IDEs and Basics

Lecture 1

CSCI 3351 & CSCI 6651
Dr. Frank Breitinger

Questions

- Who has programing experience?
- Who programmed python before?

Objectives

- Memorize the history and significance of Python.
- Set up the work environment and create a "hello world" application

Scripting vs. Programming

- Scripting languages are programming languages that don't require an explicit compilation step.
  - Lua, JS, VBScript, Perl
- Python is widely used without a compilation step, but the main implementation (CPython) does that by compiling to bytecode on-the-fly and then running the bytecode in a VM, and it can write that bytecode out to files (.pyc, .pyo) for use without recompiling.

Brief history of python

- Invented in the Netherlands, early 90s by Guido van Rossum
- Named after Monty Python
- Open sourced from the beginning, managed by *Python Software Foundation*
- Considered a scripting language, but is much more
- Scalable, object oriented and functional from the beginning
- Used by Google from the beginning

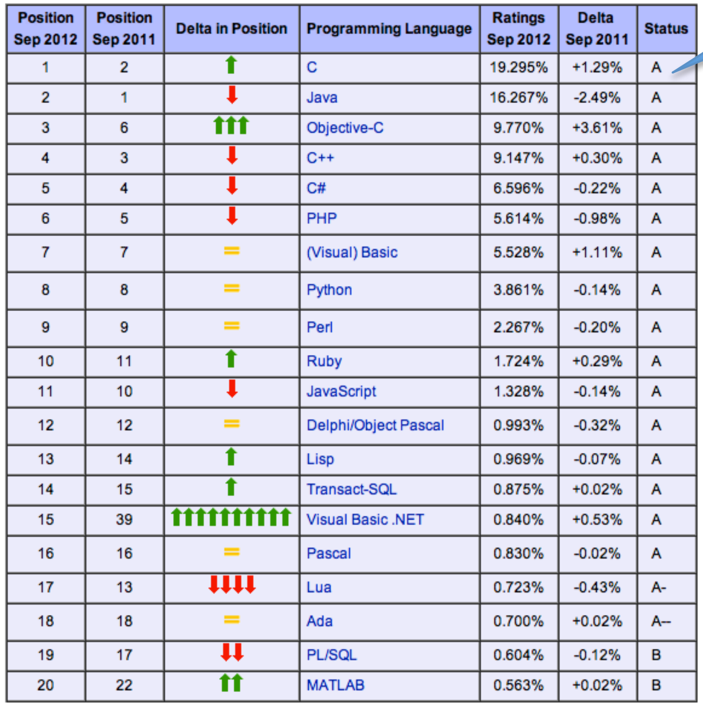
Python’s Benevolent Dictator For Life

“Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressive-ness is endangered.”

−Guido van Rossum

http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

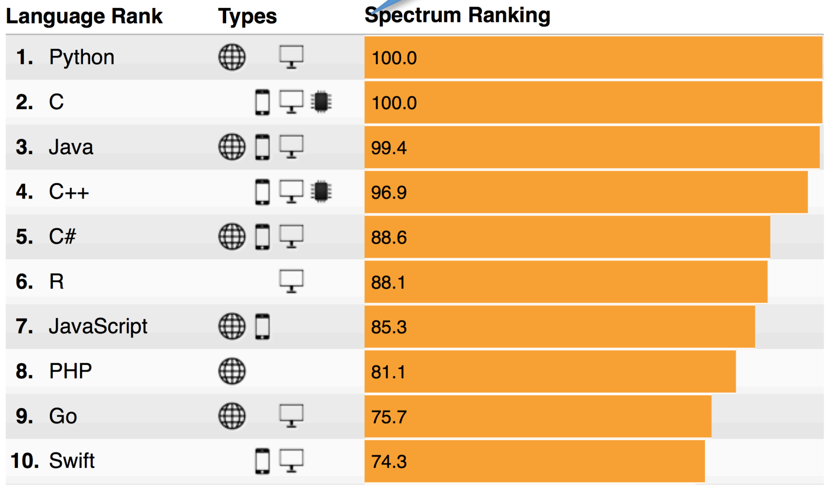
Python's place in the Market



Position Sep 2012	Position Sep 2011	Delta in Position	Programming Language	Ratings Sep 2012	Delta Sep 2011	Status
1	2	↑	C	19.295%	+1.29%	A
2	1	↓	Java	16.267%	-2.49%	A
3	6	↑↑↑	Objective-C	9.770%	+3.61%	A
4	3	↓	C++	9.147%	+0.30%	A
5	4	↓	C#	6.596%	-0.22%	A
6	5	↓	PHP	5.614%	-0.98%	A
7	7	=	(Visual) Basic	5.528%	+1.11%	A
8	8	=	Python	3.861%	-0.14%	A
9	9	=	Perl	2.267%	-0.20%	A
10	11	↑	Ruby	1.724%	+0.29%	A
11	10	↓	JavaScript	1.328%	-0.14%	A
12	12	=	Delphi/Object Pascal	0.993%	-0.32%	A
13	14	↑	Lisp	0.969%	-0.07%	A
14	15	↑	Transact-SQL	0.875%	+0.02%	A
15	39	↑↑↑↑↑↑↑↑↑↑	Visual Basic .NET	0.840%	+0.53%	A
16	16	=	Pascal	0.830%	-0.02%	A
17	13	↓↓↓↓	Lua	0.723%	-0.43%	A-
18	18	=	Ada	0.700%	+0.02%	A--
19	17	↓↓	PL/SQL	0.604%	-0.12%	B
20	22	↑↑	MATLAB	0.563%	+0.02%	B

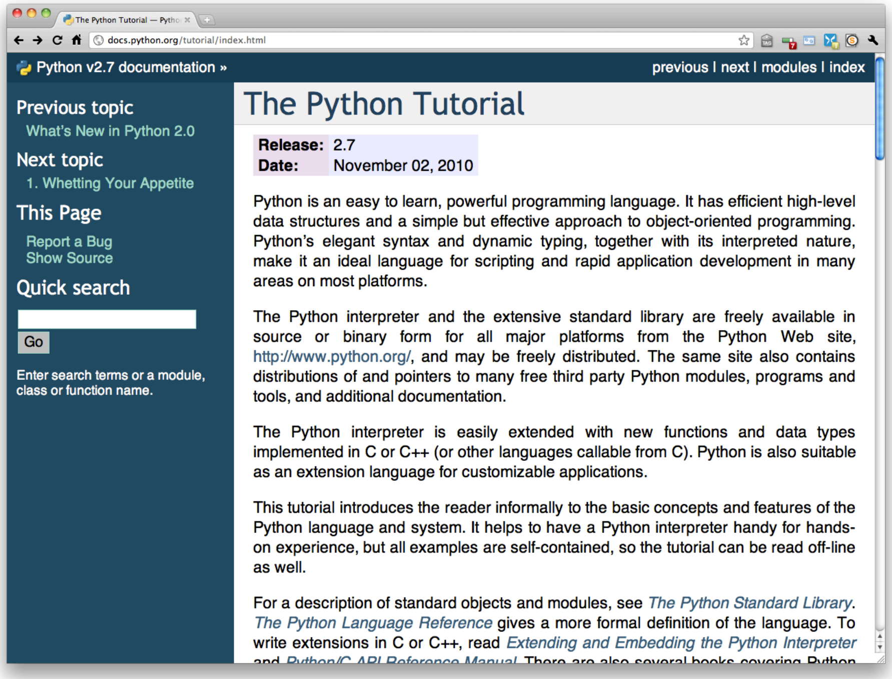
Data from 2012

Title: The 2017 Top Programming Languages



Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥	100.0
2. C	📱 🖥 ▪	100.0
3. Java	🌐 📱 🖥	99.4
4. C++	📱 🖥 ▪	96.9
5. C#	🌐 📱 🖥	88.6
6. R	🖥	88.1
7. JavaScript	🌐 📱	85.3
8. PHP	🌐	81.1
9. Go	🌐 🖥	75.7
10. Swift	📱 🖥	74.3

The Python tutorial is good!


The Python Tutorial — Pytho
docs.python.org/tutorial/index.html
Python v2.7 documentation »
previous | next | modules | index
Previous topic
What's New in Python 2.0
Next topic
1. Whetting Your Appetite
This Page
Report a Bug
Show Source
Quick search
Go
Enter search terms or a module, class or function name.

The Python Tutorial

Release:	2.7
Date:	November 02, 2010

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, http://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see *The Python Standard Library*. *The Python Language Reference* gives a more formal definition of the language. To write extensions in C or C++, read *Extending and Embedding the Python Interpreter* and *Python/C API Reference Manual*. There are also several books covering Python https://wiki.python.org/moin/Python2orPython3

Which Python?

Python 2.7

- Latest version is 2.7.3 released in mid-2012
- Last stable release before version 3
- Implements some of the new features in version 3, but fully backwards compatible

Python 3

- Released in 2008
- Many changes (including incompatible changes)
- Much cleaner language in many ways
- Strings use Unicode, not ASCII
- But: A few important third party libraries are not yet compatible with Python 3 right now

We will use 3.6.x!

RUNNING PYHON

Installing Python

- Python (CPython) is pre-installed on most Unix systems, including Linux and OS X
  – CPython is reference implementation of the Python programming language written in C.
- Two "latest versions" of CPython:
  – v2.7.3 released April 2012 and v3.6.x
  – Python 3 is a non-backward compatible version which we will use.
- Download from http://python.org/download/
- Python comes with a large library of standard modules

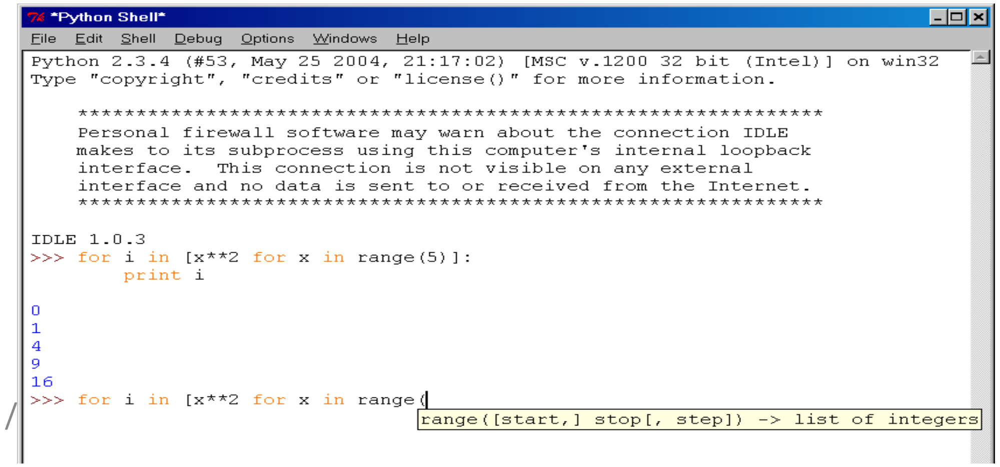
Deciding on an IDE

There are several options for an IDE:

- IDLE or PyCharm work for most OSs
- Emacs with python-mode or your favorite text editor
- Eclipse with Pydev (http://pydev.sourceforge.net/)
- I will use XCode and the Terminal

IDLE Development Environment

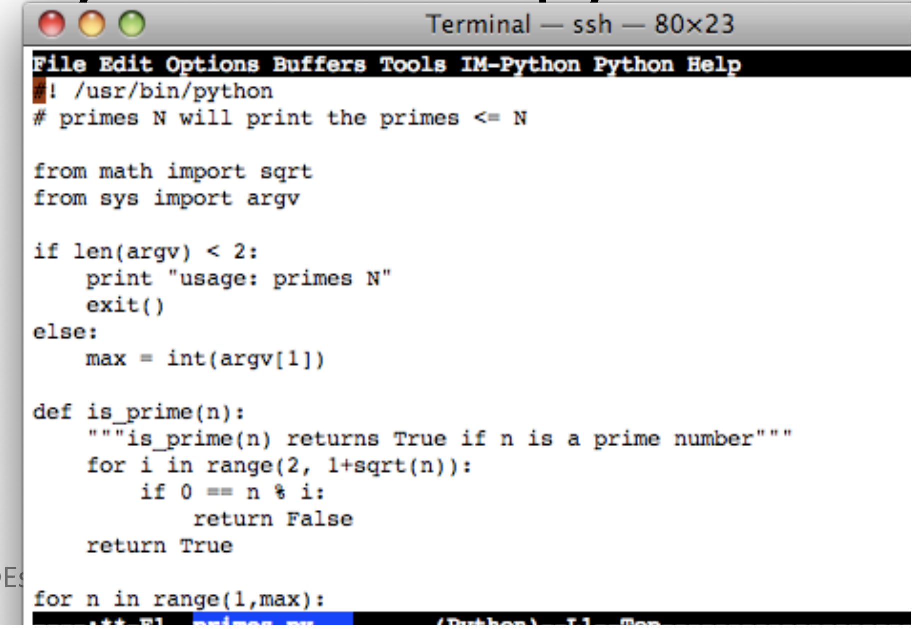
- IDLE is an Integrated DeveLopment Environment for Python, typically used on Windows
- Multi-window text editor with syntax highlighting, auto-completion, smart indent and other.
- Python shell with syntax highlighting.
- Integrated debugger with stepping, persis-tent breakpoints, and call stack visi-bility



```
Python 2.3.4 (#53, May 25 2004, 21:17:02) [MSC v.1200 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 1.0.3
>>> for i in [x**2 for x in range(5)]:
        print i

0
1
4
9
16
>>> for i in [x**2 for x in range(
                                  range([start,] stop[, step]) -> list of integers
```

Editing Python in Emacs

- Emacs python-mode has good support for editing Python, enabled by default for .py files
- Features: completion, symbol help, eldoc, and inferior interpreter shell, etc.



```
#! /usr/bin/python
# primes N will print the primes <= N

from math import sqrt
from sys import argv

if len(argv) < 2:
    print "usage: primes N"
    exit()
else:
    max = int(argv[1])

def is_prime(n):
    """is_prime(n) returns True if n is a prime number"""
    for i in range(2, 1+sqrt(n)):
        if 0 == n % i:
            return False
    return True

for n in range(1,max):
```

Running interactively on UNIX

- On Unix…

```
% python
>>> 3+3
6
```

- Python prompts with '>>>'.
- To exit Python (not Idle):
  - In Unix, type CONTROL-D
  - In Windows, type CONTROL-Z + <Enter>
  - Evaluate exit()

Running Programs on UNIX

- Call python program via the python interpreter

```
% python fact.py
```

- Make a python file directly executable by
  - Adding the appropriate path to your python interpreter as the first line of your file

    ```
    #!/usr/bin/python
    ```

  - Making the file executable

    ```
    % chmod a+x fact.py
    ```

  - Invoking file from Unix command line

    ```
    % fact.py
    ```

Example ‘script’: fact.py

```
#! /usr/bin/python
def fact(x):
    if x == 0:
        return 1
    return x * fact(x - 1)

print
print("N fact(N)")
print("---------")

for n in range(10):
    print(n, fact(n))
```

Since it is not compiled, order is important!

Python Scripts

- When you call a python program from the command line the interpreter evaluates each expression in the file
- Familiar mechanisms are used to provide command line arguments and/or redirect input and output
- Python also has mechanisms to allow a python program to act both as a script and as a module to be imported and used by another python program

Exercise – Setup

- Decide on an IDE and set up a machine that you can execute python.
  - I recommend to use your own device.
  - Make sure you use Python 3.
  - Verify that it works running the fact.py program from slide 16.

THE BASICS

Enough to Understand the Code

- **Indentation matters to code meaning**
  - Block structure indicated by indentation
- **First assignment to a variable creates it**
  - Variable types don't need to be declared.
  - Python figures out the variable types on its own.
- **Assignment is** `=` **and comparison is** `==`
- **For numbers** `+ - * / %` **are as expected**
  - Special use of `+` for string concatenation and `%` for string formatting (as in C's printf)
- **Logical operators are words (**`and, or, not`**) not symbols**
- **The basic printing command is** `print`

Whitespace

Whitespace is meaningful in Python: especially indentation and placement of newlines

- Use a newline to end a line of code
  - Use \ when must go to next line prematurely
- No braces {} to mark blocks of code, use consistent indentation instead
  - First line with less indentation is outside of the block
  - First line with more indentation starts a nested block
- Colons start of a new block in many constructs, e.g. function definitions, then clauses

A Code Sample (in IDLE)

```
x = 34 - 23            # A comment.
y = “Hello”            # Another one.
z = 3.45
if z == 3.45 or y == “Hello”:
    x = x + 1
    y = y + “ World”   # String concat.
print x
print y
```

Comments

- Start comments with #, rest of line is ignored
- Can include a “documentation string” as the first line of a new function or class you define
- Development environments, debugger, and other tools use it: it’s good style to include one

```
def fact(n):
    “““fact(n) assumes n is a positive integer and
    returns facorial of n.”””
    assert(n>0)
    return 1 if n==1 else n*fact(n-1)
```

Assignment

- Binding a variable in Python means setting a name to hold a *reference* to some *object*
  - Assignment creates references, not copies (example next slide!)
- Names in Python do not have an intrinsic type, objects have types
  - Python determines the type of the reference automatically based on what data is assigned to it
- You create a name the first time it appears on the left side of an assignment expression:
  x = 3
- A reference is deleted via garbage collection after any names bound to it have passed out of scope

Reference vs. Copy

```
a=['help', 'copyright', 'credits', 'license']
b=a
b.append('XYZ')
b['help', 'copyright', 'credits', 'license', 'XYZ']
a['help', 'copyright', 'credits', 'license', 'XYZ']
```

if you like to have a “copy” (a.k.a. deep copy) do:

```
b = a[:]
```

Assignment cont'd

- You can assign to multiple names at the same time

```
>>> x, y = 2, 3
>>> x
2
>>> y
3
```

- This makes it easy to swap values

```
>>> x, y = y, x
```

- Assignments can be chained

```
>>> a = b = x = 2
```

Naming Rules

- Names are case sensitive and cannot start with a number. They can contain letters, numbers, and underscores.
  - `bob   Bob   _bob   _2_bob_   bob_2   BoB`
- There are some reserved words:
  - `and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while`

Naming conventions

The Python community has these recommended naming conventions

- joined_lower for functions, methods and, attributes
- joined_lower or ALL_CAPS for constants
- StudlyCaps for classes
- camelCase only to conform to pre-existing conventions
- Attributes: interface, _internal, __private

Accessing non-existent variables

Accessing a name before it's been properly created (by placing it on the left side of an assignment), raises an error

```
>>> y

Traceback (most recent call last):
  File "<pyshell#16>", line 1, in -toplevel-
    y
NameError: name ‘y' is not defined
>>> y = 3
>>> y
3
```

Basic Data types - numbers

- Integers (default for numbers)
  - `z = 5 / 2   # Answer 2, integer division`
  - `z = 5 / 2.0   # Answer 2.5`
  - `z = 5 ** 2 # Answer 25`

Note, this was the behaviour of Python 2.7; Python 3.X behaves different (next slide)

- Floats
  - `x = 3.456`
- Python can handle binary, hex, octal or decimal.
  - 10 == 0xA == 0b1010 == 0o12
  - by default, print will print in decimal.

Same for Python 3.6

- Integers (default for numbers)
  - `z = 5 / 2  # Answer 2.5`
  - `z = 5 / 2.0  # Answer 2.5`
  - `z = 5 ** 2 # Answer 25`
  - `z = 5 // 2 # Answer 2`

Some special operators

- ** : exponentiation
- ^ : exclusive-or (bitwise)
- % : modulus
- // : divide with integral result (discard remainder)

Basic Data types - strings

- Can use “” or ‘’ to specify with “abc” == ‘abc’
- Unmatched can occur within the string: “matt’s”
- Use triple double-quotes for multi-line strings or strings than contain both ‘ and “ inside of them: “““a‘b“c”””
- The + can be used to concatenate strings, the * to print things multiple times, e.g., “hello “*20

Substrings

```
x = “Hello World“
x[2:] will output „llo World“
```

- Python calls this concept "slicing" and it works on more than just strings. Take a look here for a comprehensive introduction.

Slicing

```
a[start:end] # items start through end-1
a[start:]    # items start through the rest of the array
a[:end]      # items from the beginning through end-1
a[:]         # a copy of the whole array
```

Another feature is that start or end may be a negative number:

```
a[-1]    # last item in the array
a[-2:]   # last two items in the array
a[:-2]   # everything except the last two items
```

Changing a single character

```
>>> s = list("Hello zorld")
>>> s
['H', 'e', 'l', 'l', 'o', ' ', 'z', 'o', 'r',
'l', 'd']
>>> s[6] = 'W'
>>> s
['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r',
'l', 'd']
>>> "".join(s)
'Hello World'
```

http://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml/ifstatements.html

Conditions

```
x = 3
y = "hello"
if x == 2:
    print("x is 2")
elif x == 3 and y == "hello":
    print("x is 3 and hello")
else:
    print("x != 2 and x !=3")
```

Note: Python does not have case / switch. If / else are the only conditions.

Loops

```
for x in range(0, 3):
    print("We're on time %d" % x)

x = 0
while x < 3:
    print("We're on time %d" % x)
    x += 1
```

Exercise

- Write 3 programs where each prints certain substrings of a string.
  - E.g., “hello world” will result in
    - h
    - he
    - hel
    - hell
    - …
  - For loop, while loop, for loop where no “L” will be printed.

Remark!

Remember, assignments are individual work.
First time cheaters will get 0 on the assignment;
repeaters will fail the course.

Assignments

- Let us give repl.it a try:
- **Tue-Thu** Students use this link:
  - https://repl.it/classroom/invite/HdFvcwQ
- **Thursday-only** Students use this link:
  - https://repl.it/classroom/invite/Hj0wCyZ

Assignment 1

- Write a program that outputs *all* substrings of an input,
  - e.g., ABAC → A, B, A, C, AB, BA, AC, ABA, BAC, ABAC
- Bonus: remove duplicates
  - A, B, C, AB, BA, AC, ABA, BAC, ABAC
- Note, you should only use syntax / functions that were explained until this point.