หัวข้อเรียนเรื่อง Python Functions และ Module

- การนิยามฟังก์ชัน
- การสร้างและนำเข้าโมดูล
- การนิยามฟังก์ชันหลัก (__main__)

https://kmutt.me/skeic-python-2022

Functions

- “ฟังก์ชัน” ในภาษาไพธอน เป็นกลุ่มของโค้ดที่ถูกเรียกใช้ได้ สามารถส่งผ่านข้อมูล (หรือพารามิเตอร์) เข้าไปในฟังก์ชันได้ และสามารถส่งคืนข้อมูลกลับออกมานอกฟังก์ชันได้
- การนิยามฟังก์ชัน: ใช้ keyword **def** เช่น

  def my_function(): print(“Inside a function...”) # การนิยามฟังก์ชัน ไม่มีพารามิเตอร์

  def my_function(name): print(“Call me ” + name) # การนิยามฟังก์ชัน มีพารามิเตอร์
- การเรียกใช้ฟังก์ชัน: ใช้ชื่อฟังก์ชัน แล้วตามด้วยวงเล็บ ( ) เช่น

  my_function(“Ismael”)
- พารามิเตอร์ vs อาร์กิวเมนต์ (Argument):

  จากตัวอย่าง name คือ พารามิเตอร์ของฟังก์ชัน my_function ส่วน “Ismael” คือ อาร์กิวเมนต์ของฟังก์ชัน my_function
- โดยปกติ การเรียกใช้ฟังก์ชัน จำนวนอาร์กิวเมนต์ ต้องตรงกับพารามิเตอร์ ตามที่นิยามฟังก์ชันเอาไว้

  my_function(“Ismael”, “Kumar”) # ERROR

Arguments

- ใช้ * หน้าพารามิเตอร์ เมื่อไม่รู้จำนวนอาร์กิวเมนต์แน่นอนล่วงหน้า เช่น

```
def my_function(*schools):  print("Last school alphabetically... " + schools[3])

my_function(["AC", "BCC", "DS", "SK"])          # "Last school alphabetically... SK"
```

- การกำหนดค่าปริยายให้พารามิเตอร์

```
def my_function(school = "SK"):  print("I am from " + school)

my_function("BCC")                              # "I am from BCC"
my_function()                                   # "I am from SK"
```

การส่งคืนค่ากลับ

- ใช้ keyword **return** ในการส่งคืนค่ากลับออกนอกฟังก์ชัน

```
def my_function(x):  return x+x


my_function(4)                  # 8
my_function(12)                 # 24
```

- การเรียกคำสั่ง return จะหยุดการทำงานในฟังก์ชัน และส่งคืนค่ากลับออกนอกฟังก์ชันทันที

โมดูล

- เปรียบได้กับไลบรารีของโค้ด ใช้นามสกุลไฟล์ ***.py*** และสามารถนำเข้าเพื่อการใช้ซ้ำ
- ไฟล์โมดูลประกอบไปด้วยนิยามฟังก์ชัน และนิยามของตัวแปรต่าง ๆ
- โมดูลในภาษาไพธอนมีทั้งแบบ Built-in และพัฒนาขึ้นมาใหม่
  - ตัวอย่างของ Built-in Module ก็เช่น random

- ตัวอย่างการสร้างไฟล์โมดูล mymodule.py

```
# Inside mymodule.py
def my_function(name):  print("Hello, “ + name)
schools = ["AC", "BCC", "DS", "SK"]
```

การเรียกใช้โมดูล

- เรียกโดยใช้ **import**

```
import mymodule

mymodule.my_function('World')
print(mymodule.schools[3])          # "SK"
```

- เรียกโดยตั้งชื่อโมดูลใหม่โดยใช้ **import ... as**

```
import mymodule as mmd

school = mmd.schools[3]             # school = "SK"
```

- เรียกใช้เฉพาะบางนิยามโดยใช้ **from ... import**

```
from mymodule import schools

print(schools[3])                   # "SK"
```

- สังเกตว่าเมื่อใช้ from ... import จะไม่มีการอ้างถึงชื่อโมดูล เช่น

```
from mymodule import schools

print(mymodule.schools[3])          # ERROR!
```

ฟังก์ชันหลัก (__main__)

- ตัวอย่างการสร้างกำหนดฟังก์ชันหลักในภาษาไพธอน

```
def main():
    print("Hello World!")

if __name__ == "__main__":
    main()
```

In-class Exercise

- สร้างไฟล์ใหม่ 2 ไฟล์ **stats_module.py** และ **rand_stats_main.py**
- (**รอบที่ 1**) เขียนโค้ดใน **rand_stats_main.py**
  - นำเข้าโมดูล **random** และ **stats_module**
  - นิยามฟังก์ชัน **gen_int_list(start, stop, size)** และ **print_table(lst, num_col)**
  - นิยามฟังก์ชันหลัก เรียกคำสั่ง **randlist = gen_int_list(0, 9, 400)** และ **print_table(randlist, 40)**
  - Execute โปรแกรมประยุกต์ไพธอนที่เขียน
- (**รอบที่ 2**) เขียนโค้ดโมดูล **stats_module.py** โดยให้โมดูลนี้ประกอบไปด้วย 3 ฟังก์ชัน **mean(lst)**, **mode(lst)** และ **median(lst)**
  - ฟังก์ชันในโมดูลนี้ควรส่งค่าคืนกลับเฉพาะค่า mean, mode และ median เท่านั้น ไม่ควรเรียกคำสั่ง **print()**
  - แก้ไขโค้ดใน **rand_stats_main.py** ให้เพิ่มคำสั่ง ในการคำนวณและแสดงผลค่า mean, mode & median จาก **randlist**