

C9418: Intro Programming in Python

Spark 2015

Jordan Moldow (jmoldow@alum.mit.edu)

Mar. 15, 2015

1 Introduction to Programming

Programming

- A program is “a sequence of coded instructions for a computer”
- Programming is the coding of these instructions by humans
- “The purpose of programming is to create a program that exhibits a certain desired behavior.”
- Programming is “writing the source code of computer programs”

General Programming Steps

1. Pick a programming language
2. Write “source code” inside a text file
 - Source code is understandable by humans [who know the language]
 - Each language has different code syntax
3. (For compiled languages) A “compiler” translates source into binary / machine code that is understandable by computers
4. Computer executes code

Writing Python Programs

- Code can be written and saved using special programming environments – file type is .py
- Code can also be written in a normal text editor
 - Notepad, Notepad++, vim, emacs, gedit, textedit
 - NOT Word, OpenOffice, LibreOffice
- We will use Python IDLE, an officially supported integrated development environment
- Python interpreter executes code directly from your source code – Python is an **interpreted language**

2 Introduction to Python

Python Interpreter

- The first thing you see after opening Python IDLE is a command prompt
- This is a shell for the Python interpreter
- Go ahead and type stuff into it
- In its most basic form, the interpreter acts like a calculator, supporting all basic mathematical operations and orders of operations
- Of course, the shell is infinitely more powerful than this, and we will slowly build up our knowledge of what Python can do

Writing and Saving Programs

- No code you write into the interpreter is permanent – it will be lost when you close the interpreter
- You can save code into a file so that you can run it whenever you want
- In Python IDLE, File -> New Window opens a Python file, which you can write code into, save, and run

Hello World! Your First Program!

- A programming tradition – your first program simply outputs the text `Hello World!`
- “Output”, in this and most cases, means to write text on the screen

```
1 # Program: hello.py
2
3 print "Hello World!"
```

Basic Python syntax

- Python is CASE SENSITIVE!
 - This means that `Print "Hello World!"` is WRONG
- `#` starts a comment
 - Everything on the line after the `#` is the comment
 - Comments have no effect on the program
 - Use them so others can understand your program
- `"` starts and ends a string
 - A string is a sequence of characters
 - If you want the quote character, use `\`
 - * `"\"Hello World!\""` is the string consisting of the characters `"Hello World!"`
- Programs are made up of one-line statements:

```
1 do_this_first
2 then_do_that
3 finally_do_something_else
```

The `print` Statement - Part 1

- This statement is used for outputting text on the screen
- `print "Hello World!"` outputs Hello World!
- `print "text"` outputs text (literally)
- Don't forget the space after `print`, and the quotation marks!
- The enclosing quotation marks don't show up in the output
- After the text, a line break is output
- Can include line break in string with `\n` character

3 Expressions and Variables

So wait, can Python do anything besides print messages?

- Yes, it can!
- Python can calculate the results of expressions
- Python can store and manipulate data using variables

Literals

- The building blocks of expressions
- A basic representation of a simple value
- Integer literals - 0, 17, -10, etc.
- Floating point literals - 1.0, 3.14159, etc.
- String literals - "Hello World!", etc.
- Boolean literals - True, False

The `print` Statement - Part 2

- Can be used to print any literal

```
1 print 17
2 print 3.14159
3 print "Hello World!"
4 print True
5 print False
```

Arithmetic Expressions

Addition (+)	$17+5 \Rightarrow 22$	Subtraction (-)	$17-5 \Rightarrow 12$
Multiplication (*)	$17*5 \Rightarrow 85$	Division (/)	$17/5 \Rightarrow 3$
Modulus (%)	$17\%5 \Rightarrow 2$	Parenthesis (())	$(17+5)*2$
Negative (-)	$-(17+5)$		

The print Statement - Part 3

- Can be used to print any expression

```
1 print 17 + 5
2 print 17 - 5
3 print 17 % 5
```

- Can print multiple expressions on one line

```
1 print "The value of 17 + 5 is", 17 + 5
```

- IDLE shell can print expressions without typing print

Logical (Boolean) Expressions

Equality (==)	17==5 => False
Inequality (!=)	17!=5 => True
Greater than (>)	17>5 => True
Greater than or equal (>=)	17>=5 => True
Less than (<)	17<5 => False
Less than or equal (<=)	17<=5 => False

```
1 print 17 == 17
2 print 17 == 5
3 print 17 != 5
4 print 17 > 5
5 print 17 <= 5
6 print 17 == (12 + 5)
7 print True == True
8 print True == False
```

Variables

- Can store values into memory locations
- Reference this memory with **variables**

```
1 variable = expression
```

- Computes value of expression, and **assigns** it to variable

```
1 temperature = 50
2 average = (17.5 + 73.9) / 2
3 temperature = temperature - 10
```

- In the last example, the expression value overwrites the old stored value in memory
- Variable name must start with a letter, consists of letters, numbers, and underscores

The print Statement - Part 4

- Variables can be used as values, and used in expressions
- So print can display stored values

```
1 temperature = 50
2 print temperature
3 print temperature - 10
```

User input

```
1 name = raw_input("What is your name? ")
2 print "Your name is", name
3
4 temperature = input("What is the temperature? ")
5 print "That is", temperature - 32, "above freezing"
```

Coding Challenge

- Write code to take two numbers of user input, add them together, and print the result.
- Write code to take the temperature in fahrenheit and print it in celsius.

$$- C = \frac{F - 32}{1.8}$$

4 Control Flow

Conditional Execution with **if**-statements

- Execute a block of code only if an expression is `True`.

```
1 temperature = input("What is the temperature? ")
2 print "The temperature is", temperature
3 if temperature < 32:
4     print "It is below freezing!"
5     print "Don't forget to wear your jacket!"
```

- Those messages will only print when the temperature is below 32
- `if`, followed by the true/false expression, followed by a colon
- The conditional block must be indented

Conditional Execution with **else**-statements

- Execute a block of code only if the immediately preceeding `if`-statement was `False`

```
1 temperature = input("What is the temperature? ")
2 print "The temperature is", temperature
3 if temperature < 32:
4     print "It is below freezing!"
5 else:
6     print "It is", temperature - 32, "degrees above freezing"
```

- `if`-statement and block, followed by un-indented `else:` (with colon)
- The conditional block must be indented

Conditional Execution with `elif`-statements

- Execute a block of code only if all the immediately preceding `if` and `elif`-statements were `False`

```
1 temperature = input("What is the temperature? ")
2 print "The temperature is", temperature
3 if temperature < 32:
4     print "It is below freezing!"
5 elif temperature == 32:
6     print "We're at the freezing point!"
7 elif temperature < 100:
8     print "It is", temperature - 32, "degrees above freezing"
9 else:
10    print "It is really hot!"
```

- Un-indented `elif`, followed by the true/false expression, followed by a colon
- The conditional block must be indented

Coding Challenge

- Write code to take two numbers of user input, ask the user for an operation (addition, subtraction, etc.), and print the result.
- Write code to take the temperature in fahrenheit and print it in celsius, or do the reverse, depending on user input.

$$- C = \frac{F - 32}{1.8}$$

$$- F = (1.8 \times C) + 32$$

More Learning Resources

- <https://docs.python.org/2/>
- <https://docs.python.org/2/tutorial/index.html>
- <https://www.python.org/downloads/release/python-279/>
- <http://www.codecademy.com/en/tracks/python>