컴퓨터 프로그래밍 연습 (4190-103A, 2017 봄: 컴퓨터공학부 학생 제외)

* 강사: 김형주교수, (880-1826, 010-5213-1992), hjk@snu.ac.kr, (연구실: 301동 406호)
* 강의: 301동-203, 화목 3:30 – 5:20
* Office Hours: 화목 (오후 1시: Email appointment is needed)
* 조교(TA): Internet Database Lab (880-1830) 석박사통합과정생

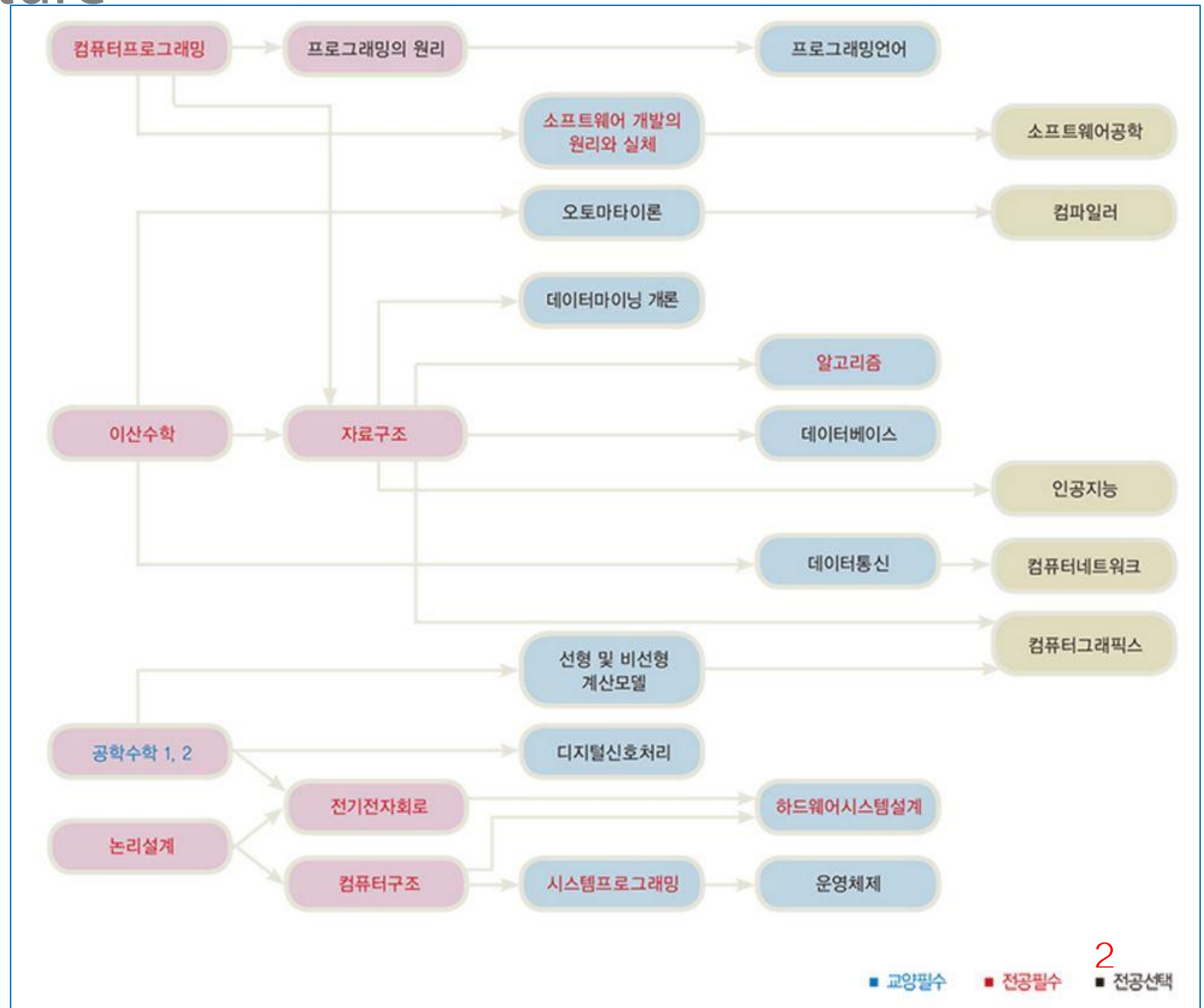
  이동준 djlee@idb.snu.ac.kr 김동효 dhkim@idb.snu.ac.kr 이명연 myyi@idb.snu.ac.kr

* Class Materials: Internet Database Lab Website: http://idb.snu.ac.kr
* 가능하면 Notebook PC를 가져오는 것이 좋음
* 평가: 5 Programming (6% each), 2 midterms (20% each) and 1 final exam (30%)
* 카카오톡 방을 만들어서 운영할 예정!

컴퓨터공학 과정 Curriculum Structure

컴퓨터공학의 개론및 실습

프로그래밍 연습



What is Programming?

- The Real World Problem: P
- Transform P into AP (Abstract Problem) through Abstraction
- Represent the AP using the given Programming Language
  - Using Basic Data Types, Advanced Data Types, User-defined Data Types
- Solve the AP with Algorithm based on Computational Thinking
  - Defining functions

Python Data Types과 연산

- Basic Data Types
  - Integer
  - Floating Number
  - Boolean
  - Character

  우리가 익숙한 mathematical notation으로 연산

  Ex: 3 + 4
- Advanced Data Types
  - Tuple
  - String
  - List
  - Dictionary
  - Set

  특정 data type에 정의된 function들을 call해서 연산

  Ex: myString = "S N U"
  myString.split()
- User-Defined Data Types
  - Student
  - Automobile
  - ……

  특정 data type에 정의된 function들을 call해서 연산

  Ex: myAuto = Automobile("GM", "2016", "5Door")
  myAuto.print()
- Library
  - Math
  - Random
  - ….

  특정 library에 정의된 function들을 call해서 연산

  Ex: import math
  math.sqrt(4)

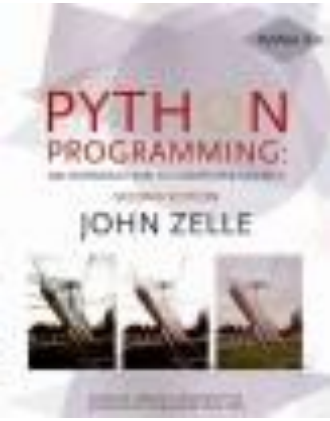
Table of Contents

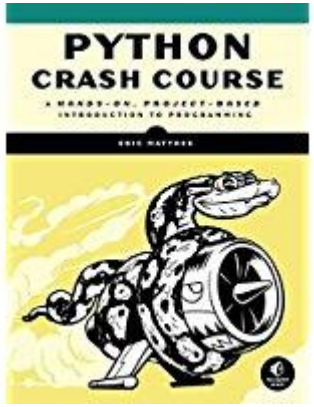
(1) Python Introduction 24pp.pptx
(2) Python Tutorial 79pp.pptx
(3) Python Basic Data Types 33pp.pptx
(4) Python Functions 39pp.pptx
(4-A) Function Practice 10pp.pptx
(4-B) Looping Function Practice 6pp.pptx
(4-C) Advanced Function Practice 33pp.pptx
(5) Python Recursion 44pp.pptx
(5-A) Python Recursion Practice 10pp.pptx
(6) Python Tuple & Set & Dictionary 60pp.pptx
(7) Code-Reading Recitation-A 18pp.pptx
(8) Pyhton File IO and Exceptions 13pp.pptx
(9) Sorting 15pp.pptx
(10) Search 33pp.pptx
(11) Simulation 40pp.pptx
(12) Data Structures 20pp.ppt
(13) Python Module and Package 11pp.pptx
(14) Python OOP 65pp.pptx
(15) Code-Reading Recitation-B 11pp.pptx
(16) Python Standard Libraries 114pp.pptx
(17) Python Tkinter 82pp.pptx

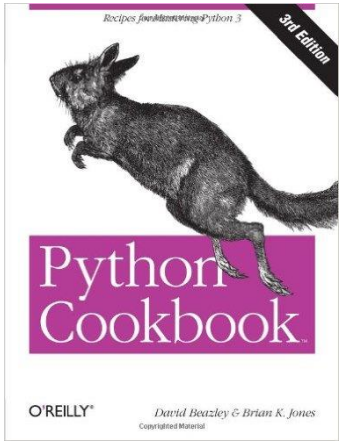
Python Introduction

Special Thanks to "2015년 1학기 컴퓨터개념및 실습"의조교 유강민 박사과정

· Python Books







· Online Tutorials

https://docs.python.org/3/tutorial/

http://www.python-course.eu/index.php

http://interactivepython.org/courselib/static/thinkcspy/index.html

· Just "class notes + Googling" is Enough!

https://docs.python.org/3/

Python 3.6.0 documentation

Welcome! This is the documentation for Python 3.6.0, last updated Jan 13, 2017.

Parts of the documentation:

What's new in Python 3.6?
or all "What's new" documents since 2.0

Tutorial
start here

Library Reference
keep this under your pillow

Language Reference
describes syntax and language elements

Python Setup and Usage
how to use Python on different platforms

Python HOWTOs
in-depth documents on specific topics

Installing Python Modules
installing from the Python Package Index & other sources

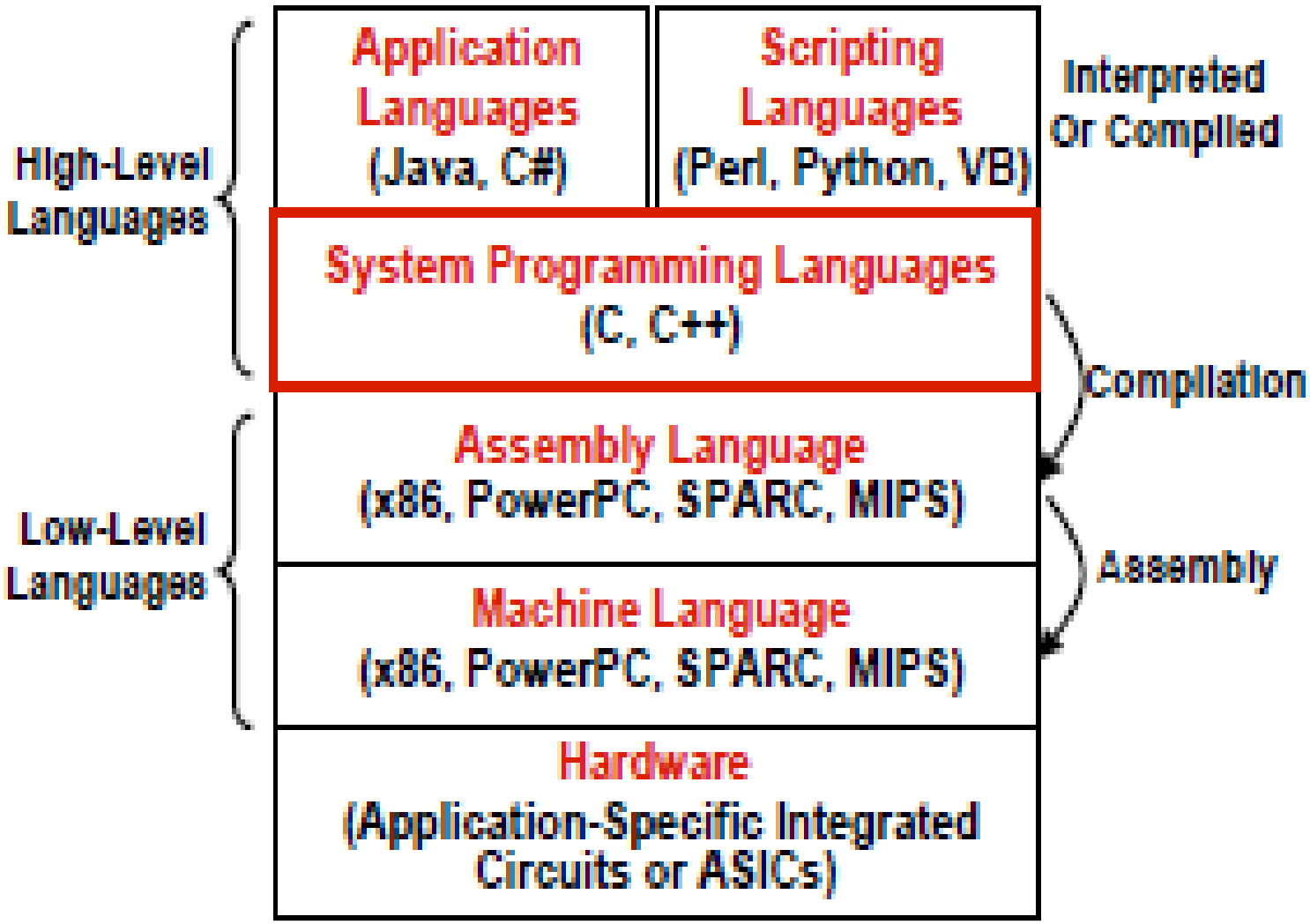
Distributing Python Modules
publishing modules for installation by others

Extending and Embedding
tutorial for C/C++ programmers

Python/C API
reference for C/C++ programmers

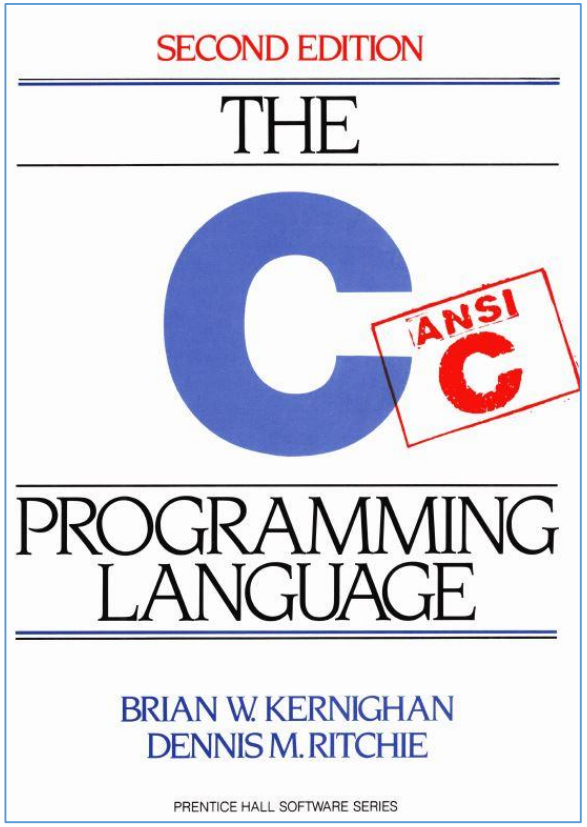
FAQs
frequently asked questions (with answers!)

Programming Levels



1972 by Kerninghan and Richie

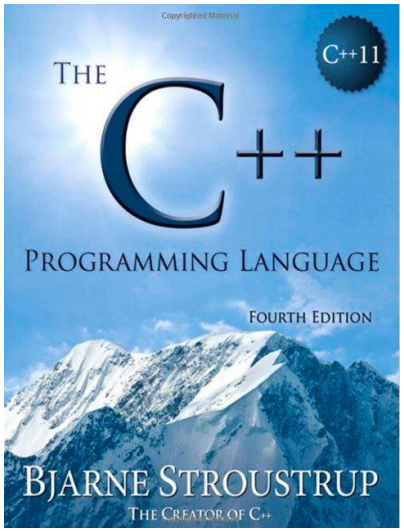
ISO/IEC: C99 (1999), C11(2011)



- Chapter 1. Tutorial Introduction
- Chapter 2. Types, Operators, and Expressions
- Chapter 3. Control Flow
- Chapter 4. Functions and Program Structure
- Chapter 5. Pointers and Arrays
- Chapter 6. Structures
- Chapter 7. Input and Output
- Chapter 8. The UNIX System Interface
- Appendix A. Reference Manual
- Appendix B. Standard Library
  - Input and Output: <stdio.h>
  - Character Class Tests: <ctype.h>
  - String Functions: <strings.h>
  - Mathematical Functions: <math.h>
  - Utility Functions: <stdlib.h>
  - Diagnosics: <assert.h>
  - Variable Argument Lists: <stdarg.h>
  - Non-local Jumps: <setjmp.h>
  - Signals: <signal.h>
  - Date and Time Functions: <time.h>
  - Implementation-defined Limits: <limits.h> and <float.h>

C++ 1983 by Bjarne Stroustrup

C++11 (2011)



Contents

Preface

Preface to Second Edition

Preface to First Edition

Introductory Material

1 Notes to the Reader
2 A Tour of C++
3 A Tour of the Standard Library

Part I: Basic Facilities

4 Types and Declarations
5 Pointers, Arrays, and Structures
6 Expressions and Statements
7 Functions
8 Namespaces and Exceptions
9 Source Files and Programs

Part II: Abstraction Mechanisms 221

10 Classes 223
11 Operator Overloading 261
12 Derived Classes 301
13 Templates 327
14 Exception Handling 355
15 Class Hierarchies 389

Part III: The Standard Library 427

16 Library Organization and Containers 429
17 Standard Containers 461
18 Algorithms and Function Objects 507
19 Iterators and Allocators 549
20 Strings 579
21 Streams 605
22 Numerics 657

Part IV: Design Using C++ 689

23 Development and Design 691
24 Design and Programming 723
25 Roles of Classes 765

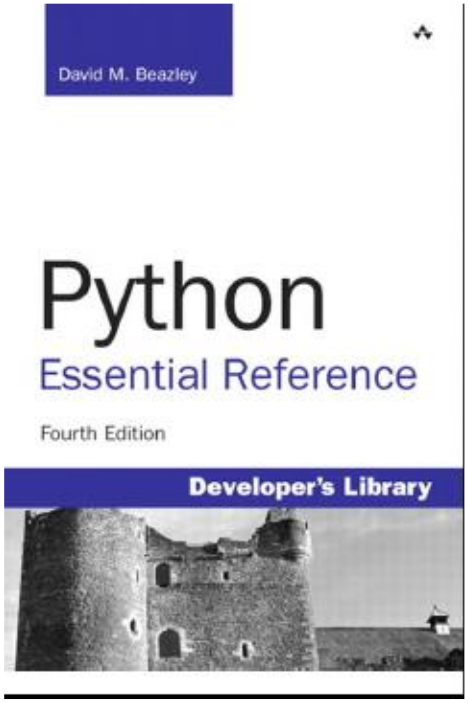
Appendices 791

A The C++ Grammar 793
B Compatibility 815
C Technicalities 827

Index 869

Python 1991 by Guido Rossum
Now owned by Python Org

Python 3.6 (2016)



Contents at a Glance

Introduction 1

Part I: The Python Language

1 A Tutorial Introduction 5
2 Lexical Conventions and Syntax 25
3 Types and Objects 33
4 Operators and Expressions 65
5 Program Structure and Control Flow 81
6 Functions and Functional Programming 93
7 Classes and Object-Oriented Programming 117
8 Modules, Packages, and Distribution 143
9 Input and Output 157
10 Execution Environment 173
11 Testing, Debugging, Profiling, and Tuning 181

Part II: The Python Library

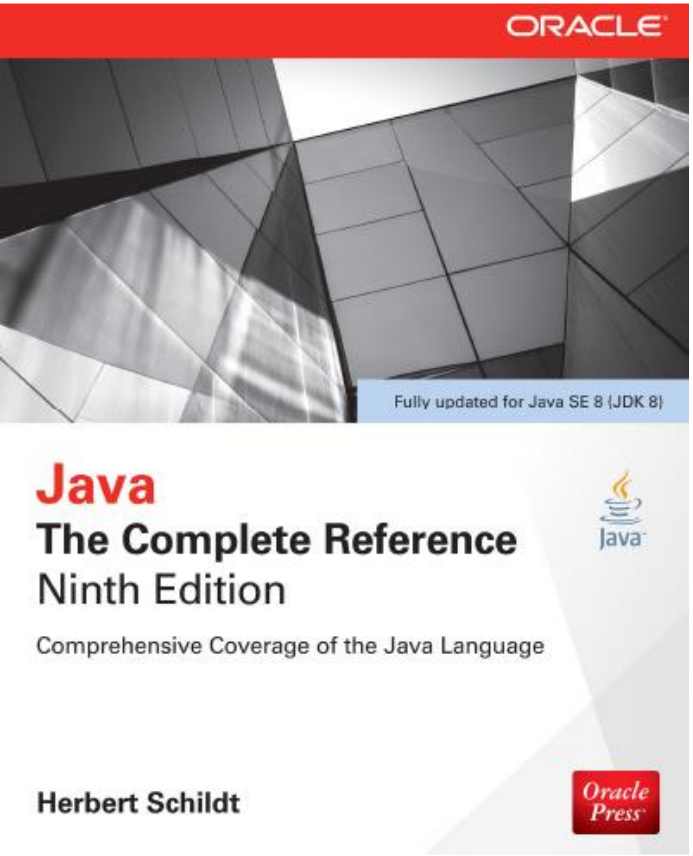
12 Built-in Functions 201
13 Python Runtime Services 219
14 Mathematics 243
15 Data Structures, Algorithms, and Code Simplification 257
16 String and Text Handling 277
17 Python Database Access 297
18 File and Directory Handling 313
19 Operating System Services 331
20 Threads and Concurrency 413
21 Network Programming and Sockets 449
22 Internet Application Programming 497
23 Web Programming 531
24 Internet Data Handling and Encoding 545
25 Miscellaneous Library Modules 585

Part III: Extending and Embedding

26 Extending and Embedding Python 591
Appendix: Python 3 621
Index 639

Java 1995 by James Gosling
Now owned by Oracle

Java 8 (2014)



Contents at a Glance

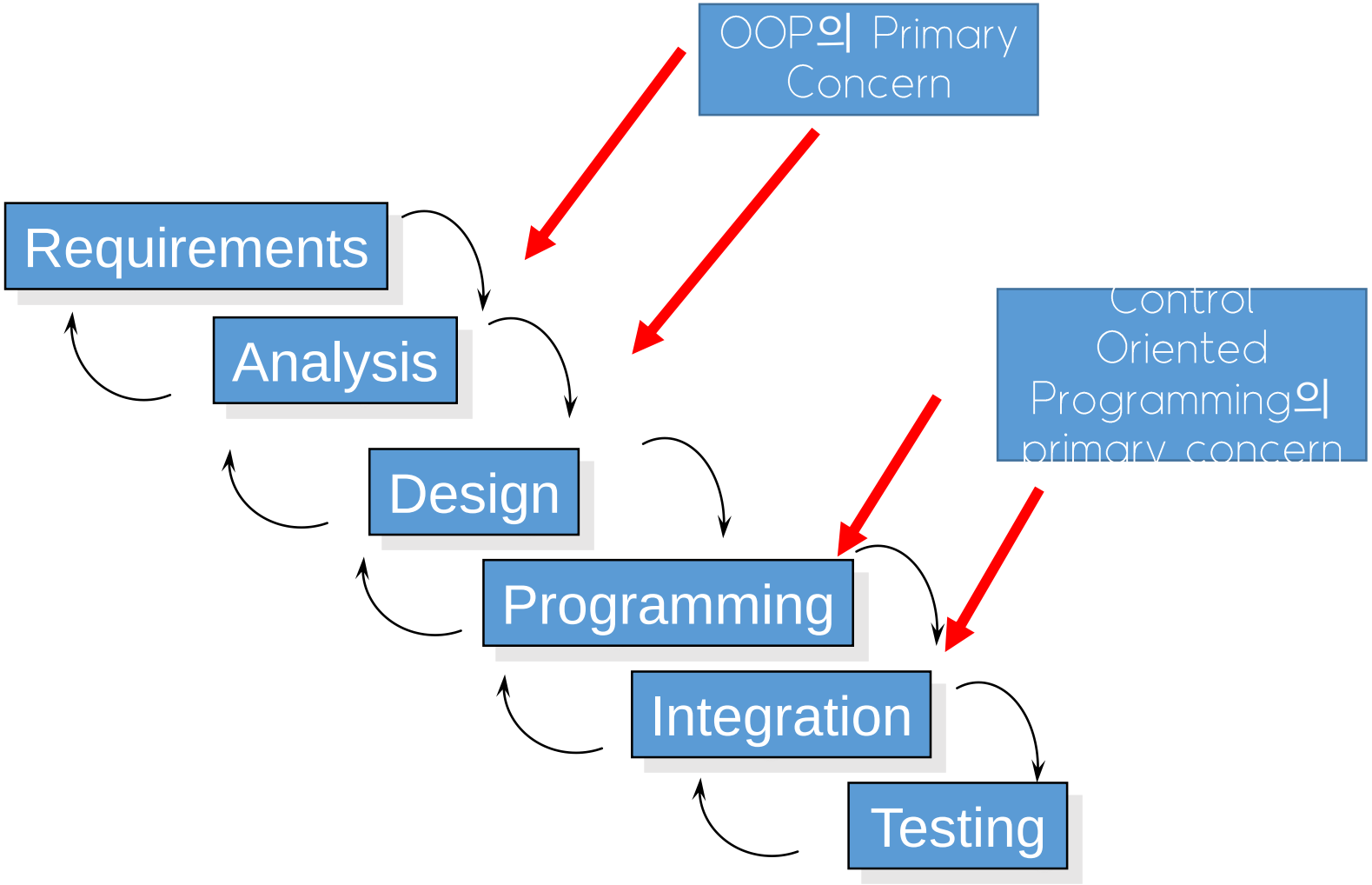
Part I	**The Java Language**	
1	The History and Evolution of Java	3
2	An Overview of Java	17
3	Data Types, Variables, and Arrays	35
4	Operators	61
5	Control Statements	81
6	Introducing Classes	109
7	A Closer Look at Methods and Classes	129
8	Inheritance	161
9	Packages and Interfaces	187
10	Exception Handling	213
11	Multithreaded Programming	233
12	Enumerations, Autoboxing, and Annotations (Metadata)	263
13	I/O, Applets, and Other Topics	301
14	Generics	337
15	Lambda Expressions	361
Part II	**The Java Library**	
16	String Handling	413
17	Exploring java.lang	441
18	java.util Part 1: The Collections Framework	497
19	java.util Part 2: More Utility Classes	579
20	Input/Output: Exploring java.io	641
21	Exploring NIO	689
22	Networking	727
23	The Applet Class	747
24	Event Handling	769
25	Introducing the AWT: Working with Windows, Graphics, and Text	797
26	Using AWT Controls, Layout Managers, and Menus	833
27	Images	885
28	The Concurrency Utilities	915
29	The Stream API	965
30	Regular Expressions and Other Packages	991
Part III	**Introducing GUI Programming with Swing**	
31	Introducing Swing	1021
32	Exploring Swing	1041
33	Introducing Swing Menus	1069
Part IV	**Introducing GUI Programming with JavaFX**	
34	Introducing JavaFX GUI Programming	1105
35	Exploring JavaFX Controls	1125
36	Introducing JavaFX Menus	1171
Part V	**Applying Java**	
37	Java Beans	1199
38	Introducing Servlets	1211
Appendix	Using Java's Documentation Comments	1235
	Index	1243

Programming Languages: Compiler vs Interpreter

- Compiled programs generally run **faster** since the translation of the source code happens only once.
- Once program is compiled, it can be executed over and over without the source code or compiler.

- Interpreted programs are more **portable**, meaning the same program can run on a Intel PC and on a Mac as long as the interpreter is available
- Interpreted languages are part of a more **flexible** programming environment since they can be developed and run interactively

Waterfall SW Development Model



The Software Development Process: The WaterFall Model

- Analyze the Problem
  - Figure out exactly the problem to be solved.
- Determine Specifications
  - Describe exactly what your program will do. (not How, but What)
  - Includes describing the inputs, outputs, and how they relate to one another.
- Create a Design
  - Formulate the overall structure of the program. (*how* of the program gets worked out)
  - You choose or develop your own algorithm that meets the specifications.
- Implement the Design (coding!)
  - Translate the design into a computer language.
- Test/Debug the Program
  - Try out your program to see if it worked.
  - Errors (Bugs) need to be located and fixed. This process is called debugging.
  - Your goal is to find errors, so try everything that might "break" your program!
- Maintain the Program
  - Continue developing the program in response to the needs of your users.
  - In the real world, most programs are never completely finished – they evolve over time.

Why Python?

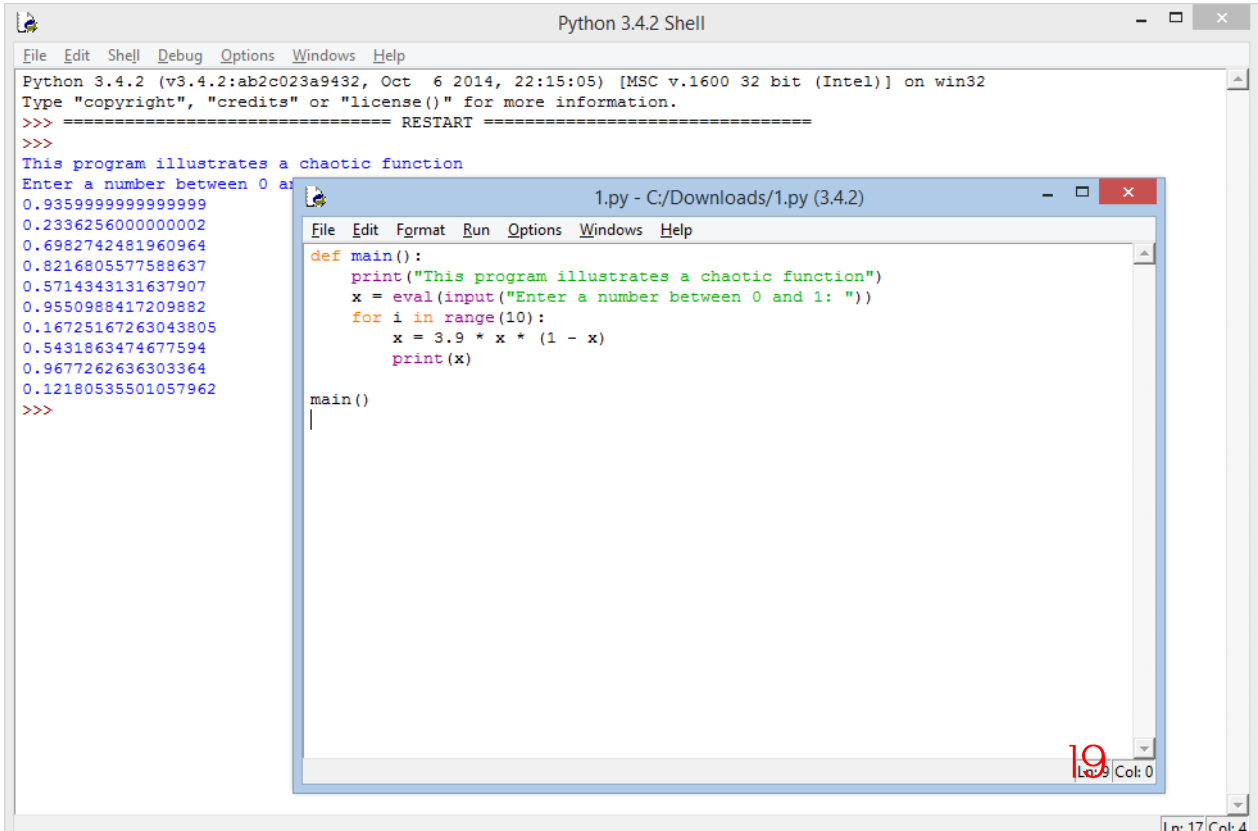
- General-purpose, High-level, Scripting Language
- First appeared 1991, invented by Guido van Rossum
- Easy to use, easy to learn
- Widely used as
  - Scientific libraries
  - Web Frameworks
  - Backend Frameworks
  - UI Frameworks
  - Graphic Frameworks
  - Data Mining Frameworks
  - And many others…

Why Python?: Advantages vs Disadvantages

- Advantages
  - Fast prototype testing
  - Minimal development effort
  - High readability
- Disadvantages
  - As a scripting language, it requires a interpreter
  - Performance might be an issue (memory, computation)
  - Weak typing might be harder to debug

Ways to Use Python: Python IDLE [1/6]

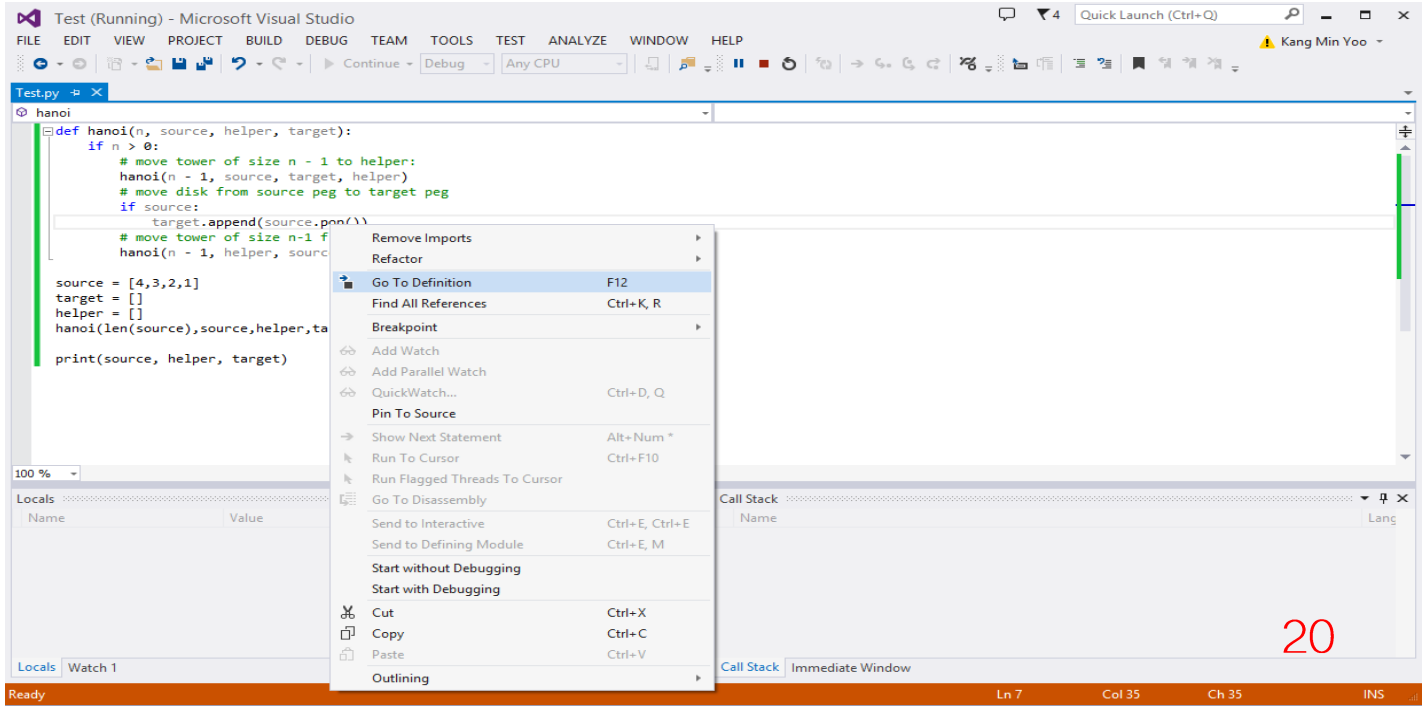
- Easy to use Interactive Development Environment (IDE)
- De-facto standard IDE for learning Python
- Provides simple debugging tool
- Provides simple **code completion**



Ways to Use Python: Python Tools for Visual Studio [2/6]

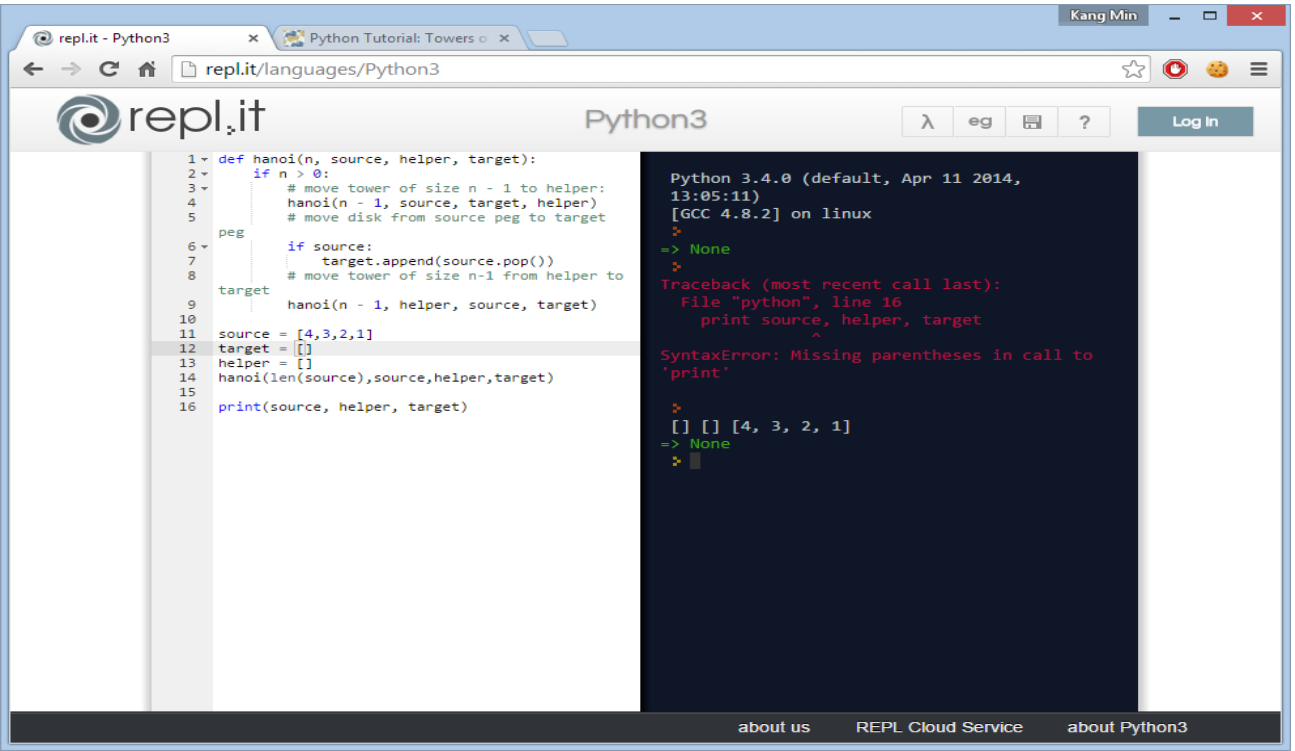
https://www.visualstudio.com/en-us/features/python-vs.aspx

- Has a steep learning curve, but very useful if used right
- Might be difficult for beginners in programming
- Supports most visual studio features
  Finding references // Code completion // Syntax checking
  Simple semantics checking // Full stack Debugging
  Inspection // And many others…



Ways to Use Python: http://repl.it/ [3/6]

- Surprisingly good and very easy to use
- Requires no installation of the interpreter on the machine
- Can be used interactively
- However, only Python 3.4.0 is available
  - The latest Python version is 3.4.3
- Scripts might be interpreted differently



Ways to Use Python: SublimeText3 with Python [4/6]

~/workspaces/pomodoro_web/functional_tests.py (pomodoro_web) - Sublime Text (UNREGISTERED)

FOLDERS
- pomodoro_web
  - pomodoro_web
  - tasks
  - .gitignore
  - db.sqlite3
  - functional_tests.
  - manage.py
  - pomodoro_web.

functional_tests.py | pomodoro_web.sublime-project

```python
import sys
print(sys.version)
from selenium import webdriver

browser = webdriver.Firefox()

browser.get('http://localhost:8000')

assert 'To-Do' in browser.title
browser.quit()
```

REPL [python] | *REPL* [python] | *REPL* [python] | *REPL* [python] | on]

```
3.4.3+ (default, Oct 14 2015, 16:03:50)
[GCC 5.2.1 20151010]
Traceback (most recent call last):
  File "functional_tests.py", line 9, in <module>
    assert 'To-Do' in browser.title
AssertionError

***Repl Closed***
```

Line 10, Column 15 | Tab Size: 4 | Python

Ways to Use Python: Eclipse with Python [5/6]

PyDev - test/test.py - Eclipse SDK

File Edit Source Refactoring Navigate Search Project Pydev Run Window Help

Quick Access | PHP | PyDev

PyDev Package Explorer

- test
  - test.py
  - python (C:₩Python33₩python.exe)
- RemoteSystemsTempFiles

test (test)

```
1 print('Hello World')
```

Console

<terminated> C:₩Users₩동진₩workspace₩test₩test.py

```
Hello World
```

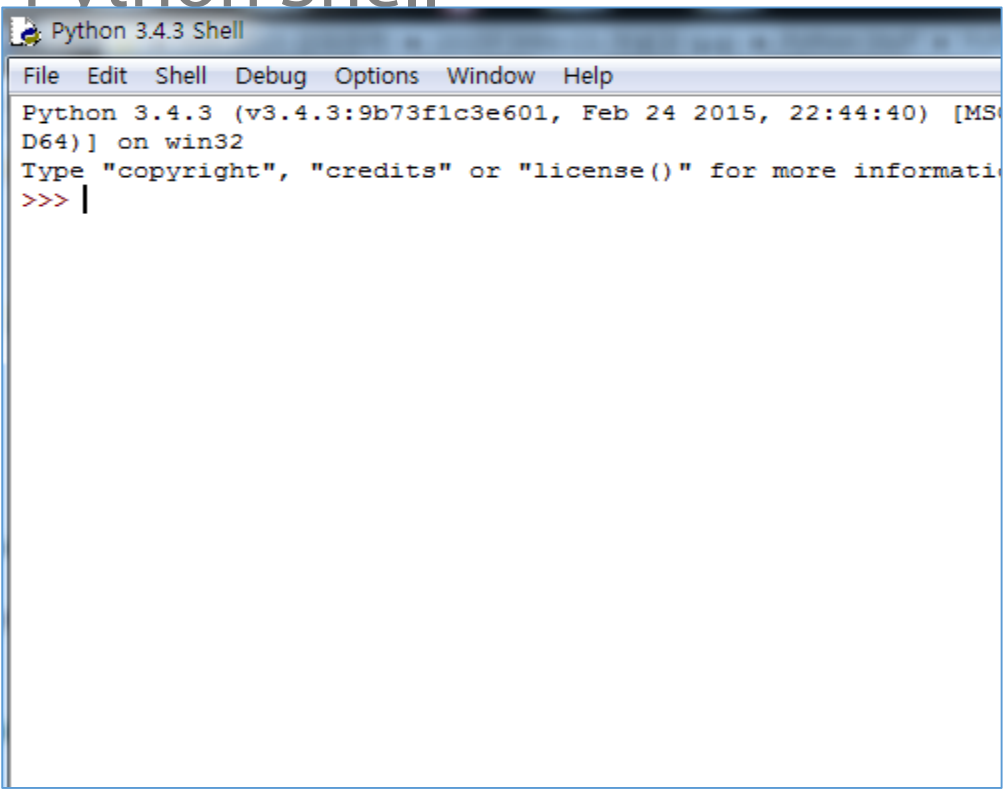
Writable | Insert | 1 : 21

Ways to Use Python [6/6]

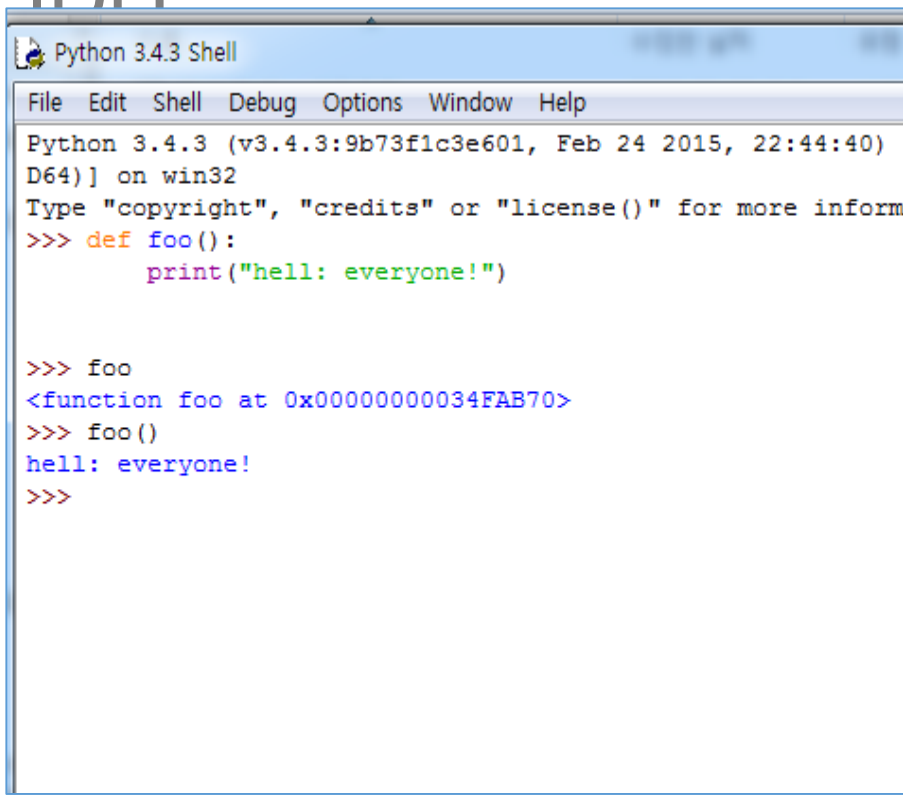
- Repl.it
  - Fast
  - Portable
  - Suitable for prototype testing
- Python IDLE
  - Readily available in the official python install package
  - Fairly easy to use
  - Features debugging
- Python Tools for Visual Studio
  - Contains the complete feature for programmers
  - The learning curve might be steep
  - Debugging, Refactoring, Syntax Checking, Syntax Highlighting, Dependency Management, and many more…
- SublimeText2 and Python
- Eclipse and Python

IDLE Screen Shots [1/5]

Initial Screen of IDLE: Python Shell



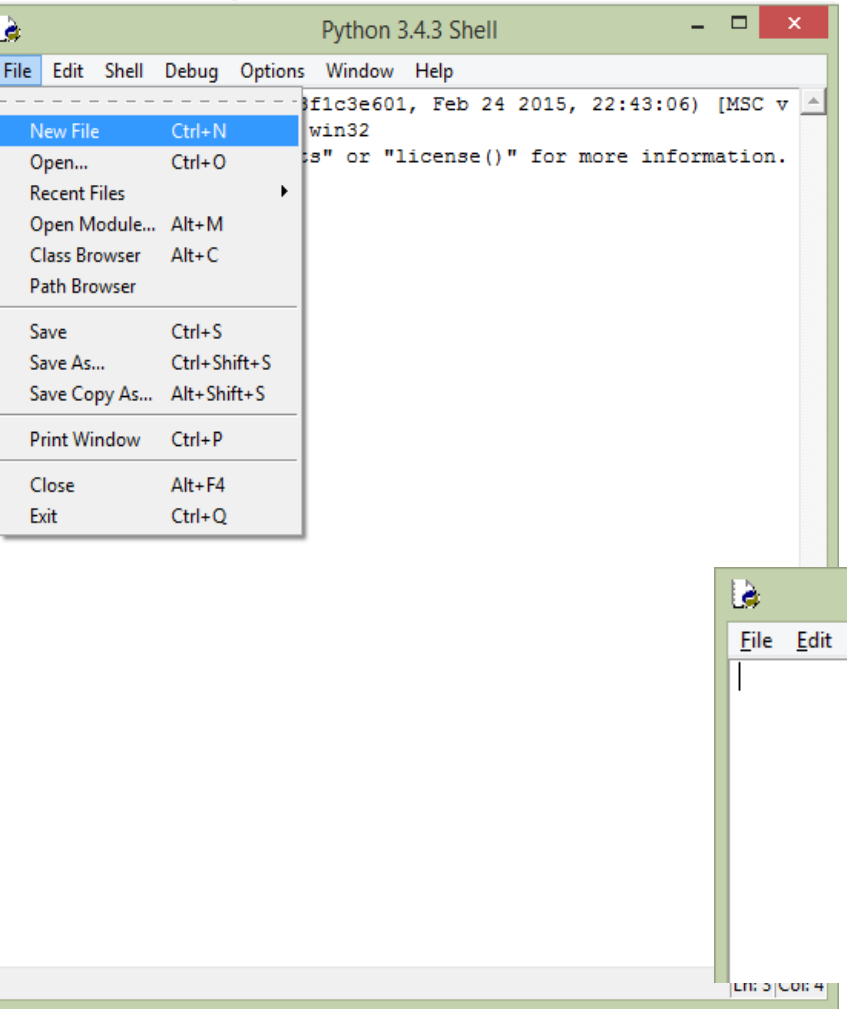
Initial Coding in IDLE



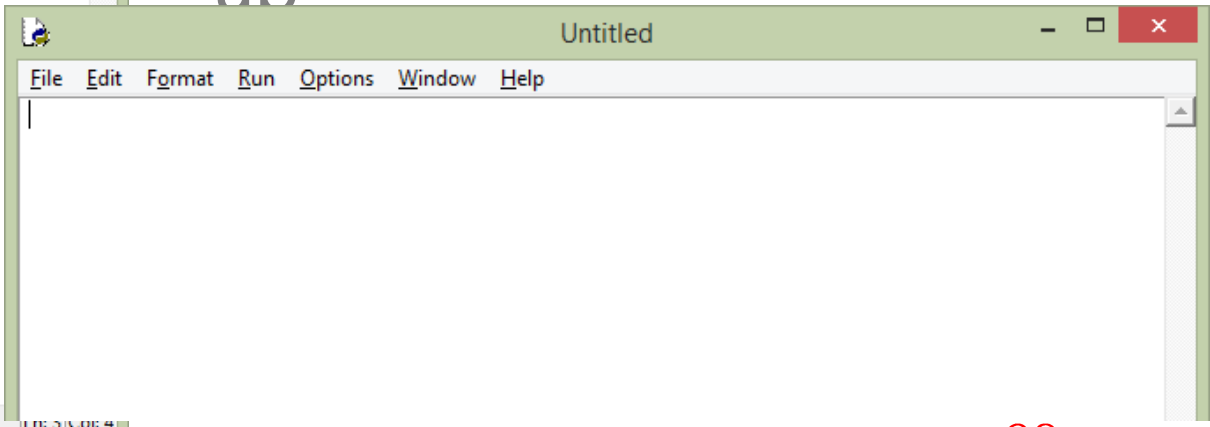
IDLE Screen Shots [2/5]

Suppose you finish up coding into IDLE
and
you want to save your Python code in your directory!

Creating a new script file in IDLE

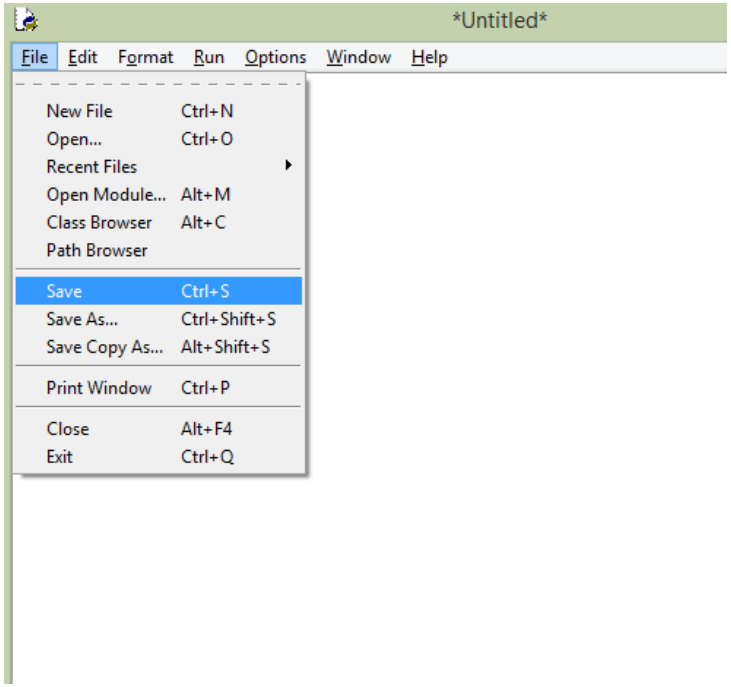


A new "untitled" window for a new script is poped up



IDLE Screen Shots [3/5]

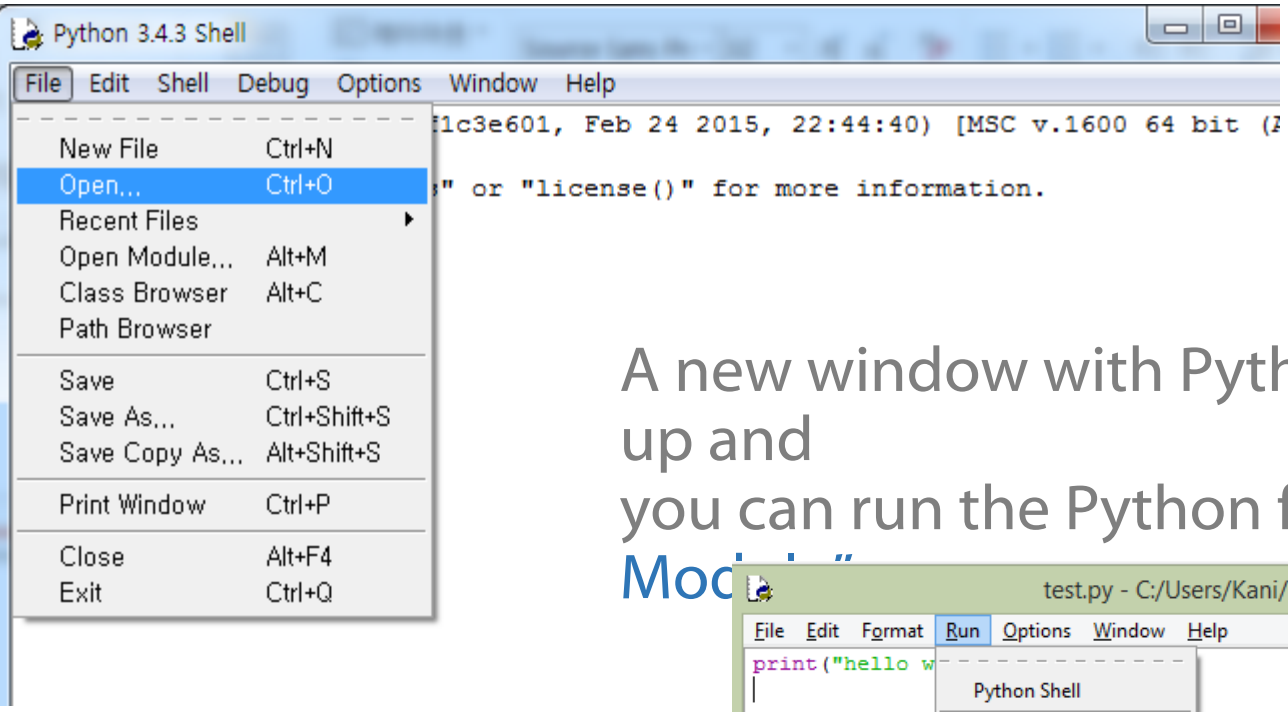
- Cut & Paste Python codes in IDLE window to "untitled" window
- Then, save the code as a new Python file (say, test.py)



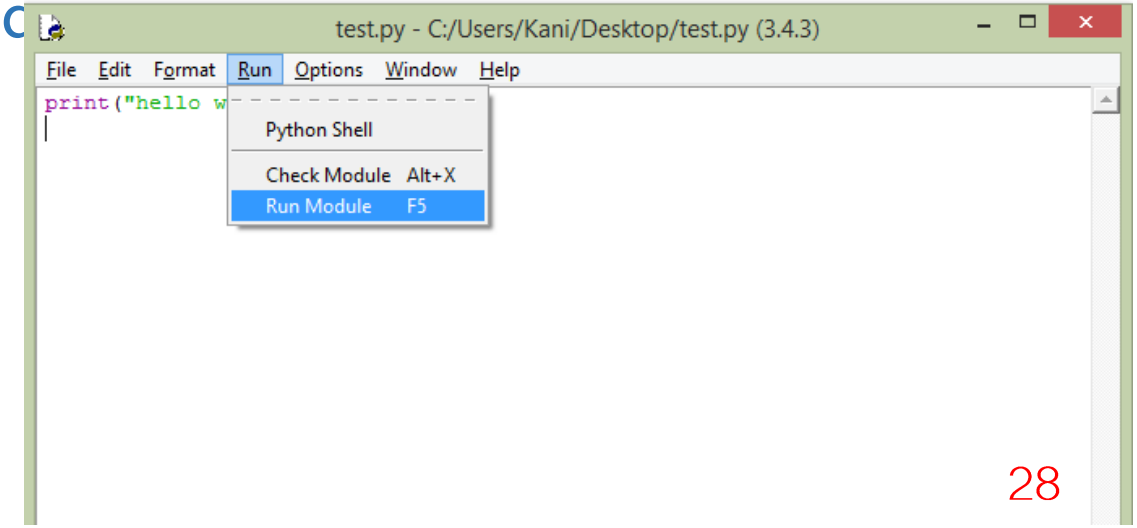
- Now you have "test.py" in your directory

IDLE Screen Shots [4/5]

If you want to read an existing Python file (say, test.py) into IDLE



A new window with Python file name is poped up and
you can run the Python file by clicking "Run Mod



IDLE Screen Shots [5/5]

Test results are displayed in a new (existing) shell window