

# Lecture 1: Basics: Introduction to Python

BT 3051 – Data Structures and Algorithms for Biology

Karthik Raman

Department of Biotechnology  
Bhupat and Jyoti Mehta School of Biosciences  
Indian Institute of Technology Madras

WHY PYTHON?

# Programming Languages

- ▶ Computers are great at following *\*explicit\** instructions
- ▶ Programming languages are our interface with the CPU
- ▶ Every programming language helps us in various ways
- ▶ C/C++/FORTRAN: Raw computational power (low-level languages)
- ▶ Java: Inter-operability and improved readability
- ▶ Python/MATLAB: High-level languages

# Programming Languages

- ▶ Computers are great at following \*explicit\* instructions
- ▶ Programming languages are our interface with the CPU
- ▶ Every programming language helps us in various ways
- ▶ C/C++/FORTRAN: Raw computational power (low-level languages)
- ▶ Java: Inter-operability and improved readability
- ▶ Python/MATLAB: High-level languages

# Programming Languages

- ▶ Computers are great at following \*explicit\* instructions
- ▶ Programming languages are our interface with the CPU
- ▶ Every programming language helps us in various ways
  - ▶ C/C++/FORTRAN: Raw computational power (low-level languages)
  - ▶ Java: Inter-operability and improved readability
  - ▶ Python/MATLAB: High-level languages

# Programming Languages

- ▶ Computers are great at following \*explicit\* instructions
- ▶ Programming languages are our interface with the CPU
- ▶ Every programming language helps us in various ways
- ▶ C/C++/FORTRAN: Raw computational power (low-level languages)
- ▶ Java: Inter-operability and improved readability
- ▶ Python/MATLAB: High-level languages

# Programming Languages

- ▶ Computers are great at following \*explicit\* instructions
- ▶ Programming languages are our interface with the CPU
- ▶ Every programming language helps us in various ways
- ▶ C/C++/FORTRAN: Raw computational power (low-level languages)
- ▶ Java: Inter-operability and improved readability
- ▶ Python/MATLAB: High-level languages

# Programming Languages

- ▶ Computers are great at following \*explicit\* instructions
- ▶ Programming languages are our interface with the CPU
- ▶ Every programming language helps us in various ways
- ▶ C/C++/FORTRAN: Raw computational power (low-level languages)
- ▶ Java: Inter-operability and improved readability
- ▶ Python/MATLAB: High-level languages



# Why Python?



- ▶ We need a method to tell the CPU what to do
- ▶ A computer cannot understand English/Hindi/Tamil (at least if you want to program it)
- ▶ A computer can also not understand Python (as is)!
- ▶ Any code written in any language needs to be translated to machine code for the computer to run it (compilation)
- ▶ “A computer can no more understand a Python program than it can understand a program written in plain English!”
- ▶ Python, however, is more human-friendly – powerful constructs render code more readable
- ▶ Python is an interpreted language (like MATLAB, but unlike C/C++/Java)

# Why Python?



- ▶ We need a method to tell the CPU what to do
- ▶ A computer cannot understand English/Hindi/Tamil (at least if you want to program it)
- ▶ A computer can also not understand Python (as is)!
- ▶ Any code written in any language needs to be translated to machine code for the computer to run it (compilation)
- ▶ “A computer can no more understand a Python program than it can understand a program written in plain English!”
- ▶ Python, however, is more human-friendly – powerful constructs render code more readable
- ▶ Python is an interpreted language (like MATLAB, but unlike C/C++/Java)

# Why Python?



- ▶ We need a method to tell the CPU what to do
- ▶ A computer cannot understand English/Hindi/Tamil (at least if you want to program it)
- ▶ A computer can also not understand Python (as is)!
- ▶ Any code written in any language needs to be translated to machine code for the computer to run it (compilation)
- ▶ “A computer can no more understand a Python program than it can understand a program written in plain English!”
- ▶ Python, however, is more human-friendly – powerful constructs render code more readable
- ▶ Python is an interpreted language (like MATLAB, but unlike C/C++/Java)

# Why Python?



- ▶ We need a method to tell the CPU what to do
- ▶ A computer cannot understand English/Hindi/Tamil (at least if you want to program it)
- ▶ A computer can also not understand Python (as is)!
- ▶ Any code written in any language needs to be translated to machine code for the computer to run it (compilation)
- ▶ “A computer can no more understand a Python program than it can understand a program written in plain English!”
- ▶ Python, however, is more human-friendly – powerful constructs render code more readable
- ▶ Python is an interpreted language (like MATLAB, but unlike C/C++/Java)

# Why Python?



- ▶ We need a method to tell the CPU what to do
- ▶ A computer cannot understand English/Hindi/Tamil (at least if you want to program it)
- ▶ A computer can also not understand Python (as is)!
- ▶ Any code written in any language needs to be translated to machine code for the computer to run it (compilation)
- ▶ “A computer can no more understand a Python program than it can understand a program written in plain English!”
- ▶ Python, however, is more human-friendly – powerful constructs render code more readable
- ▶ Python is an interpreted language (like MATLAB, but unlike C/C++/Java)

# Why Python?



- ▶ We need a method to tell the CPU what to do
- ▶ A computer cannot understand English/Hindi/Tamil (at least if you want to program it)
- ▶ A computer can also not understand Python (as is)!
- ▶ Any code written in any language needs to be translated to machine code for the computer to run it (compilation)
- ▶ “A computer can no more understand a Python program than it can understand a program written in plain English!”
- ▶ Python, however, is more human-friendly – powerful constructs render code more readable
- ▶ Python is an interpreted language (like MATLAB, but unlike C/C++/Java)

# Why Python?



- ▶ We need a method to tell the CPU what to do
- ▶ A computer cannot understand English/Hindi/Tamil (at least if you want to program it)
- ▶ A computer can also not understand Python (as is)!
- ▶ Any code written in any language needs to be translated to machine code for the computer to run it (compilation)
- ▶ “A computer can no more understand a Python program than it can understand a program written in plain English!”
- ▶ Python, however, is more human-friendly – powerful constructs render code more readable
- ▶ Python is an interpreted language (like MATLAB, but unlike C/C++/Java)

# Why Python?

[http://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Python_(programming_language))

- ▶ Python is a widely used general-purpose, high-level programming language
- ▶ Its design philosophy emphasises **code readability**, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C
- ▶ The language provides constructs intended to enable clear programs on both a small and large scale
- ▶ Python is heavily used in the CS/IT/scientific computing
- ▶ Python has a very large number of useful packages for science, math, biology, <you name it>
- ▶ Can also interface with C/C++/Java libraries etc. for raw speed!



# Why Python?

[http://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Python_(programming_language))

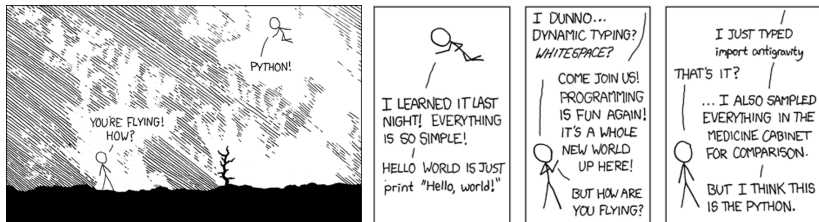
- ▶ Python is a widely used general-purpose, high-level programming language
- ▶ Its design philosophy emphasises **code readability**, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C
- ▶ The language provides constructs intended to enable clear programs on both a small and large scale
- ▶ Python is heavily used in the CS/IT/scientific computing
- ▶ Python has a very large number of useful packages for science, math, biology, <you name it>
- ▶ Can also interface with C/C++/Java libraries etc. for raw speed!

# Why Python?

[http://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Python_(programming_language))

- ▶ Python is a widely used general-purpose, high-level programming language
  - ▶ Its design philosophy emphasises **code readability**, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C
  - ▶ The language provides constructs intended to enable clear programs on both a small and large scale
- ▶ Python is heavily used in the CS/IT/scientific computing
  - ▶ Python has a very large number of useful packages for science, math, biology, <you name it>
  - ▶ Can also interface with C/C++/Java libraries etc. for raw speed!

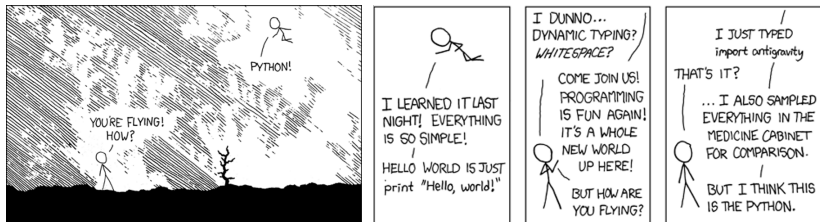
# Why Python?



*"I wrote 20 short programs in Python yesterday. It was wonderful. Perl, I'm leaving you."*

Courtesy: <http://xkcd.com/353/>

# Why Python?



*"I wrote 20 short programs in Python yesterday. It was wonderful. Perl, I'm leaving you."*

Courtesy: <http://xkcd.com/353/>

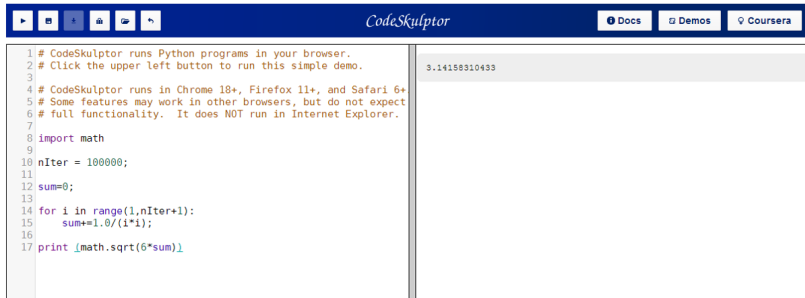
See also: [The Homogenization Of Scientific Computing Or Why Python Is Steadily Eating Other Languages' Lunch](#)

# IP[y]: IPython

Interactive Computing

- ▶ <http://ipython.org/>
- ▶ Fantastic and powerful interaction shell for Python

► Thanks, Scott Rixner (<http://www.cs.rice.edu/~rixner/>)!



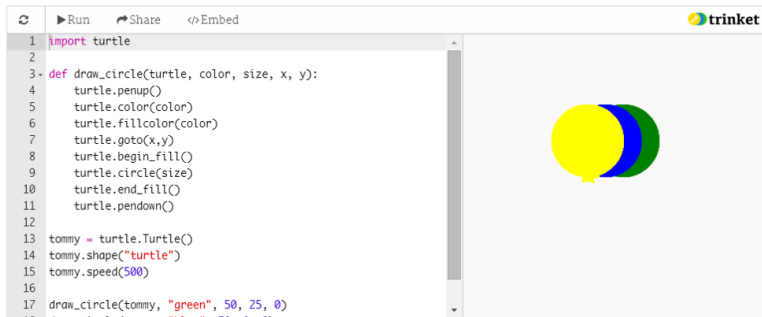
The screenshot shows the CodeSkulptor web interface. At the top, there is a dark blue header with the text "CodeSkulptor" in the center. To the left of the header are several small icons: a play button, a document, a plus sign, a lock, a speech bubble, and a refresh button. To the right of the header are three buttons: "Docs", "Demos", and "Coursera". Below the header, the main area is divided into two panels. The left panel contains a Python program with line numbers 1 through 17. The right panel shows the output of the program, which is the number 3.14158310433.

```
1 # CodeSkulptor runs Python programs in your browser.
2 # Click the upper left button to run this simple demo.
3
4 # CodeSkulptor runs in Chrome 18+, Firefox 11+, and Safari 6+
5 # Some features may work in other browsers, but do not expect
6 # full functionality. It does NOT run in Internet Explorer.
7
8 import math
9
10 nIter = 100000;
11
12 sum=0;
13
14 for i in range(1,nIter+1):
15     sum+=1.0/(i*i);
16
17 print (math.sqrt(6*sum))
```

3.14158310433

## Teach Python Straight from Your Class Website

Skip the installs and get back to teaching with **trinket's** interactive Python:



```
1 import turtle
2
3 def draw_circle(turtle, color, size, x, y):
4     turtle.penup()
5     turtle.color(color)
6     turtle.fillcolor(color)
7     turtle.goto(x,y)
8     turtle.begin_fill()
9     turtle.circle(size)
10    turtle.end_fill()
11    turtle.pendown()
12
13 tommy = turtle.Turtle()
14 tommy.shape("turtle")
15 tommy.speed(500)
16
17 draw_circle(tommy, "green", 50, 25, 0)
```

# Coding Style Guidelines

Courtesy: Scott Rixner, Coursera Principles of Computing

- ▶ Use *docstrings*: check out `__doc__` in Python
- ▶ Comment appropriately
- ▶ Do not use global variables
- ▶ Indent your code, or obviously, Python will throw a fit!
- ▶ Use [4] spaces; **NEVER USE TABS!**
- ▶ Name variables/functions/classes descriptively
- ▶ Class fields should be private; can only be accessed by a method, if at all
- ▶ Design your functions to do a single thing but do it well
- ▶ Use `pylint` to clean your code to conform to accepted style guidelines, e.g. PEP8<sup>a</sup>!

---

<sup>a</sup><https://www.python.org/dev/peps/pep-0008/>



# Coding Style Guidelines

Courtesy: Scott Rixner, Coursera Principles of Computing

- ▶ Use *docstrings*: check out `__doc__` in Python
- ▶ Comment appropriately
- ▶ Do not use global variables
- ▶ Indent your code, or obviously, Python will throw a fit!
- ▶ Use [4] spaces; **NEVER USE TABS!**
- ▶ Name variables/functions/classes descriptively
- ▶ Class fields should be private; can only be accessed by a method, if at all
- ▶ Design your functions to do a single thing but do it well
- ▶ Use `pylint` to clean your code to conform to accepted style guidelines, e.g. PEP8<sup>a</sup>!

---

<sup>a</sup><https://www.python.org/dev/peps/pep-0008/>

# Coding Style Guidelines

Courtesy: Scott Rixner, Coursera Principles of Computing

- ▶ Use *docstrings*: check out `__doc__` in Python
- ▶ Comment appropriately
- ▶ Do not use global variables
- ▶ Indent your code, or obviously, Python will throw a fit!
- ▶ Use [4] spaces; **NEVER USE TABS!**
- ▶ Name variables/functions/classes descriptively
- ▶ Class fields should be private; can only be accessed by a method, if at all
- ▶ Design your functions to do a single thing but do it well
- ▶ Use `pylint` to clean your code to conform to accepted style guidelines, e.g. PEP8<sup>a</sup>!

---

<sup>a</sup><https://www.python.org/dev/peps/pep-0008/>

# Coding Style Guidelines

Courtesy: Scott Rixner, Coursera Principles of Computing

- ▶ Use *docstrings*: check out `__doc__` in Python
- ▶ Comment appropriately
- ▶ Do not use global variables
- ▶ Indent your code, or obviously, Python will throw a fit!
- ▶ Use [4] spaces; **NEVER USE TABS!**
- ▶ Name variables/functions/classes descriptively
- ▶ Class fields should be private; can only be accessed by a method, if at all
- ▶ Design your functions to do a single thing but do it well
- ▶ Use `pylint` to clean your code to conform to accepted style guidelines, e.g. PEP8<sup>a</sup>!

---

<sup>a</sup><https://www.python.org/dev/peps/pep-0008/>

# Coding Style Guidelines

Courtesy: Scott Rixner, Coursera Principles of Computing

- ▶ Use *docstrings*: check out `__doc__` in Python
- ▶ Comment appropriately
- ▶ Do not use global variables
- ▶ Indent your code, or obviously, Python will throw a fit!
- ▶ Use [4] spaces; **NEVER USE TABS!**
- ▶ Name variables/functions/classes descriptively
- ▶ Class fields should be private; can only be accessed by a method, if at all
- ▶ Design your functions to do a single thing but do it well
- ▶ Use `pylint` to clean your code to conform to accepted style guidelines, e.g. PEP8<sup>a</sup>!

---

<sup>a</sup><https://www.python.org/dev/peps/pep-0008/>

# Coding Style Guidelines

Courtesy: Scott Rixner, Coursera Principles of Computing

- ▶ Use *docstrings*: check out `__doc__` in Python
- ▶ Comment appropriately
- ▶ Do not use global variables
- ▶ Indent your code, or obviously, Python will throw a fit!
- ▶ Use [4] spaces; **NEVER USE TABS!**
- ▶ Name variables/functions/classes descriptively
- ▶ Class fields should be private; can only be accessed by a method, if at all
- ▶ Design your functions to do a single thing but do it well
- ▶ Use `pylint` to clean your code to conform to accepted style guidelines, e.g. PEP8<sup>a</sup>!

---

<sup>a</sup><https://www.python.org/dev/peps/pep-0008/>

# Coding Style Guidelines

Courtesy: Scott Rixner, Coursera Principles of Computing

- ▶ Use *docstrings*: check out `__doc__` in Python
- ▶ Comment appropriately
- ▶ Do not use global variables
- ▶ Indent your code, or obviously, Python will throw a fit!
- ▶ Use [4] spaces; **NEVER USE TABS!**
- ▶ Name variables/functions/classes descriptively
- ▶ Class fields should be private; can only be accessed by a method, if at all
- ▶ Design your functions to do a single thing but do it well
- ▶ Use `pylint` to clean your code to conform to accepted style guidelines, e.g. PEP8<sup>a</sup>!

---

<sup>a</sup><https://www.python.org/dev/peps/pep-0008/>

# Coding Style Guidelines

Courtesy: Scott Rixner, Coursera Principles of Computing

- ▶ Use *docstrings*: check out `__doc__` in Python
- ▶ Comment appropriately
- ▶ Do not use global variables
- ▶ Indent your code, or obviously, Python will throw a fit!
- ▶ Use [4] spaces; **NEVER USE TABS!**
- ▶ Name variables/functions/classes descriptively
- ▶ Class fields should be private; can only be accessed by a method, if at all
- ▶ Design your functions to do a single thing but do it well
- ▶ Use `pylint` to clean your code to conform to accepted style guidelines, e.g. PEP8<sup>a</sup>!

---

<sup>a</sup><https://www.python.org/dev/peps/pep-0008/>

# Coding Style Guidelines

Courtesy: Scott Rixner, Coursera Principles of Computing

- ▶ Use *docstrings*: check out `__doc__` in Python
- ▶ Comment appropriately
- ▶ Do not use global variables
- ▶ Indent your code, or obviously, Python will throw a fit!
- ▶ Use [4] spaces; **NEVER USE TABS!**
- ▶ Name variables/functions/classes descriptively
- ▶ Class fields should be private; can only be accessed by a method, if at all
- ▶ Design your functions to do a single thing but do it well
- ▶ Use `pylint` to clean your code to conform to accepted style guidelines, e.g. PEP8<sup>a</sup>!

---

<sup>a</sup><https://www.python.org/dev/peps/pep-0008/>



# Questions?

Also remember

STANFORD UNIVERSITY  
ILLINOIS  
PRINCETON UNIVERSITY  
Berkeley  
MIT  
piazza

