A Brief Review of Python

COMPSCI 270
(Thanks for Richard Guo's slides)

Zhenyu Zhou

CS Dept., Duke University

zzy@cs.duke.edu

@Duke University
January 21, 2015

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Installing

- Mac OS (Recommended)
  - Already done after OS X 10.8
  - Course website
- Linux (Sometimes the same as OS X, but may not be tested by myself)
  - apt-get
- Windows (Not recommended)
  - Official website
    - https://www.python.org/downloads/windows/
- Version
  - 2.7
  - python --version

Installing

- Try to run pacman.py
  - python pacman.py (in terminal)

- Text editor
  - Vim
  - Sublime
  - Notepad++
  - Etc.

Installing

- Eclipse (Recommended IDE)
  - Plugin
    - PyDev
  - Install plugins (not only for PyDev)
    - Install new software/eclipse marketplace (Mac)
  - Tell Eclipse where is your python
    - Preference -> PyDev -> Interpreters -> Python Interpreter
    - The location of your interpreter (Mac & Linux): whereis python
  - Create new python project
    - File -> New -> Project -> PyDev -> Pydev Project

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Executing

- Terminal
  - Shell prompt - python abc.py
    - File name ends with .py
  - Python prompt
    - After typing “python” under shell prompt
    - Type the script directly

Executing

- Shell prompt

```
search — bash — 80×24
Zhenyus-MacBook-Pro:search zzy$ python pacman.py
Pacman died! Score: -533
Ending graphics raised an exception: 0
Average Score: -533.0
Scores:        -533
Win Rate:      0/1 (0.00)
Record:        Loss
```

- Python prompt

```
search — Python — 80×24
Zhenyus-MacBook-Pro:search zzy$ python
Python 2.7.5 (default, Mar  9 2014, 22:15:05)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.0.68)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print 'hello world'
hello world
>>> python pacman.py
  File "<stdin>", line 1
    python pacman.py
                   ^
SyntaxError: invalid syntax
>>>
```

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

The Python Style

- Interpreter
  - Interactive running
  - Running a script

```
while True:
    s = input('Enter something : ')
    if s == 'quit':
        break
    print('Length of the string is', len(s))
print('Done')
```

- Dynamic
  - Use variables without declaration

- Structured by **Indent** and colon
  - Options: TAB, 2 spaces, 4 spaces
  - Be consistent throughout the program!
  - Q: which one is used in pacman.py?

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Basic Operators

- Most are straightforward
  - + - / * % []
- Division: integer vs. fractional
  - 10/3=3
  - 10.0/3=3.3333..
  - Trick: a * 1.0 / b
  - Can be different in other python version
- Power: ** (not ^)

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Variable Types

- Integer
  - 100
- Double
  - 3.14
- Bool
  - True
- None
  - Just like null in Java

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Built-in Data Structures

- List
  - An array storing different kinds of elements
    - >>> a = [] *(initialize)*
    - >>> a = [“hello”, 1, 5.5]
    - >>> listOfList = [[1,2],[3,4]]
  - Indexed from 0
  - Operations
    - a.append(x)
    - removal
      - del a[0]
      - if “hello” **in** a: a.remove(“hello”) *(first occurrence of the value)*
    - a.sort() *(do not forget the brackets, it is a method)*
    - a.reverse()
    - a + listOfList *(concatenation)*

Built-in Data Structures

- Tuple
  - x = (1,2,'ok')
  - Much like a list, but cannot be changed
  - One-element tuple: (1,)

```
>>> (1)
1
>>> (1,)
(1,)
>>>
```

- String
  - s = 'hello '+'world'
  - Newline: \n
  - Quote: \'
  - Q: how to print out '\' itself?

Built-in Data Structures

- Dictionary
  - A hash table: key -> value
    - mydict = {} *(initialize)*
    - mydict[1] = "one" *(automatically adding a new key-value pair)*
  - Keys do NOT have the same order as you put in them
  - Key must be of an immutable type:
    - string, number
    - tuple
      - mydict[(-1,0)] = "west"
  - Operations
    - mydict.keys() *(in the form of a list)*
    - mydict.values() *(in the form of a list)*
    - print mydict[(-1,0)]
    - del mydict[(-1,0)]
  - How to get the keys sorted by value?
    - `for w in sorted(mydict, key=mydict.get):`
    - `  print w, mydict[w]`

Built-in Data Structures

- Set
  - An unordered collection of unique elements
  - Efficient to test if an element is marked/visited
    - x **in** exploredSet
  - Initialization
    - setOfShapes = set() *(empty set)*
    - setOfShapes = set(["circle", "triangle", "square", "circle"])
    - setOfShapes = {"circle", "triangle", "square", "circle"}
  - Operations
    - setOfShapes.add("hexagon")
    - setOfShapes.remove("circle")
    - set1 | set2
    - set1 & set2
    - set1 – set2

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Control Flows

- If statement
  - age = 20
  - if age >= 6: # Don’t forget “:”
  - print 'teenager'
  - elif age >= 18:
  - print 'adult'
  - else:
  - print 'kid'

- Q: What’s the output?

Control Flows

- For statement
  - sum = 0
  - for x in range(101): # what is range()?
  - sum = sum + x
  - print sum

  - names = ['Michael', 'Bob', 'Tracy']
  - for name in names:
  - print name

- Q: how to translate *for(int i = 50; i < 100; i += 2)* into python?

Control Flows

- While statement
  - i = 0
  - while i<100:
  - i = i + 1

- Special clauses in loops
  - break
  - continue
  - else
    - Executed if no “break” is executed in the loop

```
for answer in possibleAnswers:
    if isRightAnswer(answer):
        break
else:
    print “No answer found”
```

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Function

- Defining a function
  - `def myadd(x, y):`
  - `    z = x+y`
  - `    return z` *(would return None if without this line)*

- Calling functions
  - `myNumbers = [2, 4]`
  - `print myadd(myNumbers[0], myNumbers[1])`
  - `print myadd(*myNumbers)`

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Class

- Defining a class
  - `def classname(baseClassName)`

- Providing data attributes and methods with “self”
  - `self.title = “a simple class”`
  - `def showTitle(self, repeats=1):`
  - `  for t in range(0, repeats):`
  - `  print self.title`

- Construction
  - `def __init__(self, someArg):`

- Making a variable looking private by naming with a leading underscore
  - Unlike C++, Python does not enforce data hiding mechanism

Class – Queue

```
class Queue:
    def __init__(self):
        self.queueList = []

    def push(self, x):
        self.queueList.append(x)

    def pop(self):
        z = self.queueList[0]
        del self.queueList[0]
        return z

    def isEmpty(self):
        return (len(self.queueList)==0)
```

Class - TreeNode

- Try to implement a TreeNode class on your own
  - Parent
  - Path from root
  - “May” be helpful to your assignments

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Module

- Similar to library and package in Java
- A .py file is called a module
- Import a module
  - `import util`
  - `myQueue = util.Queue()`
- Please learn more about util.py

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Useful Tricks

- Try to print everything out when debugging
  - More advanced
    - assert
    - logging
    - pdb

- Looping with ease
  - Iterate with multiple variables
    - `for i, v in enumerate(["a","b","c"])`
    - `  print i, v`

Useful Tricks

- About list
  - `>>> classmates = ['Michael', 'Bob', 'Tracy']`
  - `>>> classmates[-1]`
  - `'Tracy'`
  - `>>> classmates[0:2]`
  - `['Michael', 'Bob']`

  - `>>> S = [x**2 for x in range(4)]`
  - `>>> S`
  - `[0, 1, 4, 9]`

Useful Tricks

- Copy before modifying
  - What would you expect?
    - x = [1,2,3,4]
    - y = x
    - y.append(5)
    - print x
  - Even more careful when passing them to a function
    - x = [1,2,3,4]
    - def sumUp(z):
    - z.append(sum(z))
    - return z[-1]
    - sumUp(x)
    - print x
  - Solution
    - X = copy.copy(Y)
    - X = copy.deepcopy(Y)
    - X = Y.copy()

Outline

- Installing
- Executing
- The Python Style
- Basic Operators
- Variable Types
- Built-in Data Structures
- Control Flows
- Function
- Class
- Module
- Useful Tricks
- Reference

Reference

- Thanks for Richard Guo's python review slides. Some portions of this slide will be based on his slides.
- Official tutorial
  - http://docs.python.org/2/tutorial/
- Python/UNIX tutorial on the course webpage
  - http://inst.eecs.berkeley.edu/~cs188/fa10/projects/tutorial/tutorial.html#Python
- A Byte of Python
  - http://swaroopch.com/notes/python/
- Python Information and Examples
  - http://www.secnetix.de/olli/Python/
- Learning Python (O'Reilly)
- Python Pocket Reference (O'Reilly)

Thank you!

Questions?