w01-Lab2

Getting Started

Assembled for 204217

2015 S1

by Kittipitch Kuptavanich

Guttag, John V. *Introduction to Computation and Programming Using Python, Revised*

Programming Languages

- ภาษาที่ใช้ในการเขียนโปรแกรม (หรือ **Programming Language**) มีจำนวนมาก (หลายร้อยภาษา)
- แต่ละภาษามีความเหมาะสมกับงานที่ใช้มากน้อยต่างกันไป
  - **MATLAB** เป็นภาษาที่เหมาะสมกับงานประเภทที่เกี่ยวข้องกับการคำนวณ **Vector** และ **Matrix**
  - **C** เป็นภาษาที่เหมาะสำหรับงานด้าน **Data Network** หรืองานที่เกี่ยวกับ **Hardware**
  - **PHP** เป็นภาษาที่เหมาะสำหรับงานสร้าง **Web Site**
  - **Python** เป็นภาษาที่เหมาะกับงานทั่วไป (**General-purpose Language**)

Python Language

- ภาษา **Python** มีข้อดีคือ
  - เป็นภาษาที่ไม่ซับซ้อนและง่ายที่จะเรียนรู้
  - ในระหว่าง **run** จะได้ **Runtime Feedback** ที่เป็นประโยชน์สำหรับผู้เขียนโปรแกรมในชั้นเริ่มต้น
  - มี **Library** เพิ่มเติมจำนวนมากที่สามารถดาวน์โหลดได้ โดยไม่เสียค่าใช้จ่าย เพื่อนำมาใช้เพิ่มความสามารถของ **Python** ในการทำงานต่าง ๆ
  - ภาษา **Python** เหมาะสำหรับงานเขียนโปรแกรมทุกประเภทที่ไม่เกี่ยวข้องกับการเข้าถึงระดับ **Hardware** โดยตรง

Guttag, John V. *Introduction to Computation and Programming Using Python, Revised*

Python Language [2]

- **Python เป็นภาษาในการเขียนโปรแกรมระดับสูง (like C, C++, Perl, and Java)**
- **There are also Low-level languages referred to as machine languages or assembly languages**
- **Computer can only execute program in low-level language**

A First Book of ANSI C, 4th Edition

Low-level Language

- **Low-level Language (ภาษาระดับต่ำ): use instructions that are directly tied to one type of computer**

  ภาษาระดับต่ำใช้คำสั่งที่ขึ้นอยู่กับชนิดของเครื่องคอมพิวเตอร์

  - **ภาษาเครื่อง (Machine Language)**
  - **ภาษาแอสเซมบลี (Assembly Language)**

A First Book of ANSI C, 4th Edition

Machine Language

- ภาษาเครื่องเป็นชุดคำสั่งที่อยู่ในรูปของเลขฐานสอง ซึ่งเป็นคำสั่งที่เครื่องคอมพิวเตอร์เข้าใจได้โดยที่ไม่ต้องมีตัวแปลภาษา ตัวอย่างเช่น

```
11000000000000000001000000000010
11110000000000000001000000000011
```

A First Book of ANSI C, 4th Edition

Assembly Language

- เป็นภาษาที่พัฒนาต่อมาจากภาษาเครื่อง จึงมีความใกล้เคียงกับภาษาเครื่องมาก แต่ยังต้องการตัวแปลภาษา
- **Assembler:** ใช้สำหรับแปลภาษา **Assembly** ไปเป็นภาษาเครื่อง
- ตัวอย่างคำสั่งที่เขียนด้วยภาษา **Assembly**

```
LOAD   first
 ADD   second
 MUL   factor
STORE   answer
```

A First Book of ANSI C, 4th Edition

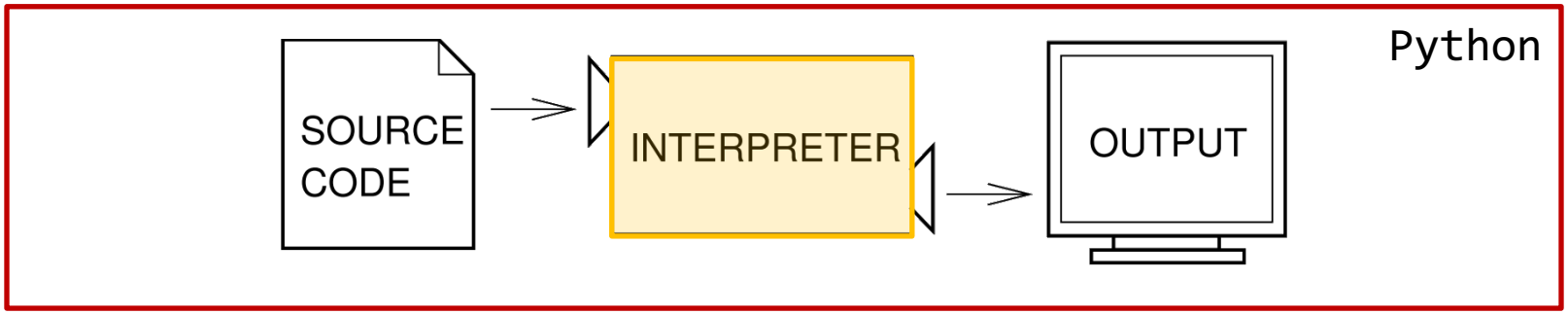
Assembly Language (2)

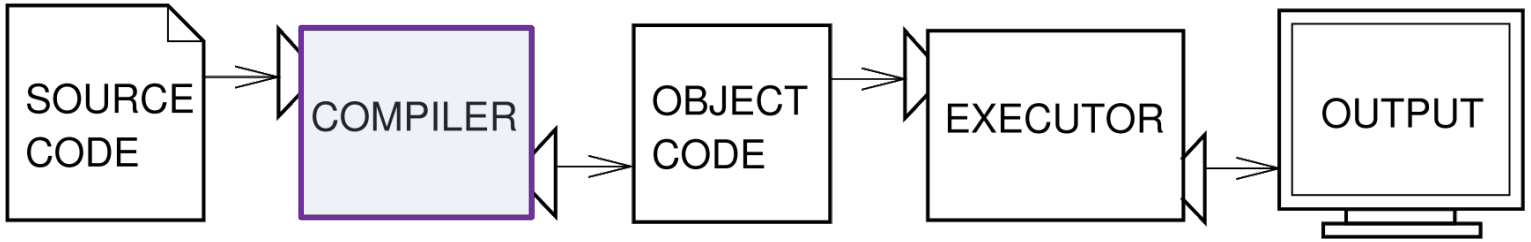
Figure 1.9 Assembly programs must be translated

Interpreter vs Compiler

- **Two kinds of programs process high-level languages into low-level languages:**
  - **Interpreters**
    - **อ่าน Program แล้ว execute ทีละส่วน**
  - **Compilers**
    - **อ่าน Program ทั้งหมดและแปลเป็น Machine Language ก่อนแล้ว execute ทีเดียว**

Python Language





- **Python จัดอยู่ในประเภท Interpreted Language**

https://en.wikipedia.org/wiki/CPython

Python Interpreter

- **Python Interpreter ที่ใช้ใน Class นี้คือ CPython (https://www.python.org/) ซึ่งเป็น Implementation มาตรฐานอ้างอิง โดย Guido van Rossum ผู้ให้กำเนิดภาษา Python และเป็น Implementation ที่มีผู้ใช้มากที่สุด**
- **นอกจากนี้ยังมี Python Interpreter อื่น ๆ เช่น**
  - **Jython, written in Java for the JVM**
  - **PyPy, written in RPython**
  - **IronPython, written in C#**
- **โดย Interpreter ใน CPython จะมีสอง Mode คือ Command-line Mode และ Script Mode**

Interactive Mode

```
Python 3.4.3 (default, May  5 2015, 17:04:32)
[GCC 4.9.2] on cygwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> print(1 + 1)
2
```

- ข้อดีของภาษาที่เป็น Interpreter
  - REPL (Read-Eval-Print-Loop)
- นอกจากการ install Python ลงในเครื่องแล้วเราสามารถลองใช้ Online Version ของ Python Interactive Interpreter ได้ที่ http://repl.it/languages/Python

Script Mode

```
$ python test.py
2
```

- **Script Mode คือการเขียน Code ทั้งหมดลงในไฟล์ (นามสกุล .py) และให้ Interpreter ทำการ execute Code ใน ไฟล์นั้น**
- **เราเรียกไฟล์ที่มี Python Code นั้นว่า Script**

Installing Python

- **Version 3.4**
- **Current Release:**
  http://www.python.org/downloads/release/python-343/
- **Integrated Development Environment (IDE)**
  - **Python มี built-in IDE ชื่อ IDLE**
  - **หรือสามารถใช้ Text Editor อื่น ๆ**
- **สามารถใช้ Installation Package (Python + Cygwin + Editor etc.) ได้จาก Website ของรายวิชา**

http://programmers.stackexchange.com/questions/206755/ide-vs-text-editor

Text Editor vs IDE

ในกระบวนวิชานี้เราใช้ IDLE ซึ่งเป็น IDE ที่มากับ package Python มาตรฐาน

Integrated Development Environment

- **Strengths**
  - Integrated testing
  - Compilation
  - Breakpoints/stepping through code
  - Integration with other services (database views), automated class diagrams
- **Weaknesses**
  - Large memory footprint
  - Cost
  - low support for code completion (intellisense features)

Text Editor

- **Strengths**
  - Fast
  - Easy to extend (macros, plugins)
  - Text edit functions (Ex: sublime text 2 unending keyboard shortcuts)
- **Weaknesses**
  - Need to use another service to compile

Basic Program Instructions

A few basic instructions appear in just about every language:

- **Input**
- **Output**
- **Math**
- **Conditional Execution**
- **Repetition**

เราสามารถพิจารณาการเขียนโปรแกรมว่าเป็นการแบ่งปัญหาที่ใหญ่และซับซ้อน ลงเป็นปัญหาย่อยที่เล็ก และซ้อบซ้อนน้อยลง จนกว่าจะสามารถแก้ปัญหาย่อย ๆ นั้น ๆ ได้ ด้วยชุดคำสั่งพื้นฐานดังกล่าว

http://www.kosbie.net/cmu/spring-13/15-112

Python Basics

- **Hello World**

```
>>> print("Hello World")
Hello World
```

  - **หรือ Script Mode**

    - **สร้างไฟล์ใน bash shell ชื่อ `myscript.py` แล้ว run**

```
$ echo 'print("Hello World")' > myscript.py
$ python myscript.py
Hello World
```

http://www.kosbie.net/cmu/spring-13/15-112

Python Comments

- **Comments**

```
>>> print("Hello World")      # this is a comment
Hello World
>>> # Another comment
>>>
```

```
23 ''' this is also
24 A comment  (script mode only)'''
25
26 """ double quotes and single quotes
27 are the same"""
```

http://www.kosbie.net/cmu/spring-13/15-112

Line Delimiter

- ในภาษา python ไม่จำเป็นต้องมีเครื่องหมายแสดงการจบบรรทัด (ใน C ใช้ ;)
- Python ใช้การขึ้นบรรทัดใหม่ (NEWLINE) เป็นการแยกคำสั่งแต่ละบรรทัด
- หากจำเป็นต้องเขียน expression ข้ามบรรทัด ทำได้โดยการใช้เครื่องหมาย backslash \ หรือ วงเล็บ ( )

```
>>> 1 + 3 + 4 \
    + 5
13
>>> (1 + 3 + 4
     + 5)
13
```

Explicit line joining

Implicit line joining

http://www.kosbie.net/cmu/spring-13/15-112

Syntax, Runtime and Logical Errors

- **Syntax Errors (Compile - Time Errors)**

```
>>> print("Uh oh!)  # ERROR!  Untrue!!!
SyntaxError: EOL while scanning string literal
```

- **Runtime Errors ("Crash")**

```
>>> print(1 / 0)    # ERROR!  Division by zero!
ZeroDivisionError: integer division or modulo by zero
```

- **Logical Errors(Compiles and Runs, but is Wrong!)**

```
>>> print("2 + 2 = 5")     # ERROR!  Untrue!!!
2 + 2 = 5
```

http://www.kosbie.net/cmu/spring-13/15-112

Basic Console Output

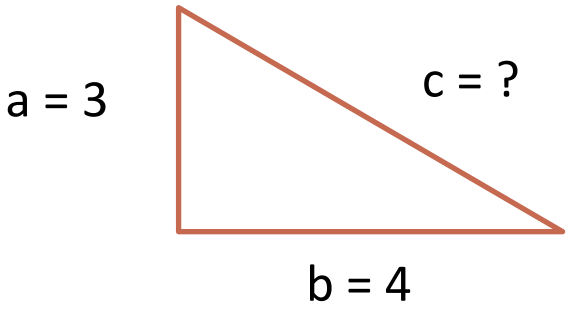
```
01 # Basic print() Function
02 print("1: Carpe")
03 print("diem")
04
05 # Print on the Same Line
06 print("2: Carpe", end="")
07 print("diem")
08
09 # Print Multiple Items
10 print("3: Carpe", "diem")
```

```
1: Carpe
diem
2: Carpediem
3: Carpe diem
```

http://www.kosbie.net/cmu/spring-13/15-112

Basic Console Output [2]

- **การหาด้านตรงข้ามมุมฉากของสามเหลี่ยม (Hypotenuse)**



$$c^2 = a^2 + b^2$$
$$c = \sqrt[2]{a^2 + b^2}$$
$$= (a^2+b^2)^{\frac{1}{2}}$$

```
03 # Compute the hypotenuse of a right triangle
04 a = 3
05 b = 4
06 c = ((a ** 2) + (b ** 2)) ** 0.5
07 print("side a =", a)
08 print("side b =", b)
09 print("hypotenuse c = {0:.2f}".format(c))
```

http://www.kosbie.net/cmu/spring-13/15-112

Basic Console Input

```
# String with input()
>>> name = input("Enter your name: ")
Enter your name: Jon Snow
>>> print("Your name is:", name)
Your name is: Jon Snow

# Number with input() (error!)
>>> x = input("Enter a number: ")
Enter a number: 6
>>> print("One half of", x, "=", x / 2)
TypeError: unsupported operand type(s) for /: 'str' and 'int'

# Number with input() and int()
>>> x = int(input("Enter a number: "))
Enter a number: 6
>>> print("One half of", x, "=", x / 2)
One half of 6 = 3.0
```

ฟังก์ชัน `input()` จะอ่านทีละบรรทัดจนพบ newline character `'\n'`

http://www.kosbie.net/cmu/spring-13/15-112

Importing Modules

```
# Fails
>>> print(math.sqrt(5))
NameError: name 'math' is not defined

# Work
>>> import math
>>> print(math.sqrt(5))
2.23606797749979

>>> help(math.sqrt)   # Also try dir(math) and help(math)
Help on built-in function sqrt in module math:

sqrt(...)
    sqrt(x)

    Return the square root of x.
```

กรณี **built-in Module** ใช้

```
dir(__builtins__)
```

หรือ

```
help(print)
```

https://docs.python.org/3.4/library/math.html

Math Functions

`math.ceil(x)`	`math.acos(x)`
`math.fabs(x)`	`math.asin(x)`
`math.factorial(x)`	`math.atan(x)`
`math.floor(x)`	`math.cos(x)`
`math.trunc(x)`	`math.sin(x)`
`math.exp(x)`	`math.tan(x)`
`math.log(x[, base])`	`math.degrees(x)`
`math.log2(x)`	`math.radians(x)`
`math.log10(x)`	`math.pi`
`math.pow(x, y)`	`math.e`
`math.sqrt(x)`	

http://www.kosbie.net/cmu/spring-13/15-112

Getting Started with Variables

```
# Define a variable and use it
>>> x = 3
>>> print(x)
3

# Same, but with a nicer UI
>>> x = 3
>>> print("x =", x)
x = 3

# Use a variable without defining it
>>> print("yikes =", yikes)
NameError: name 'yikes' is not defined
```

http://www.kosbie.net/cmu/spring-13/15-112

Getting Started with Variables [2]

```
# Assigning and re-assigning
>>> x = 3
>>> print("x =", x)
x = 3
>>> x = 4
>>> print("x =", x)
x = 4
# Using two variables
>>> x = 3
>>> y = 4
>>> print("x =", x)
x = 3
>>> print("y =", y)
y = 4
>>> print("x + y =", x + y)
x + y = 7
```

http://www.kosbie.net/cmu/spring-13/15-112

GETTING STARTED WITH FUNCTIONS

http://www.kosbie.net/cmu/spring-13/15-112

Function Basics

```
# functions with no parameters
>>> def do_something():
        print("Hello")

>>> do_something()
Hello
>>> do_something()
Hello

# functions with one parameter
>>> def print_square(x):
        print(x, "** 2 =", (x * x))

>>> print_square(2)
2 ** 2 = 4
>>> print_square(3)
3 ** 2 = 9
```

Function Call

- เราสามารถเรียกใช้ฟังก์ชันจากใน **script** ได้โดยตรง

```
01 #!/usr/bin/env python3
02
03 def hypotenuse(a, b):
04     h = ((a ** 2) + (b ** 2)) ** 0.5
05     return h
06
07 s1 = int(input("input a: "))
08 s2 = int(input("input b: "))
09 h = hypotenuse(s1, s2)
10
11 print("hypotenuse = {0:.2f}".format(h))
```

- หากไม่มีการเรียกใช้ฟังก์ชันที่สร้างขึ้นมา เมื่อ **run Script** ก็จะไม่มีการดำเนินการใด ๆ
- ในภาษา **Python** การเรียกใช้ฟังก์ชันหรือ **Function Call** จะต้องเกิดขึ้นหลังจาก **Function Definition** เสมอ

Think Python: How to Think Like a Computer Scientist

Function Call [2]

Python ไม่ได้จำกัดให้ใช้ชื่อ `main()`
- แต่เป็นชื่อ function หลักในภาษาอื่น ๆ
- การใช้ชื่อ `main()` ทำให้เข้าใจหน้าที่ของฟังก์ชันทันทีที่เห็น

- โดยปกติแล้ว ในโปรแกรมหนึ่ง ๆ จะต้องทำงานกับหลาย ๆ ฟังก์ชัน การนำคำสั่งดำเนินการหลักไว้ที่ส่วนท้ายสุดของไฟล์ อาจจะไม่สะดวกต่อการอ่านและแก้ไข
- เราสามารถนำคำสั่งดำเนินการหลักรวมไว้ในฟังก์ชัน `main()` ด้านบน
- แล้วเรียกใช้ `main()` ที่ส่วนล่างสุดของโปรแกรม

```
02 def main():
03     s1 = int(input("input a: "))
04     s2 = int(input("input b: "))
05     h = hypotenuse(s1, s2)
06     print("hypotenuse = {0:.2f}".format(h))
07
08 def hypotenuse(a, b):
09     h = ((a ** 2) + (b ** 2)) ** 0.5
10     return h
11
12 main()          # เรียกใช้ฟังก์ชันใน global scope
```

http://www.kosbie.net/cmu/spring-13/15-112

Function Examples

```
01 # functions with multiple parameters
02 def printSum(x, y):
03     print(x, "+", y, "=", x + y)
04
05
06 printSum(2, 3)
07 printSum(3, 4)
08
09
10 # int functions (Functions that return a value)
11 def square(x):
12     return x * x
13
14
15 print(square(3))
16 print(square(4))
17 x = square(3) + square(4)
18 print(x)
```

```
2 + 3 = 5
3 + 4 = 7
9
16
25
```

http://www.kosbie.net/cmu/spring-13/15-112

Function Examples [2]

```
01 # Functions calling other functions
02 def square(x):
03     return x * x
04
05
06 def printSquare(x):
07     print(x, "** 2 =", square(x))
08
09 printSquare(3)
10 printSquare(4)
```

```
3 ** 2 = 9
4 ** 2 = 16
```

http://www.kosbie.net/cmu/spring-13/15-112

Function Examples [3]

```
01 # Some other return types
02 def cubeRoot(d):
03     return d ** (1.0 / 3.0)
04
05
06 def isPositive(x):
07     return (x > 0)
08
09 print(cubeRoot(8))
10 print(isPositive(8))
11 print(isPositive(-8))
```

```
2.0
True
False
```

http://www.kosbie.net/cmu/spring-13/15-112

Function Examples [4]

```
01 # Local variables
02 def min_squared(x,y):
03     smaller = min(x,y)
04     return smaller * smaller
05
06 print(min_squared(3,4))
07 print(min_squared(4,3))
```

```
2.0
True
False
```

```
10 def is_even_positive(x):                # Another example:
11     is_even = ((x % 2) == 0)
12     is_positive = (x > 0)
13     return (is_even and is_positive)
14
15 print(is_even_positive(-2))
16 print(is_even_positive(-1))
17 print(is_even_positive(0))
18 print(is_even_positive(1))
19 print(is_even_positive(2))
```

```
False
False
False
False
True
```

http://www.kosbie.net/cmu/spring-13/15-112

Function Examples [5]

```
01 # Parameter and Local variable scope
02 def add_one(x):
03     x = x + 1
04     print("in add_one, x =", x)
05     return x
06
07 x = 5
08 print("first, x =", x)
09 result = add_one(x)
10 print("calling add_one(x) returned:", result)
11 print("and now x =", x)
```

```
first, x = 5
in add_one, x = 6
calling add_one(x)
returned: 6
and now x = 5
```

```
14 def print_n():       # Another example / Globals
15     print(n)
16 # n not local -- so it is global (bad idea!!!)
17
18 n = 5
19 print_n()
```

```
5
```

http://www.kosbie.net/cmu/spring-13/15-112

Function Examples [6]

min_sq.py

```
01 # Test functions
02 def min_squared(x, y):
03     smaller = min(x, y)
04     return smaller * smaller
05
06
07 def test_min_squared():
08     print("Testing min_squared... ")
09     assert(min_squared(2, 3) == 4)
10     assert(min_squared(3, 2) == 4)
11     print("Passed all tests!")
12
13
14 test_min_squared()
```

```
$ python min_sq.py
Testing min_squared...
Passed all tests!
```

Writing Modules

- ฟังก์ชันที่เราเขียนขึ้นใน Script หนึ่ง ๆ สามารถนำไปใช้ใน Script อื่น ๆ ได้ โดยการใช้คำสั่ง `import` ในลักษณะเดียวกับการ "`import math`" Module
- Script ที่เราเขียนฟังก์ชันต่าง ๆ ไว้ เมื่อเรียกใช้งานจากอีก Script ก็ถือเป็น module เช่นกัน

```
import min_sq                    # ใช้คำสั่ง import ตามด้วยชื่อไฟล์
                                 # โดยไม่ต้องใส่ .py
```

ชื่อ Module ต้องตรงกับชื่อไฟล์ (ตัวพิมพ์เล็ก และ underscore)

Writing Modules [2]

- ทั้งนี้หากในไฟล์ `min_sq.py` หากมีชุดคำสั่งอื่น ๆ นอกเหนือจาก **Function Definition** ใน **Global Scope**
  - เช่น การเรียกใช้ฟังก์ชันเพื่อการทำ **testing** `test_min_square()` หรือการเรียกใช้ฟังก์ชัน `main()` หรือชุดคำสั่งเหล่านั้นก็จะถูกเรียกใช้งานไปด้วยเมื่อมีการ `import`
    - ได้ **Output** ที่ **User** ไม่ต้องการ (**User** เพียงต้องการเรียกใช้ฟังก์ชัน)
- เราสามารถป้องกันการถูกเรียกใช้งานดังกล่าว โดยการ*ตั้งเงื่อนไข* ให้ชุดคำสั่งดังกล่าว ถูกเรียกใช้ในกรณีที่ **Script** ถูก **run** โดยตรงเท่านั้น (ไม่ถูก **run** ในกรณี `import`) ดังแสดงด้านล่าง

```
26 if __name__ == '__main__':
27     main()
```

References

- Guttag, John V. ***Introduction to Computation and Programming Using Python, Revised***
- Allen B. Downey ***Think Python:How to Think Like a Computer Scientist***
- Gary J. Bronson ***A First Book of ANSI C***, 4th Edition
- https://en.wikipedia.org/wiki/CPython
- https://docs.python.org/3/reference/lexical_analysis.html
- http://www.kosbie.net/cmu/spring-13/15-112/handouts/notes-getting-started.html
- http://www.kosbie.net/cmu/spring-13/15-112/handouts/notes-writing-functions.html