w02-Lab

Types, Literals, Variables, Operators, and Expressions

Assembled for 204211

by Kittipitch Kuptavanich

What is a Program?

- **A program is a sequence of instructions that specifies how to perform a computation.**
  - **Mathematical**
    - แก้ระบบสมการ
    - หารากต่าง ๆ ของพหุนาม (Polynomials)
  - **Symbolic Computation, such as**
    - ค้นหาและเปลี่ยนคำที่ต้องการในข้อความ
    - Compile โปรแกรม

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

Programming languages are formal languages that have been designed to express computations…

… and not natural languages

Natural Language

- **Spoken Language**
  ภาษาพูดเช่น ภาษาอังกฤษ
- **Evolve Naturally**
- **Unclear**

Formal Language

- **Used for specific application**
- **Designed by people**
- **To be nearly unambiguous**
  ค่อนข้างชัดเจนไม่คลุมเครือ
  - **A statement can have only one meaning**

Python

- **Interactive Mode หรือ Python Shell**
  - **REPL (Read, Eval, Print, Loop)**

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

- **Script Mode**

```
$ python test.py                                             bash shell
2
```

  - **Script Mode คือการเขียนคำสั่งทั้งหมดลงในไฟล์ Python โปรแกรม (นามสกุล .py) และให้ Interpreter ทำการ execute Code ใน ไฟล์นั้น ๆ**

IDE vs Text Editor

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**

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**
  - **low support for code completion (intellisense features)**

Values and Types

- **ใน Python มีชนิดข้อมูลพื้นฐาน อยู่สองประเภทคือ**
  - **แบบที่ไม่สามารถแบ่งย่อยลงไปได้อีก (atomic, scalar) มี 4 ชนิดได้แก่**
    - `int` – **แทนจำนวนเต็ม เช่น 3, -8**
    - `float` – **แทนจำนวนจริง เช่น 2.36**
    - `bool` – **แทนค่าทางตรรกะ** `True` **(จริง) หรือ** `False` **(เท็จ)**
    - `None` – **เป็นชนิดข้อมูลที่มีค่าเป็น None ได้อย่างเดียว**
  - **แบบที่สามารถแบ่งย่อยลงไปได้ เช่น**
    - `str` – **สายอักขระ เช่น 'hello' "ก" หรือ "" (สังเกตเครื่องหมายคำพูด) สามารถเข้าถึงข้อมูลแยกทีละอักขระได้**
    - `complex` – **จำนวนเชิงซ้อน ประกอบด้วย ส่วน Real และ ส่วน Imaginary เช่น** `1 + 2j`

Values and Types [2]

- ตัวเลข หรือ สายอักขระ (ที่อยู่ระหว่างเครื่องหมายคำพูด) ในโปรแกรมใด ๆ ถือเป็นค่าคงที่ (Literals)
- เราสามารถตรวจสอบชนิดของ Literals (และ Variable) ใน Python ได้โดยการใช้ ฟังก์ชัน `type()`

```
>>> type('Hello, World!')                    python shell
<class 'str'>
>>> type(17)
<class _____>
>>> type(3.2)
<class _____>
>>> type('17')
<class _____>
>>> type(071)
<class _____>
```


Variables

- **Variable** (ตัวแปร) เป็นชื่อที่ใช้อ้างถึงข้อมูล (Data Object)
- การสร้าง Variable ใน Python ทำได้โดยการตั้งค่าให้กับชื่อ โดยการใช้เครื่องหมาย = (เท่ากับ)

```
pi = 3.14
radius = 11
area = pi * radius * radius
```

- ตัวแปร area มีชนิดข้อมูลเป็น_____?
- คำสั่งที่ใช้สร้าง Variable ขึ้นพร้อม ๆ กับให้ค่าในลักษณะนี้ เรียกว่า Assignment Statement

Aside

Expressions and Statements

- An expression is a combination of values, variables, and operators.
  - An expression can be evaluated to a value
    เราสามารถประเมินค่าของ Expression ได้
- A statement is a unit of code that the Python interpreter can execute.
  Statement คือหน่วยย่อยของชุดคำสั่งที่ Python Interpreter ดำเนินการได้
  - For example, assignment statement
- Expression has value; a statement does not.

```
>>> x = 3       # statement
>>> x == 3      # expression
True
>>> x           # expression
3
```


Variables [2]

```
>>> theSum = 0
>>> theSum
0
>>> theSum = theSum + 1
>>> theSum
1
>>> theSum = True
>>> theSum
True
```

ในภาษา Programming หลาย ๆ ภาษาเช่น Python ชนิดของตัวแปรสามารถเปลี่ยนแปลงได้ (Dynamic Typing)

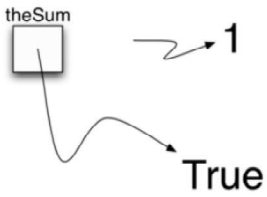


Figure 1.3: Variables Hold References to Data Objects

Figure 1.4: Assignment changes the Reference

Multiple Assignment

- ใน Python เราใช้เครื่องหมาย = เพื่อทำการ assign ค่าให้ Variable
  - แต่ = ไม่ได้มีความหมายเดียวกับเครื่องหมายเท่ากับในทางคณิตศาสตร์ (Equality Sign)
  - `a = 5` is legal but `5 = a` is not
- จากตัวอย่างก่อนหน้านี้ เราสามารถ assign ค่าให้ Variable ใด ๆ กี่ครั้งก็ได้ (Multiple Assignments)

```
>>> a = 7
>>> print(a)
7
>>> b = a
>>> print(b)
7
>>> a = 5
>>> print(a, b)
```


Updating Variable

- กรณีหนึ่งที่พบบ่อยในการทำ Multiple Assignment คือการ ***update*** ค่า Variable
  - ค่าใหม่ที่ assign มีความเกี่ยวข้องกับค่าเก่า

```
x = x + 1
```

  - มีความหมายคือ อ่านค่าจาก x, นำมาบวกด้วย 1 แล้ว assign x ด้วยค่าผลลัพธ์นั้น

```
>>> x = x + 1
NameError: name 'x' is not defined
```

  - การ update Variable ที่ไม่ได้มีการ assign ค่าไว้ก่อนจะเกิด Error
  - การเพิ่มค่า x ด้วย 1 ดังตัวอย่างมีชื่อเฉพาะเรียกว่าการ ***In***crement
  - กรณี `x = x - 1` เรียกว่าการ ***De***crement

Variable Names

- ความยาวไม่จำกัด
- ใช้ได้แค่ตัวอักษร ตัวเลข และเครื่องหมาย Underscore
- อักขระตัวแรกของชื่อ Variable ต้องเป็นตัวอักษรเท่านั้น (ควรใช้ตัวพิมพ์เล็ก)
- ชื่อ Variable (หรือ Identifier อื่น ๆ) นั้นมีความเป็น Case Sensitive กล่าวคือ `value` `Value` `vAlue` และ `valUE` ถือเป็น Variable คนละตัวกัน
- จะต้องไม่ซ้ำกับ Keyword ใน Python
- มาตรฐานการตั้งชื่อ Variable ใน Python ให้ใช้ตัวพิมพ์เล็กทั้งหมดและพิจารณาการใช้ Underscore คั่นระหว่างคำเพื่อทำให้อ่านง่ายขึ้น เช่น `max_score`

Aside

Python Keywords

```
False     None      True      and
as        assert    break     class
continue  def       del       elif
else      except    finally   for
from      global    if        import
in        is        lambda    nonlocal
not       or        pass      raise
return    try       while     with
yield
```

- เราสามารถแสดง list ของ keyword ได้โดยการใช้คำสั่ง

```
>>> import keyword
>>> keyword.kwlist
```

Variable Names [2]

- พิจารณาชุดคำสั่งด้านล่าง

```
a = 3.14159                pi = 3.14159
b = 11.2          VS       diameter = 11.2
c = a * b * b              area = pi * diameter * diameter
```

- ในมุมมองของ Python Interpreter ทั้งสองคำสั่งมีความหมายเหมือนกัน
- แต่ในสายตาผู้อ่าน ชุดคำสั่งทางด้านซ้าย ดูเหมือนทำงานได้เป็นปกติ
- ในขณะที่ชุดคำสั่งทางด้านขวา อาจมีข้อผิดพลาด
  - ชื่อตัวแปรควรเป็น `radius` แทนที่จะเป็น `diameter`?
  - หรือควรนำ `diameter` มาหารด้วย 2 ก่อนนำไปหาพื้นที่?
- การตั้งชื่อตัวแปรที่ดี ช่วยทำให้ Code เข้าใจง่ายและลดข้อผิดพลาด

หมายเหตุ ในการตั้งชื่อตัวแปรที่ใช้เก็บค่าที่ได้จากการวัดที่มีหน่วยต่าง ๆ กัน ควรมีการระบุหน่วยในชื่อตัวแปร เพื่อความชัดเจน เช่น `len_km`, `speed_mph`, `weight_lb`

Numeric and Boolean Operators

Category	Operators
Arithmetic	+, -, *, /, //, **, %, - (unary), + (unary)
Relational	<, <=, >=, >, ==, !=,
Bitwise	<<, >>, &, |, ^, ~
Assignment	+=, -=, *=, /=, //=, **=, %=, <<=, >>=, &=, |=, ^=
Logical	and, or, not

Note

- `/` is normal division — `3 / 2 == 1.5`
- `//` is *floored* division — `3 // 2 == 1`, `-3 // 2 == -2`
- `**` is power

Think Python: How to Think Like a Computer Scientist

Operator Precedence

- Operator ใน*ทางคณิตศาสตร์*ใน Python เป็นไปตามกฎการคำนวณปกติ (PEMDAS) โดยมีลำดับการดำเนินการดังนี้
  - Parentheses
  - Exponentiation
  - Multiplication and Division
  - Addition and Subtraction
- ในกรณีที่ operator อยู่ในลำดับเดียวกัน เช่น + และ – ให้ทำ Operation จากซ้ายไปขวา
- 2^{3^5} มีค่าเท่ากับเท่าไร

```
>>> 2**3**5 == (2**3)**5

>>> 2**3**5 == 2**(3**5)
```


Operator Precedence [2]

Operator	Description
(expressions...), [expressions...], {key: value...},{expressions...}	Binding or tuple display, list display, dictionary display, set display
x[index], x[index:index], x(arguments...), x.attribute	Subscription, slicing, call, attribute reference
**	Exponentiation — Mathematical Operators
+x, -x, ~x	Positive, negative, bitwise NOT
*, /, //, %	Multiplication, division, remainder
+, -	Addition and subtraction
<<, >>	Shifts
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
in, not in, is, is not, <, <=, >, >=, !=, ==	Comparisons, including membership tests and identity tests
not x	Boolean NOT
and	Boolean AND
or	Boolean OR
if – else	Conditional expression
lambda	Lambda expression

high → low

Basic String Operations

- โดยทั่วไป เราไม่สามารถใช้ Operation ในการคำนวณกับสายอักขระ (String) ได้

```
#all illegal
'2' - '1'     'eggs' / 'easy'       ' third' * 'a charm'
```

- แต่เครื่องหมาย + สามารถใช้ได้กับ String โดยจะเป็นการนำ String ทั้งสองมาต่อกัน (Concatenation)

```
>>> x = 'hello'
>>> y = 'world'
>>> x + y
'helloworld'
```

- ในทำนองเดียวกันเครื่องหมาย * สามารถใช้ได้โดยให้ผลคล้ายการบวกซ้ำ

```
>>> x * 3
'hellohellohello'
```


Comments

- นอกจากการตั้งชื่อ Variable ให้สื่อความหมายแล้ว เรายังสามารถทำให้ Code อ่านง่ายขึ้นโดยการเพิ่ม Comments ลงใน Code
- ใน Python เครื่องหมาย # ใช้แสดงจุดเริ่มของComment ในบรรทัดนั้น ๆ
  - Python Interpreter จะไม่อ่านตัวอักษรใด ๆ ที่อยู่หลังจาก # ในบรรทัดนั้น ๆ
- โดยปกติเราใช้ Comment เพื่ออธิบาย*วัตถุประสงค์*และรายละเอียดของ Code ในส่วนนั้น ๆ ของโปรแกรม

Comments [2]

- Comments are most useful when they document non-obvious features of the code - useful to explain why.
  - This comment is redundant with the code and useless:

```
v = 5       # assign 5 to v
```

  - This comment contains useful information that is not in the code:

```
v = 5       # velocity in meters/second.
```

ควรใส่ comment สำหรับแต่ละตัวแปร ว่ามีไว้เพื่อเก็บค่าอะไร มีหน่วยเป็นอะไร

Python Script Mode

- ที่ bash prompt สร้าง file เปล่า (คำสั่ง touch) แล้วเปิดไฟล์มา edit ด้วย IDLE

bash shell
```
$ touch hello.py
$ idle hello.py &
```

- Python script ควรมีการระบุบรรทัดแรกเป็น `#!/usr/bin/env python3` (ไม่ต้องพิมพ์เลขบรรทัด) เพื่อระบุว่าเป็น Script ของ Python 3

hello.py
```
01 #!/usr/bin/env python3
02
```

- จากนั้นให้แสดง String `'Hello World!!'` โดยใช้ฟังก์ชัน `print()`

hello.py
```
01 #!/usr/bin/env python3
02
03 print("Hello World!!")
```

Python Script Mode [2]

- Save แล้ว กด F5 ในหน้าต่าง IDLE เพื่อ run script

```
>>>
Hello World!!
>>>
```

- เราสามารถ run script จากหน้าต่าง bash shell ได้เช่นกัน

```
$ python hello.py
Hello World!!
```

- หรือเปลี่ยนประเภทไฟล์ให้เป็น executable ด้วยคำสั่ง `chmod +x` (ทำครั้งเดียว) แล้ว run ด้วยชื่อไฟล์ (ต้องใส่ path ในที่นี้คือ `./`)

```
$ chmod +x hello.py
$ ./hello.py
Hello World!!
```

Cygwin, Linux and OS X only

Python Script Mode [3]

```
>>> x = 'hello'
>>> y = 'world'
>>> x + y
'helloworld'
```

- ใน Interactive Mode เมื่อพิมพ์ Expression ใด ๆ ลงไป Python Shell จะแสดงค่าของ Expression นั้น ๆ

hello_world.py

```
08 x = 'hello'
09 y = 'world'
10 x + y
```

- แต่หากเรา Run Script ด้านบนจะพบว่าไม่มี Output ใด ๆ

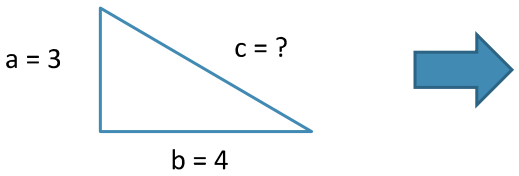
```
$ python hello_world.py
```

- ใน Script mode หากต้องการให้มีการแสดงผล Expression ใด ๆ เราจำเป็นต้องใช้ฟังก์ชัน `print()`

```
10 print(x + y)
```


Problem 1: Hypotenuse

- การหาด้านตรงข้ามมุมฉากของสามเหลี่ยม (Hypotenuse)



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

Practice 1: Hypotenuse [2]

- สร้างไฟล์ชื่อ `Lab02_1_5XXXXXXXX.py`

Note: กรณีต้องแสดงค่าหลายตัวแปร ทำได้โดยใช้ syntax

```
print("a = %.2f b = %.2f c = %.2f" % (a, b, c))
```

```
01 #!/usr/bin/env python3
02
03 # Compute the hypotenuse of a right triangle
04 a = 3
05 b =
06 c =
07 print("side a =", a)
08 print(                          )
09 print("hypotenuse c = %.2f" %c)
```

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

- เติมส่วนที่เหลือ (บรรทัดที่ 5, 6, และ 8) ให้ได้ output ตามที่ปรากฏด้านขวา

```
side a = 3
side b = 4
hypotenuse c = 5.00
```

The input() Function

- จะสังเกตได้ว่า โปรแกรมคำนวณด้านตรงข้ามมุมฉากที่เขียนขึ้น จะได้ผลเหมือนเดิมทุกครั้งที่ run
- จริง ๆ แล้วในการเขียนโปรแกรมโดยมาก เราจำเป็นจะต้องรับ Input หรือข้อมูลนำเข้าจาก User แทนการระบุค่าลงไปในโปรแกรม
- ใน Python สามารถทำได้โดยการใช้ฟังก์ชัน input()

```
>>> name = input("Hello, what is your name? ")
Hello, what is your name? Jon Snow
>>> print("Nice to meet you,", name)
Nice to meet you, Jon Snow
```

ฟังก์ชัน input() จะอ่านทีละบรรทัดจนพบ newline character '\n'

The input() Function [2]

```
01 #!/usr/bin/env Python3
02
03 x = input("Give me a number: ")
04 print("Half of that number is", x / 2)
```

```
 print("Half of that number is", x / 2)
TypeError: unsupported operand type(s) for /: 'str' and
'int'
```

- Output ที่ได้จาก ฟังก์ชัน input() (ในที่นี้คือ x) จะมีชนิดเป็น str เสมอ ดังนั้นหากต้องมีการคำนวณทางคณิตศาสตร์ ก็จำเป็นจะต้องเปลี่ยนชนิดของข้อมูลก่อนโดยการใช้ ฟังก์ชัน int() หรือ float()

```
03 x = float(input("Give me a number: "))
```


Practice 2: Fahrenheit to Celsius

$$\frac{C}{5} = \frac{F - 32}{9}$$

- สร้างไฟล์ชื่อ Lab02_2_5XXXXXXXX.py
- เขียนโปรแกรมเพื่อรับ input อุณหภูมิเป็นองศาฟาเรนไฮต์ และแปลงเป็นองศาเซลเซียส โดยให้มีผลการ run ดังแสดงด้านล่าง

```
Input temperature in Fahrenheit: 50
50.00 degree Fahrenheit is 10.00 degree Celsius
```

- การวิเคราะห์ปัญหา
  - Input: จำนวนข้อมูล_____ชนิดข้อมูล_____
  - Output: จำนวนข้อมูล_____ชนิดข้อมูล_____

Practice 3: Body Mass Index

- สร้างไฟล์ชื่อ Lab02_3_5XXXXXXXX.py
- เขียนโปรแกรมคำนวณดัชนีมวลกาย โดยศึกษาวิธีการคำนวณจาก http://th.wikipedia.org/wiki/ดัชนีมวลกาย โดยให้มีผลการ run ดังนี้

```
Input height (m): 1.735
Input weight (kg): 62.2
BMI is 20.6629
```

- การวิเคราะห์ปัญหา
  - Input: จำนวนข้อมูล_____ชนิดข้อมูล_____
  - Output: จำนวนข้อมูล_____ชนิดข้อมูล_____
- ทดสอบผลการคำนวณโดยการเปรียบเทียบกับ online BMI calculator เช่น http://www.nhs.uk/Tools/Pages/Healthyweightcalculator.aspx

References

- **https://docs. Python.org/3.4/reference/expressions.html**
- **https://docs. Python.org/3.4/tutorial/inputoutput.html**
- **https://docs. Python.org/3.4/library/stdtypes.html#old-string-formatting**
- **http://www.cs.cmu.edu/~112/notes/notes-data-and-exprs.html**
- **Miller, B., and Ranum, D. *Problem Solving with Algorithms and Data Structures Using Python*,**
- **Guttag, John V. *Introduction to Computation and Programming Using Python, Revised***