C10156: Intro Programming in Python

Splash 2015

Jordan Moldow (`jmoldow@alum.mit.edu`)

Nov. 22, 2015

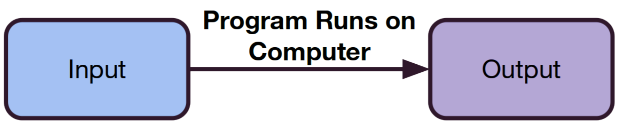
Learning Goals for this Lesson

1. Know what programming is
2. Have an idea of the power of programming
3. Know how to get started with a Python coding environment
4. Be able to write simple Python programs that print text
5. Be able to use expressions to calculate values from multiple pieces
6. Be able to assign to, and manipulate, variables
7. Be able to work with user input
8. Be able to use conditionals to control the flow of programs
9. Apply all of the above concepts to solve math challenges or create simple games

1 Introduction to Programming

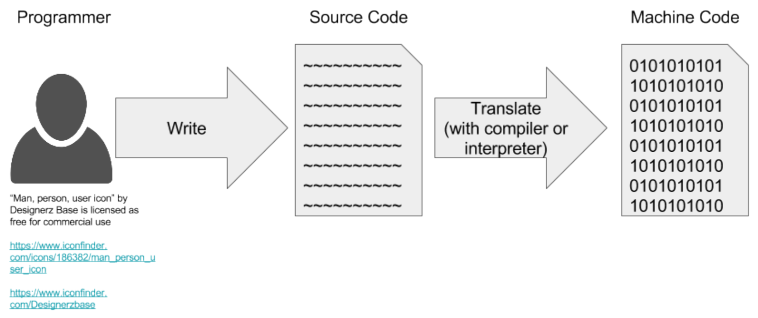
What does a program do?

- Provides instructions to a computer
- Produces output
  - Print text to console
  - Write to file
  - Draw on screen
- Acts on input
  - Keyboard presses
  - Mouse clicks
  - File contents



What does a programmer do?

- Write human-readable "source code"
- Use another program to interpret that as machine instructions

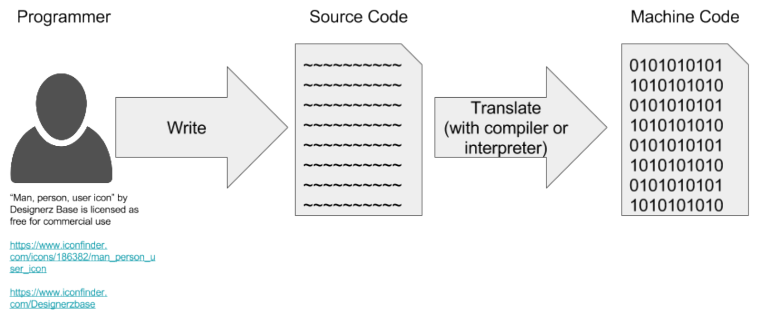

Programmer
Source Code
Machine Code
Write
Translate (with compiler or interpreter)
~~~~~~~~~~~
0101010101
1010101010
"Man, person, user icon" by Designerz Base is licensed as free for commercial use
https://www.iconfinder.com/icons/186382/man_person_user_icon
https://www.iconfinder.com/Designerzbase

Why do we care about programming?

Google docs
Google
YouTube
NETFLIX
Stanford SPLASH!
education for students, by students

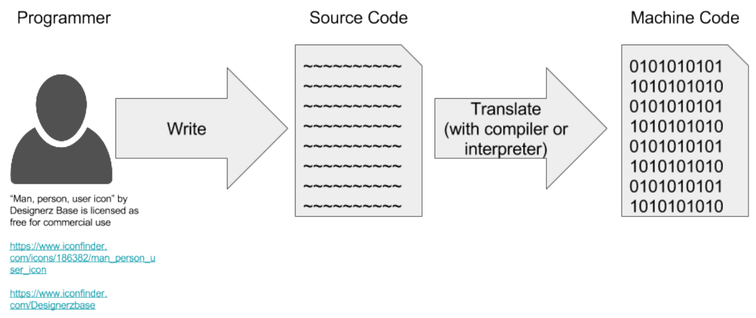
General Programming Steps

1. Pick a programming language
2. Write "source code" inside a text file
3. Use "compiler" or "interpreter" to translate source into binary / machine code that is understandable by computers
4. Computer executes code


Programmer
Source Code
Machine Code
Write
Translate (with compiler or interpreter)
~~~~~~~~~~~
0101010101
1010101010
"Man, person, user icon" by Designerz Base is licensed as free for commercial use
https://www.iconfinder.com/icons/186382/man_person_user_icon
https://www.iconfinder.com/Designerzbase

Compilers vs. Interpreters

- Compiler outputs machine code into new file
  - This binary file is executable
- Interpreter immediately executes machine code
  - The source code file is executable by interpreter
  - Python is an interpreted language


Programmer
Source Code
Machine Code
Write
~~~~~~~~~~~
Translate (with compiler or interpreter)
0101010101
1010101010
"Man, person, user icon" by Designerz Base is licensed as free for commercial use
https://www.iconfinder.com/icons/186382/man_person_user_icon
https://www.iconfinder.com/Designerzbase

Python Programs

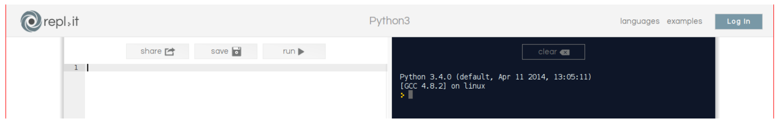
- Source code file type is .py
- Code is written in a text editor
  - Notepad, Notepad++, vim, emacs, gedit, textedit, etc.
  - NOT Word, OpenOffice, LibreOffice
- Use the program called python (the interpreter) to execute code
- Optionally, an IDE can do both steps
  - Python IDLE
  - Web IDEs, e.g. `https://repl.it/languages/python3`

2 Introduction to Python

Getting Set Up

Instructions

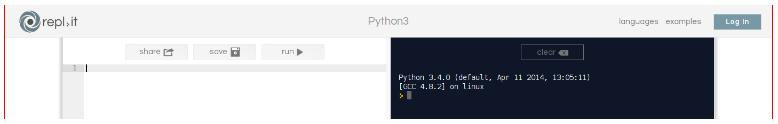
1. Log in to your computer
2. Open the web browser
3. Go to `https://repl.it/languages/python3`
4. Type something on line 1 in the left box
5. Press "save"
6. Email the link to yourself, or write it down



```
Python 3.4.0 (default, Apr 11 2014, 13:05:11)
[GCC 4.8.2] on linux
>
```
~~~~~~~~~~~

repl.it Python Interpreter

- Left side is source code, right side is interactive interpreter
- Type stuff into the right and press “Enter” key
- Type stuff into the left and press “run” button
  - Don’t forget to press “save” button periodically
- In its most basic form, the interpreter acts like a calculator, supporting all basic mathematical operations and orders of operations
- Of course, Python is infinitely more powerful than this, and we will slowly build up our knowledge of what it can do



```
Python 3.4.0 (default, Apr 11 2014, 13:05:11)
[GCC 4.8.2] on linux
>
```

Writing and Saving Programs

- No code you write into the interpreter on the right is permanent – it will be lost when you re-run programs from the left
- For simple one-line statements, use the interpreter on the right to try them out
- For anything longer, write it into the program window, then “save” and “run”

```
Python 3.4.0 (default, Apr 11 2014, 13:05:11)
[GCC 4.8.2] on linux
>
```

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

Instructions

1. Copy this program into the program window on the left

```
# Program: hello.py

print("Hello World!")
```

2. Press “save”
3. Press “run”

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:

```
do_this_first
then_do_that
finally_do_something_else
```

The `print` Function - Part 1

- This function is used for outputting text on the screen
- `print("Hello World!")` outputs `Hello World!`
- `print("text")` outputs `text` (literally)
- Don't forget the parentheses 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` Function - Part 2

- Can be used to print any literal

```
print(17)
print(3.14159)
print("Hello World!")
print(True)
print(False)
```

Arithmetic Expressions

Addition (+)	`17+5`	`22`
Subtraction (−)	`17-5`	`12`
Multiplication (*)	`17*5`	`85`
Division (/)	`17/5`	`3.3999999999999999`
Integer Division (//)	`17//5`	`3`
Modulus (%)	`17%5`	`2`
Parenthesis (())	`(17+5)*2`	`44`
Negative (−)	`-(17+5)`	`-22`

The `print` Function - Part 3

- Can be used to print any expression

```
print(17 + 5)
print(17 - 5)
print(17 % 5)
```

- Can print multiple expressions on one line

```
print("The value of 17 + 5 is", 17 + 5)
```

- Interactive interpreter 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`

```
print(17 == 17)
print(17 == 5)
print(17 != 5)
print(17 > 5)
print(17 <= 5)
print(17 == (12 + 5))
print(True == True)
print(True == False)
```

Variables

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

```
variable = expression
```

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

```
temperature = 50
average = (17.5 + 73.9) / 2
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` Function - Part 4

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

```
temperature = 50
print(temperature)
print(temperature - 10)
```

User input

```
name = input("What is your name? ")
print("Your name is", name)

temperature = int(input("What is the temperature? "))
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`.

```
temperature = int(input("What is the tempurature? "))
print("The temperature is", temperature)
if temperature < 32:
    print("It is below freezing!")
    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`

```
temperature = int(input("What is the temperature? "))
print("The temperature is", temperature)
if temperature < 32:
    print("It is below freezing!")
else:
    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 preceeding `if` and `elif`-statements were `False`

```
temperature = int(input("What is the temperature? "))
print("The temperature is", temperature)
if temperature < 32:
    print("It is below freezing!")
elif temperature == 32:
    print("We're at the freezing point!")
elif temperature < 100:
    print("It is", temperature - 32, "degrees above freezing")
else:
    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 Ideas

- 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.
  - $C = \frac{F - 32}{1.8}$
  - $F = (1.8 \times C) + 32$
- Write a basic game, such as rock-paper-scissors.

More Learning Resources

- https://docs.python.org/3/
- https://docs.python.org/3/tutorial/index.html
- https://www.python.org/downloads/release/python-350/
- https://en.wikibooks.org/wiki/Python_Programming
- http://www.diveintopython3.net/
- http://www.codecademy.com/en/tracks/python
- https://wiki.python.org/moin/PythonBooks