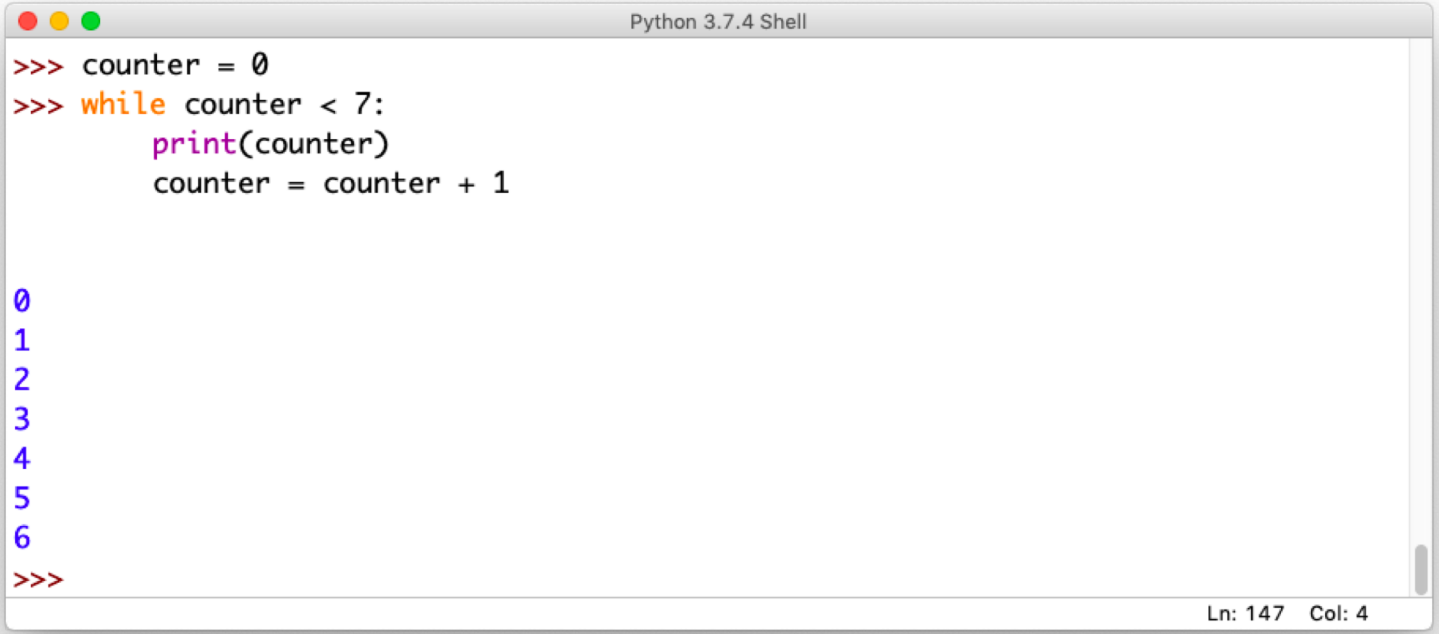
Objects of type range are created using the range() function. This is a versatile function to create iterables yielding arithmetic progressions.

```
range(start, stop, step)
```

Range

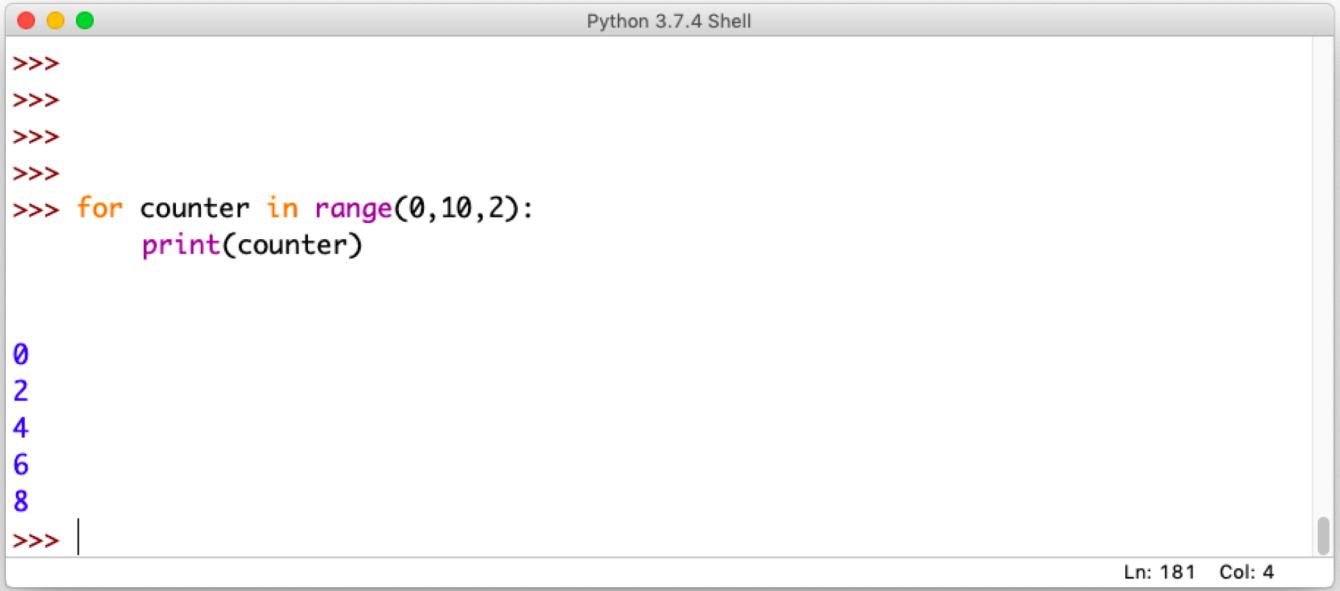
range(4, 16, 3)	Generates the sequence: 4, 7, 10, 13
range(4, 16, -3)	Makes no sense. Will not generate anything
range(16, 4, -3)	Generates the sequence: 16, 13, 10, 7
range(4, 16, -3)	Makes no sense. Will not generate anything.
range(10)	Generates 0, 1, 2, … 8, 9
range(-5)	Makes no sense. Will not generate anything

While Loop Construct



For Loop Construct

The for statement iterates over the items of any sequence (list, string, range among others), in the order that they appear in the sequence.



Practice Questions

Q1. Write a program to add two objects if both objects are an integer type.

Practice Questions

Q2. Write a program to find the numbers between 1500 and 2700, which are divisible by 7 and are multiples of 5.

- *Start with hardcoding the given numbers.*
- *Modify the above program to seek input from the user.*
- *Modify the above program to cut down the number of comparisons.*

Practice Questions

Q3. Write a program to find numbers between 100 and 400 (both included) where each digit of a number is an even number. The numbers obtained should be printed in a comma-separated sequence.

- *Start with hardcoding the given numbers.*
- *Modify the above program to seek input from the user.*
- *Modify the above program to cut down the number of comparisons.*

Practice Questions

Q4. Write a program that accepts a positive number and subtract from this number the sum of its digits. Keep repeating this process until the number becomes negative or zero. Count the number of subtractions.

- *First consider any number do brute force subtractions*
- *Instead of subtracting any sum, repeat the process of finding sum of digits till the number is single digit.*
  97 => 9 + 7 => 16 = >1 + 6 = 7.
  Subtract this number repeatedly till number becomes zero or negative.