Introduction to Python

Outline

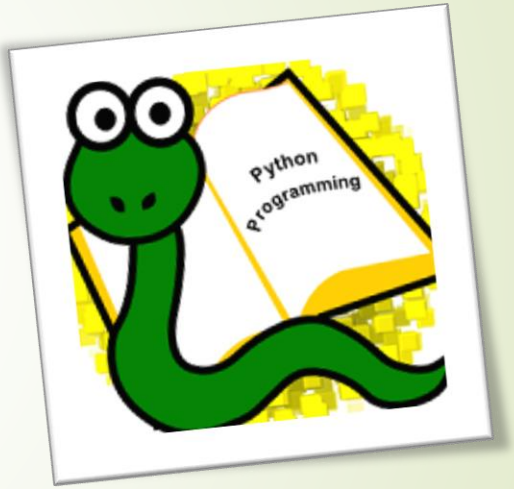
- Introduction
- Getting Started
- Python Basics
  - Standard Data Types
  - Operators
- Control Structures

Introduction

- Python is a widely-used general-purpose high-level programming language
- Invented by Guido van Rossum in 1991 at CWI in the Netherlands
- It combines the power of systems languages, such as C and Java, with the ease and rapid development of scripting languages, such as Ruby

Language Features

- Simple and Minimalistic
- Easy to Learn
- High-level Language
- Portable
- Extensible
- Embeddable
- Extensive Libraries
- Free, Open Source, … and Fun!



Programming Paradigms

- Object-Oriented Programming
- Structured Programming
- Functional Programming
- Aspect-Oriented Programming
- Logic Programming (by extension)
- Design by Contract (by extension)

Language Philosophy

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Flat is better than nested
- Sparse is better than dense
- Readability counts

Language Philosophy (Cont'd)

- Special cases aren't special enough to break the rules
- Errors should never pass silently, unless explicitly silenced
- There should be one– and preferably only one – obvious way to do it
- If the implementation is hard to explain, it's a bad idea
- Namespaces are one honking great idea – let's do more of those!

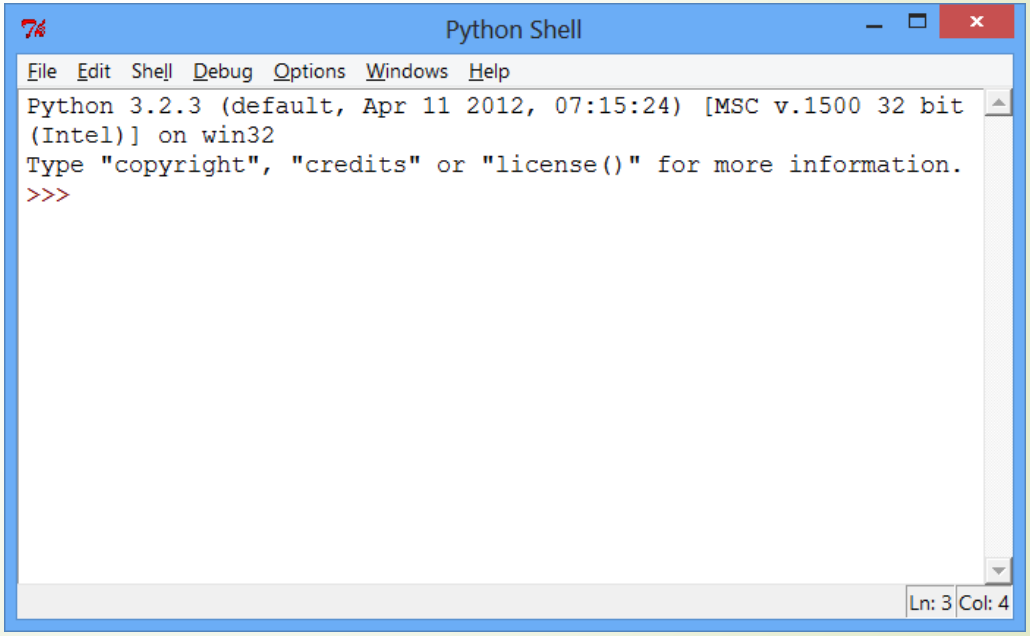
Python 2 and 3

- Python 2.0 was released in 2000, with many new features added
- Python 3.0, adjusting several aspects of the core language, was released in 2008
- Python 3.0 is backwards-incompatible
  - Codes written for Python 2.x may not work under 3.x
- Python 2.x is legacy, Python 3.x is the present and future of the language
- We use Python 3.x in this course

Getting Started

Getting Started

- Download Python from http://python.org
- Install it
- Run it



Python Shell

- Python's interactive interpreter is one of the most powerful tools used in everyday Python development
- It enables you to test a few lines of code without needing to create, edit, save, and run a source file
- Not only it verifies your code's correctness, but it also enables inspecting data structures or altering key values, prior to adding it to your source files

Using Shell

- Python shell evaluates the expression entered after prompt and displays the result

```
>>> 2 + 5
7
>>> 2 ** 100
1267650600228229401496703205376
```

- We can explicitly use print function to do this

```
>>> print(3 * 10)
30
```

First Program

- A Python program (or script) is a sequence of Python statements that goes into a text file, having a .py extension
- This is our first Python program:

```
print('Hello World!')
```

- People usually judge the quality of a programming language by the simplicity of the Hello World! program
- By this standard, Python does about as well as possible

Comments

- Comments in Python are denoted with the hash mark (#)

```
# hello.py
# By: Hamid Zarrabi-Zadeh

print('Hello World!')  # short comment
```

- Comments are also used to prevent working code from executing
- Typical usage in configuration files to disable/enable options

Python Basics

Variables

- Python is a dynamically typed language
- Variables can be thought of as names that refer to otherwise anonymous objects, which contain the actual values involved
- Any given variable can have its value altered at any time

```
>>> x = 2
>>> x
2
>>> x = 'Ali'
>>> x
'Ali'
```

Basic Data Types

- Boolean
- Strings
- Numbers
- None

```
>>> type(True)
<class 'bool'>
>>> type('Hello')
<class 'str '>
>>> type(5)
<class 'int '>
>>> type(5.2)
<class 'float '>
```

Operators

- Basic Operators
  - Arithmetic (+, - , *, /, %, //, **)
  - Assignment (=, +=, -=, *= /=, %=, **= , //=)
  - Comparison (<, >, <=, >=, ==, !=)
  - Logical (and, or, not)
- Notes:
  - + on strings does string concatenation
  - * on (string, int) repeats string

Type Conversion

- Python has strong typing (unlike JavaScript)

```
>>> 'Ali' + 10
Traceback (most recent call last):
  File "<pyshell #0>", line 1, in <module >
    'Ali' + 10
TypeError: Can't convert 'int' to str implicitly
```

- We need to use type converter functions

```
>>> '123' + str(45)
'12345'
>>> int('123') + 45
168
```

Input

- We can use input function to get user input
- The return value is always a string

```
radius = input('Enter radius: ')
r = float(radius)
area = 3.14159 * r ** 2
print('The area is:', area)
```

Control Structures

Control Structures

- Conditionals
  - if, else, elif
- Loops
  - for
  - while

Conditionals

- Like other languages, Python features if and else statements
- Python's "else-if" is spelled elif

```
ans = input("Enter 'y' or 'n': ")
if ans == 'y':
    print "Entered 'y'"
elif ans == 'n':
    print "Entered 'n'"
else:
    print 'Invalid key pressed!'
```

Truth Value Testing

- Any object can be tested for truth value, for use in a condition
- The following values are considered False
  - None
  - False
  - zero of any numeric type, e.g., 0, 0.0, 0j.
  - any empty sequence or dictionary, e.g., '', (), [], {}
- All other values are considered True

While Loop

- The while loop continues to execute the same body of code until the conditional statement is no longer True

```
i = 0
while i < 10:
    print(i)
    i += 1
```

- We can use break and continue inside the loops

For Loop

- The for loop in Python is much simpler that other C-like languages

```
for x in ['Ali', 'Mahsa', 'Navid', 'Zahra']:
   print('Hello ' + x + '!')
```

- We can use range() function to produce a list of numbers

```
for i in range(10):
   print(i)
```

References

- Python Official Website
  - http://python.org/
- Python 3 Documentation
  - http://docs.python.org/3/
- Python Web Development with Django
  - By Jeff Forcier, Paul Bissex, Wesley Chun