

การสอนเขียนโปรแกรมด้วยภาษาไพธอน

วิทยาการคำนวณ การออกแบบ และเทคโนโลยี

Special Lecture: Teaching Python & Tech

ชวณัฐ นาคะสันต์

ศูนย์สื่อสารสนเทศ มหาวิทยาลัยคานาซาวา

2021-01-18

หัวข้อการบรรยาย / Topics

ภาพรวม
Overview

ชุดคำสั่ง (โปรแกรม)
Programs

ภาษาไพธอน
Python

วิทยาการคอมพิวเตอร์
พื้นฐาน
Basic Computer Science

การสอนปฏิบัติการ
คอมพิวเตอร์
Teaching Computer Labs

หลังจากการบรรยายวันนี้ นักศึกษามีความรู้พื้นฐาน สามารถอธิบายเนื้อหาเกี่ยวกับวิทยาการคำนวณได้ และเขียนโปรแกรมง่ายๆ ได้ หากเทียบเป็นหลักสูตรวิศวกรรม/วิทยาการคอมพิวเตอร์ นักศึกษาจะมีความรู้เสมือนกำลังขึ้นชั้นปีที่สอง

เกี่ยวกับวิทยากร / ประวัติการศึกษาและการทำงาน



วศ.บ. (วิศวกรรมคอมพิวเตอร์)

เกียรตินิยมอันดับ 1

มหาวิทยาลัยเกษตรศาสตร์

Doctor of Engineering

(Information Science)

Nara Institute of Science and Technology
Japan

Assistant Professor

Information Media Center

Kanazawa University
Japan

เกี่ยวกับวิทยาการ / หัวข้อวิจัยและการทำงาน

Multipath Routing
(Computer Networks)
การค้นหาเส้นทางแบบหลาย
แขนงสำหรับระบบเครือข่าย

Data Science for Education
วิทยาการข้อมูลเพื่อการศึกษา

Computer Science
Education
การสอนวิทยาการคอมพิวเตอร์

Information Security
Management System
ระบบจัดการความมั่นคง
สารสนเทศ

Cybersecurity Policy
นโยบายความมั่นคงไซเบอร์

ก่อนจะเริ่ม

- การเขียนโปรแกรมไม่ใช่เรื่องไกลตัว
- การเขียนโปรแกรมไม่ได้ยาก
- การเขียนโปรแกรมไม่ใช่ทุกสิ่งของวิทยาการคำนวณ หรือวิทยาการคอมพิวเตอร์
- วันนี้จะไม่สอนเทคนิคหรือเนื้อหาใดๆ (Programming, IDE, ฯลฯ) เนื่องจากสามารถทำตามคู่มือการสอนได้อยู่แล้ว
- อย่างกังวลมากเรื่องคำศัพท์ บางครั้งเราอาจใช้คำต่างกัน หรือมีนิยามภาษาไทย/อังกฤษต่างกัน

ภาพรวม

Overview

เทียบเท่ารายวิชา: ปฐมนิเทศ

หลักการสอนและพัฒนาตนเองและผู้อื่นอย่างยั่งยืน

ฟังเฉยๆ

(Slide intentionally left blank.)

อย่าสับสนระหว่างศาสตร์กับเครื่องมือ

“กระป๋องอยู่ที่ใจ

หากเยี่ยมยุทธแล้วไซ้ร้ แค่กิ่งไผ่ก็ไร่เทียมทาน”

-- โกวเล้ง

อย่าสับสนระหว่างศาสตร์กับเครื่องมือ

ศาสตร์ (Science)

ตรรกศาสตร์

ชนิดข้อมูล

การแก้ปัญหา อัลกอริทึม

การเขียนโปรแกรม

ฐานข้อมูล

ทฤษฎีฐานข้อมูลเชิงสัมพันธ์ (Relational Theory)

ปัญญาประดิษฐ์

Machine Learning

เครื่องมือ (Tool) หรือ เทคนิค

ภาษาโปรแกรม เครื่องแปลภาษาโปรแกรม
(ภาษาไพธอน ภาษาซี ฯลฯ)

MySQL, MariaDB, MS Access, ฯลฯ

Regression, Neural Network, Deep Learning, ฯลฯ

ช่วยตัวเอง หาความรู้ด้วยตัวเอง

Python » English » 3.9.1 » Documentation »

Download
Download these documents

Docs by version
 Python 3.10 (in development)
 Python 3.9 (stable)
 Python 3.8 (stable)
 Python 3.7 (security-fixes)
 Python 3.6 (security-fixes)
 Python 3.5 (EOL)
 Python 2.7 (EOL)
 All versions

Other resources
 PEP Index
 Beginner's Guide
 Book List
 Audio/Visual Talks
 Python Developer's Guide

Python 3.9.1 documentation

Welcome! This is the documentation for Python 3.9.1.

Parts of the documentation:

What's new in Python 3.9?
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 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)

คู่มือการใช้ภาษาโปรแกรม

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

stackoverflow About Products For Teams Search...

Home PUBLIC Stack Overflow

Tags Users FIND A JOB Jobs Companies TEAMS What's this? Free 30 Day Trial

Implementing a stack using a linked list in python. Issues with questions about mutability

Asked 4 years ago Active 1 year, 4 months ago Viewed 3k times

I am trying to implement a stack using a linked list based off of just a node class. I am having some issues with the pop method of my class which doesn't seem to exhibit mutability. When I use the pop class method it returns the top of the stack correctly, but it fails to update the stack.

```

x=stack_linked(1)
x=x.insert(2)
x=x.insert(3)
x.print() # This is correct and prints 3,2,1
print(x.pop()) # This is correct and prints 3, but doesn't actually modify my list
x.print() # This prints 3,2,1
  
```

Why is self not mutable? Also how can I modify my class without completely blowing it up or creating a wrapper for it? Here is my class.

```

class stack_linked(object):
    def __init__(self,data):
        self.data=data
        self.next=None
  
```

กระดานถามตอบ

<https://stackoverflow.com/>

ผู้สอนต้องรู้มากขนาดไหน?

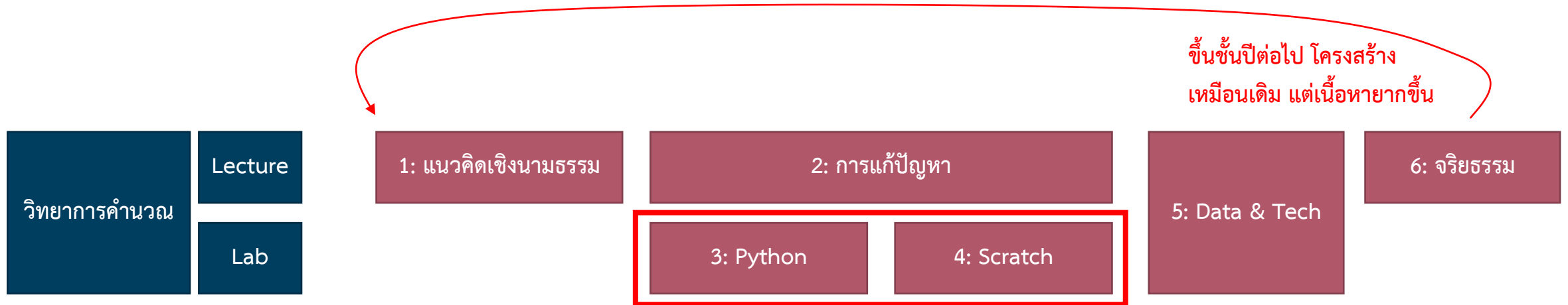
เรา^{รู้} 10

เราสอน 3

นักเรียน ได้ 1

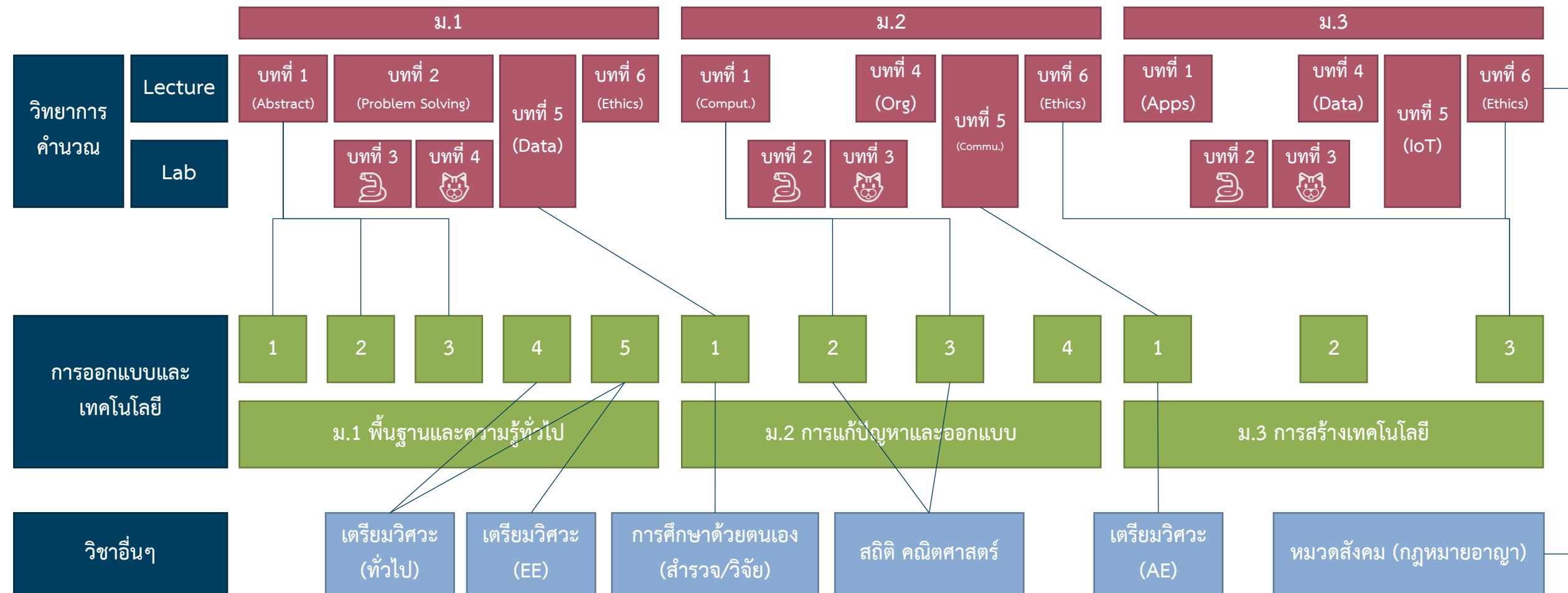
ผู้สอนต้องลุ่มลึกในศาสตร์ และมีความสามารถในการถ่ายทอด
ยิ่งถ้าจะเป็นอาจารย์ที่จะไปพัฒนาหรืออบรมครู (instructor-trainer)
ยิ่งต้องรู้ $10^2 = 100$ เท่าของหลักสูตร!

ผ่า! หลักสูตรเทคโนโลยี/วิทยาการคำนวณ สสวท. (ม.1)



หลักการของแต่ละชั้นปีเหมือนกัน แต่เนื้อหาจะยากขึ้น (ในส่วนของเทคนิค) และมองภาพกว้างมากขึ้น (ในส่วนของทฤษฎี)

ผ่า! หลักสูตรเทคโนโลยี/วิทยาการคำนวณ สสวท. (ภาพรวม)



Python และ Scratch สามารถใช้ประกอบกิจกรรมการเรียนรู้ได้ทุกหน่วยทุกวิชา

ชุดคำสั่ง (โปรแกรม) Programs

เทียบเท่ารายวิชา: CS 111 (Programming)

คอมพิวเตอร์ และภาษาโปรแกรมทั้งหมดในโลก มีหลักการเดียวกัน

```

sample.c + (D:/src) - GVIM1
File Edit Tools Syntax Buffers Window Help
#include<stdio.h>
#include<math.h>

typedef struct {
    int x,y;
} Point;

double distance (Point p1, Point p2){
    return sqrt(pow((p1.x - p2.x),2) +
               pow((p1.y - p2.y),2));
}

int main(){
    Point p1, p2;
    p1.x = 29;
    p1.y = 35;
    p2.x = 14;
    p2.y = 12;
    printf("%f", distance(p1, p2));
    return 0;
}

```

```

sample.s (D:/src) - GVIM3
File Edit Tools Syntax Buffers Window Help
# -mstackrealign -mveroupper

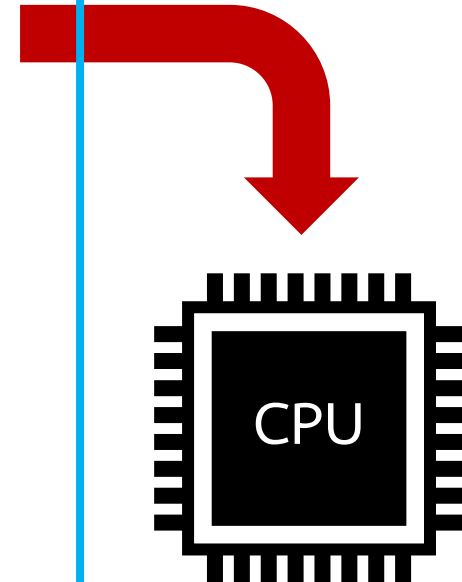
.text
.globl distance
.def distance; .scl 2; .type 32; .endef
.seh_proc distance
distance:
    pushq %rbp #
    .seh_pushreg %rbp
    movq %rsp, %rbp #,
    .seh_setframe %rbp, 0
    subq $48, %rsp #,
    .seh_stackalloc 48
    movaps %xmm6, -16(%rbp) #,
    .seh_savexmm %xmm6, 32
    .seh_endprologue
    movq %rcx, 16(%rbp) # p1, p1
    movq %rdx, 24(%rbp) # p2, p2
    # sample.c:9: return sqrt(pow((p1.x - p2.x),2) + pow((p1.y - p2.y),2);
    movl 16(%rbp), %edx # p1.x, _1
    # sample.c:9: return sqrt(pow((p1.x - p2.x),2) + pow((p1.y - p2.y),2);
    movl 24(%rbp), %eax # p2.x, _2
    # sample.c:9: return sqrt(pow((p1.x - p2.x),2) + pow((p1.y - p2.y),2);
    subl %eax, %edx # _2, _1
    movl %edx, %eax # _1, _3
    # sample.c:9: return sqrt(pow((p1.x - p2.x),2) + pow((p1.y - p2.y),2);
    cvtsi2sd %eax, %xmm0 # _3, _4
    movsd .LC0(%rip), %xmm1 #, tmp100

```

```

a.exe + (D:/src) - GVIM2
File Edit Tools Syntax Buffers Window Help
00000000: 4d5a 9000 0300 0000 0400 0000 ffff 0000 MZ.....*M
00000010: b800 0000 0000 0000 4000 0000 0000 0000 .....@.....*M
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 .....*M
00000030: 0000 0000 0000 0000 0000 0000 8000 0000 .....*M
00000040: 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468 .....!.!.!Th*M
00000050: 6973 2070 726f 6772 616d 2063 616e 6e6f is program canno*M
00000060: 7420 6265 2072 756e 2069 6e20 444f 5320 t be run in DOS *M
00000070: 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000 mode....$......*M
00000080: 5045 0000 6486 0f00 f2e7 0460 0076 0000 PE..d.....'.v..*M
00000090: f204 0000 f000 2700 0b02 021e 002a 0000 .....'.*.....*M
000000a0: 0046 0000 000a 0000 e014 0000 0010 0000 .F.....*M
000000b0: 0000 4000 0000 0000 0010 0000 0002 0000 ..E.....*M
000000c0: 0400 0000 0000 0000 0500 0200 0000 0000 .....*M
000000d0: 0030 0100 0004 0000 422b 0100 0300 0000 .0.....B+.....*M
000000e0: 0000 2000 0000 0000 0010 0000 0000 0000 .....*M
000000f0: 0000 1000 0000 0000 0010 0000 0000 0000 .....*M
00000100: 0000 0000 1000 0000 0000 0000 0000 0000 .....*M
00000110: 0090 0000 8c07 0000 0000 0000 0000 0000 .....*M
00000120: 0060 0000 b802 0000 0000 0000 0000 0000 .'.*.....*M
00000130: 0000 0000 0000 0000 0000 0000 0000 0000 .....*M
00000140: 0000 0000 0000 0000 0000 0000 0000 0000 .....*M
00000150: 4050 0000 2800 0000 0000 0000 0000 0000 @P..(.....*M
00000160: 0000 0000 0000 0000 dc91 0000 a001 0000 .....*M
00000170: 0000 0000 0000 0000 0000 0000 0000 0000 .....*M
00000180: 0000 0000 0000 0000 2e74 6578 7400 0000 .....text...*M
00000190: c828 0000 0010 0000 002a 0000 0004 0000 .(.(*.....*M
000001a0: 0000 0000 0000 0000 0000 0000 6000 5060 .....'.P'..*M
000001b0: 2e64 6174 6100 0000 d000 0000 0040 0000 .data.....@..*M

```



รหัสต้นฉบับ
Source Code

รหัสสัญลักษณ์
Symbolic Code
("Assembly Code")

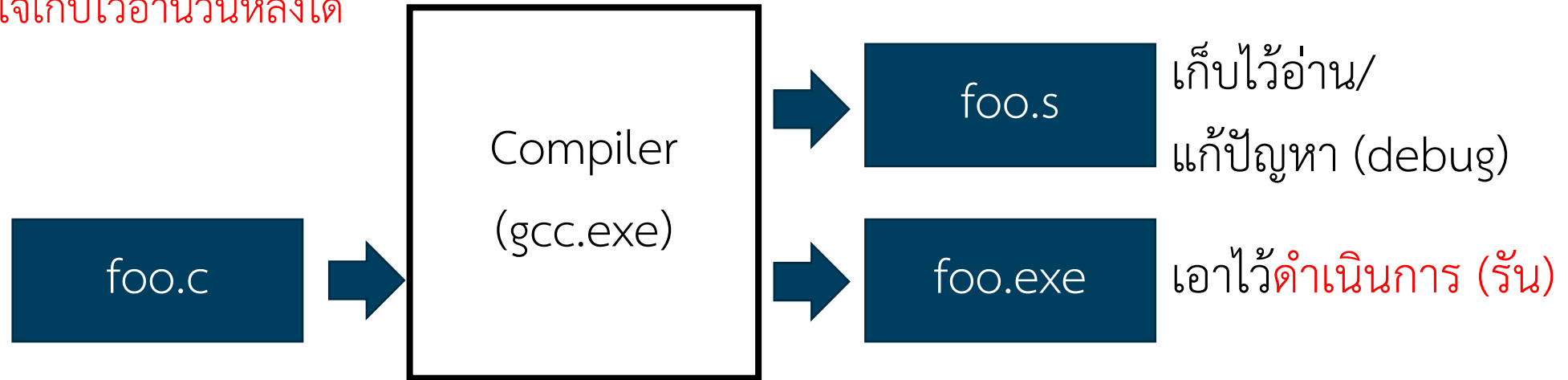
รหัสดำเนินการ
Executable Code ("EXE")

Assembly กับ EXE มีความหมายตรงกันทุกประการ (Assembly ออกแบบมาให้มนุษย์อ่านรู้เรื่อง)

การแปลรหัส (Compiling and Interpretation)

ซับซ้อนนิดนึง ถ้าไม่เข้าใจเก็บไว้อ่านวันหลังได้

ภาษาคอมไพล์
(Compiled Language)
เช่น ภาษาซี ภาษาเบสิก



ภาษาอินเทอร์พรีท
(Interpreted Language)
เช่น ภาษาไพธอน



ถ้าไม่เข้าใจ ลองศึกษารายวิชาอื่นเพิ่มเติม

- การเขียนโปรแกรม (Programming)
- ระบบปฏิบัติการ (Operating System)
- องค์ประกอบคอมพิวเตอร์ (Computer Architecture)

ภาษาไพธอน

Python

เทียบเท่ารายวิชา: CS 111 (Programming)

ภาษาไพธอนเขียนยังไง (stat.py) ← โหลดและรันเองได้จากคลังโค้ดที่จะให้ไว้

```
import statistics
```

เรียกใช้ชุดคำสั่งเสริม

```
data = [34.5, 18.4, 25.2, 40.5, 25.7, 28.8]  
label_mean = "Mean"  
label_stdev = "Standard Deviation"
```

กำหนดค่าตัวแปร

```
mean = statistics.mean(data)  
stdev = statistics.stdev(data)
```

คำนวณโดยใช้ฟังก์ชัน
เอาค่าที่ได้ใส่ตัวแปรใหม่

```
print(label_mean, mean)  
print(label_stdev, stdev)
```

แสดงผลข้อมูล

ตัดเกรด (grade.py)

`score = int(input())` ← รับค่าจากผู้ใช้งาน และแปลงเป็นจำนวนเต็ม

`if score >= 80:` ← ใช้ if ในการตรวจเงื่อนไขและแสดงเกรด

ย่อหน้าโค้ดด้วยการเว้นวรรค →

```
    print("A")
elif score >= 70:
    print("B")
elif score >= 60:
    print("C")
elif score >= 50:
    print("D")
else:
    print("F")
```

วิทยาการคอมพิวเตอร์พื้นฐาน

Basic Computer Science

เทียบเท่ารายวิชา: CS 101 (Intro. Computer Science), CS 221 (Data Types)

ชนิดข้อมูลพื้นฐาน (Primitive Data Types) และโครงสร้างข้อมูล (Structs)



ตัวอย่างเปิด (duck.py)

```
class Duck:  
    color = ""  
    weight = 0  
    def quack(self):  
        print("QUACK!")
```

```
D = Duck()
```

```
D.color = "white"  
D.weight = 20
```

```
print(D.color, D.weight)  
D.quack()
```

นิยามคลาส (ชนิดของวัตถุ) ว่าเปิดคืออะไร

สร้างเปิด ตั้งชื่อตัวแปร D

กำหนดค่าให้เปิด D

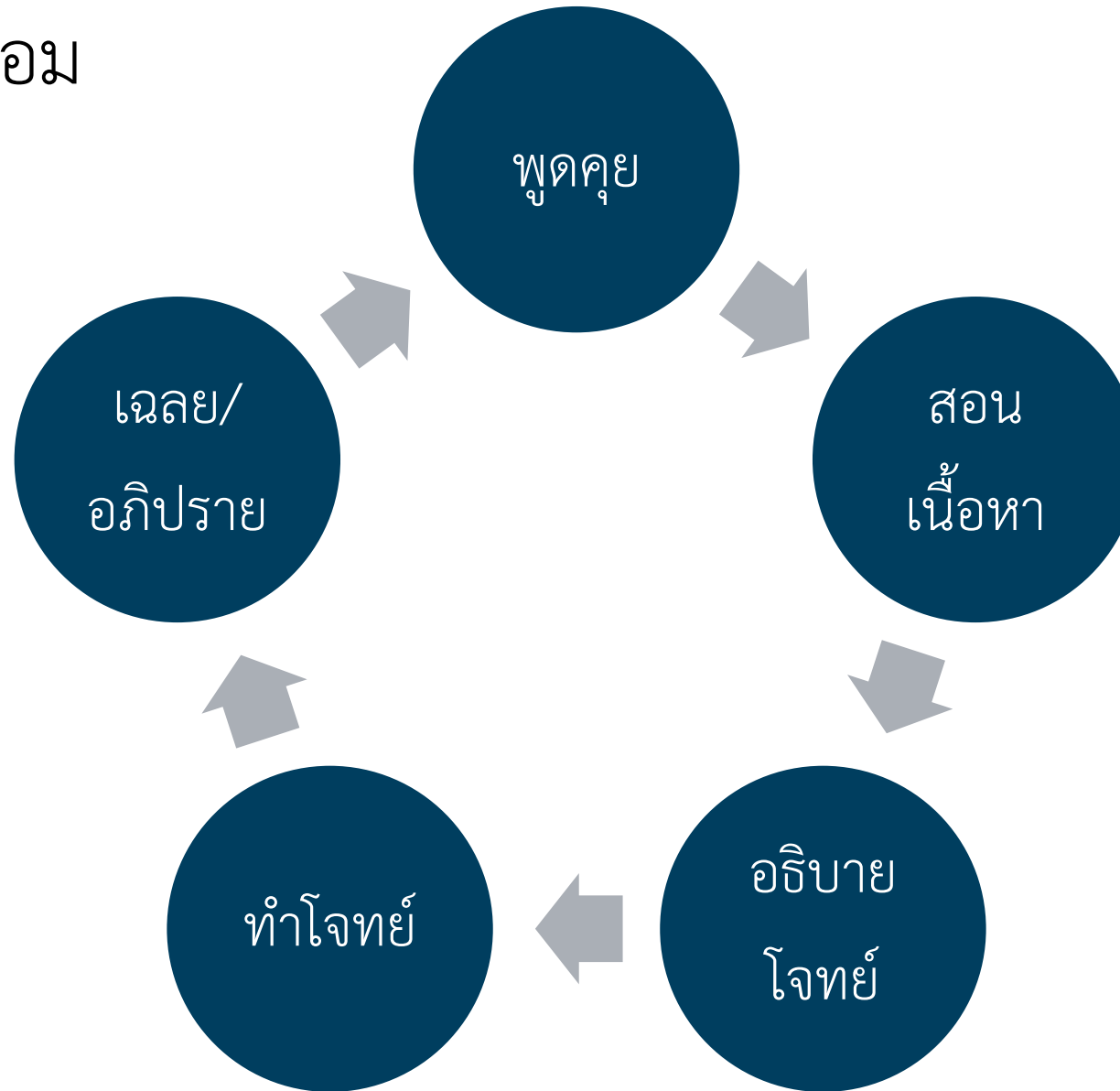
แสดงค่าจากเปิด D

เรียกใช้ฟังก์ชันของเปิด

การสอนปฏิบัติการคอมพิวเตอร์ Teaching Computer Labs

เทียบเท่ารายวิชา: ไม่มี วิศวะคอมไม่สอน แต่ถ้ากลับไปเป็น TA หรือได้ทำค่ายจะชั่งกับมันมาก

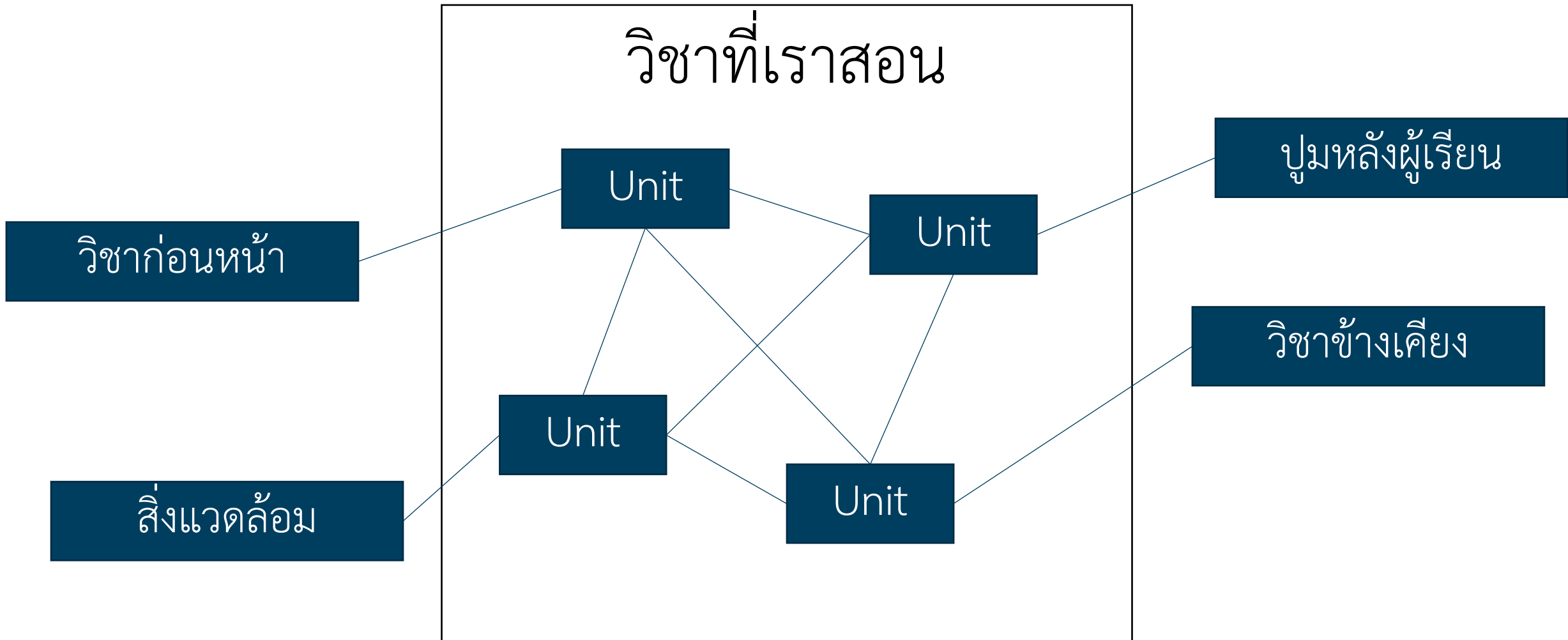
วงจรวิชาห้องคอม



ปัญหาที่มักเจอ

- นักเรียนพื้นฐานต่างกัน
- นักเรียนวอกแวก (distracted) ได้ง่าย
- นักเรียนช่วงชั้นที่สาม วัยกำลังเล่นได้ที
- การคุมชั้นเรียนคอมพิวเตอร์ (แล็บ) ไม่เหมือนการคุมชั้นเรียนในห้อง (เลคเชอร์)
- ครูมีความรู้ไม่พอ จึงไม่มั่นใจที่จะสอนนักเรียน
- ครูไม่รู้จะประเมินนักเรียนยังไง

สร้างความเชื่อมโยง



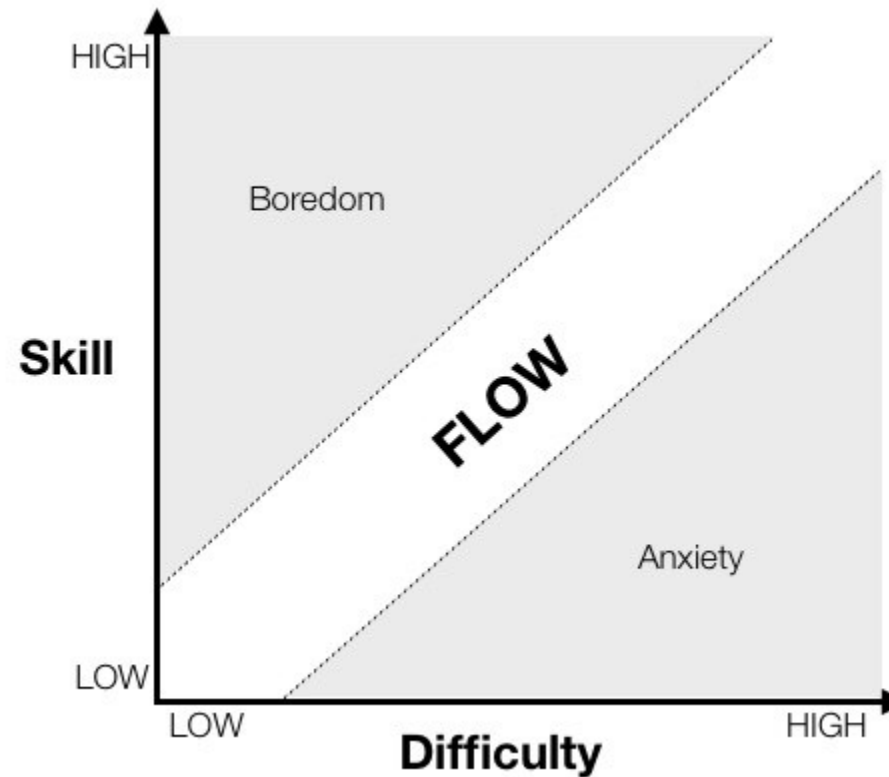
อย่าให้ยากไป อย่าให้ง่ายไป

Challenge

Encourage

Teach

Mentor



บทสรุป

ฟังเฉยๆ

(Slide intentionally left blank.)