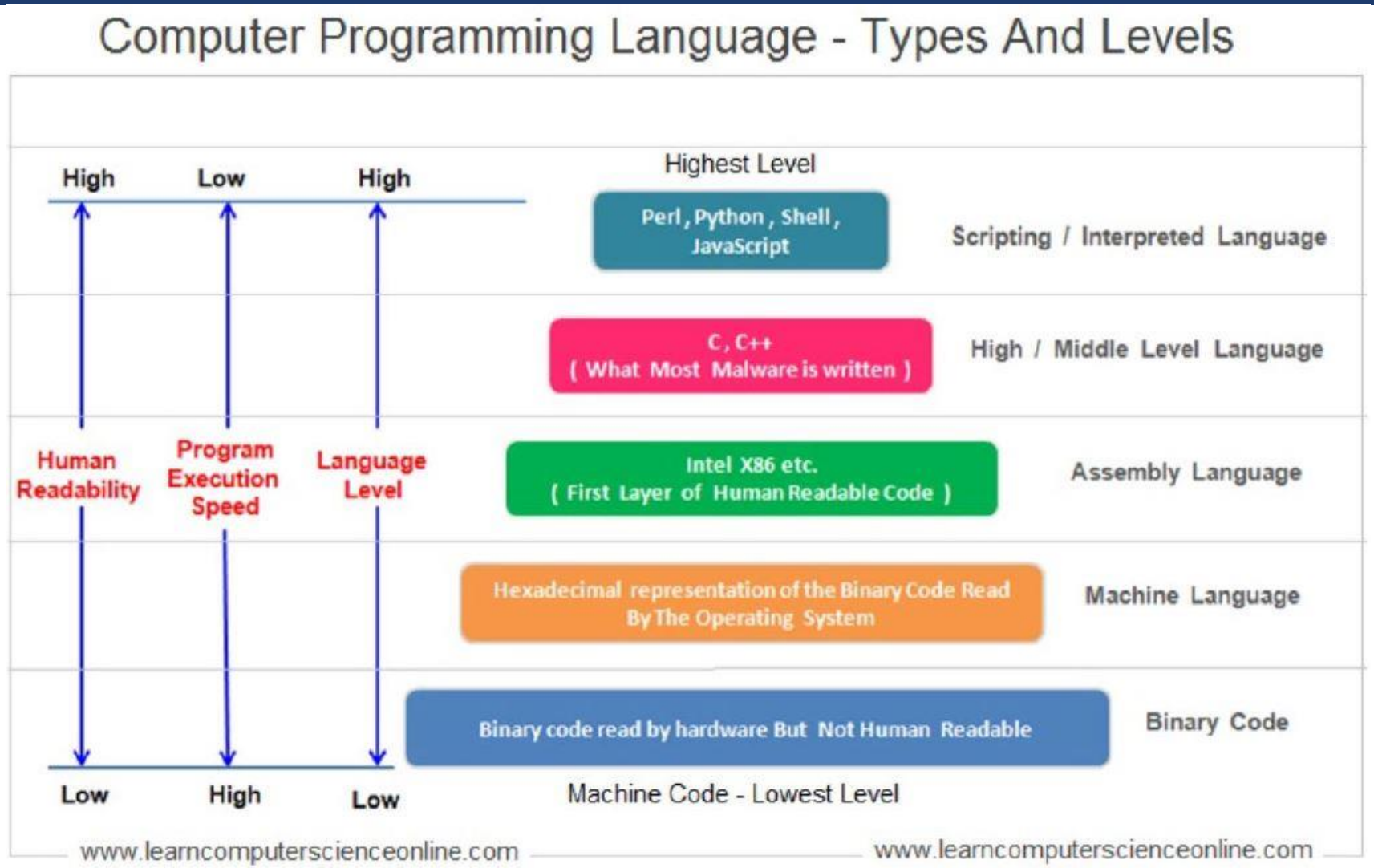
Data Engineering
Python Revision

Taught by Pichaya Tandayya

PRINCE OF SONGKLA UNIVERSITY

Python and Other Languages

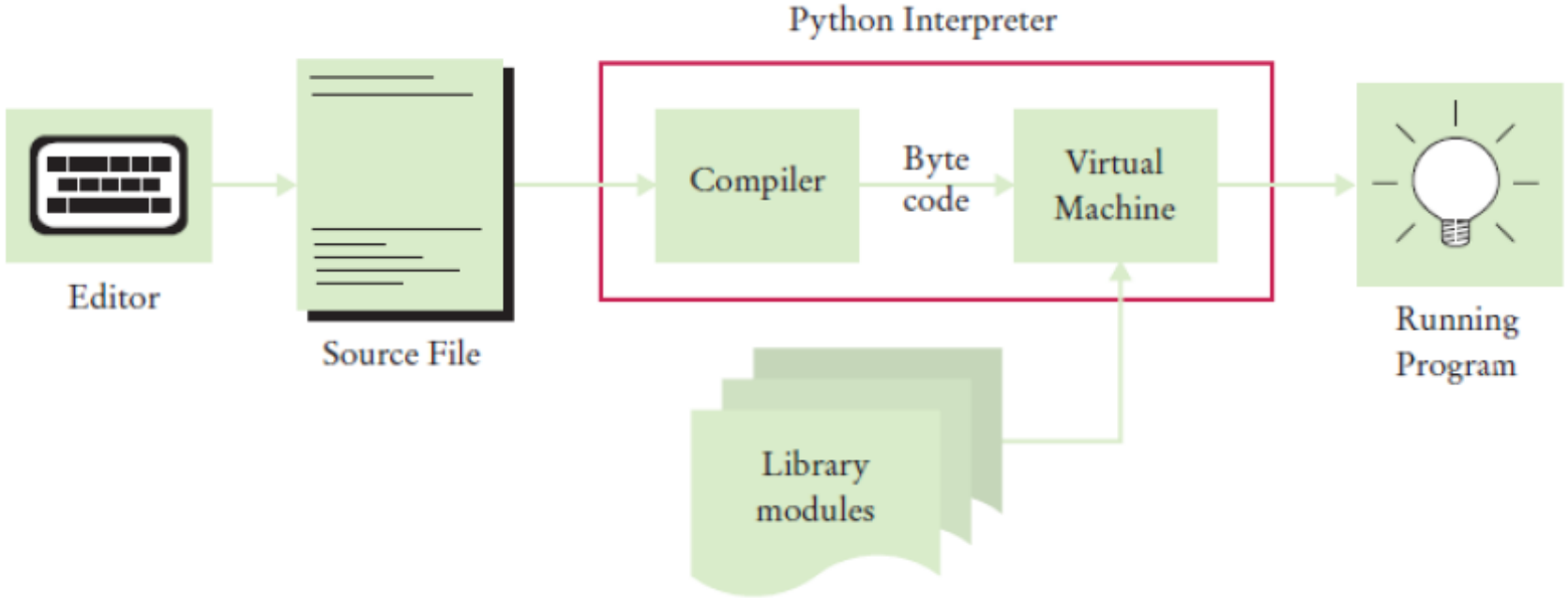


Python

- ภาษาโปรแกรมระดับสูง (high-level language)
- ได้รับความนิยมสูงในปัจจุบัน
- พัฒนาจากภาษาโปรแกรม C, C++, Java
- มี Module Libraries มากมาย ใช้งานได้หลากหลาย
- Open Source ใช้งานฟรี พัฒนาต่อยอดได้
- Multi-Programming Paradigms: Imperative, Structured, Procedural and OOP
- Current Version: Python 3

รูปแบบการเขียนโปรแกรมภาษาไพธอน

- การเขียนโปรแกรมรูปแบบปกติ คือ เขียนโปรแกรม ด้วย Editor สร้างไฟล์ Source Code (file.py) แล้วใช้ **Python Interpreter** แปลงเป็นโปรแกรมที่รันได้



- การเขียนโปรแกรมไพธอนรูปแบบโต้ตอบทันที (interactive mode) คือ การรันโค้ดบน ***Python Interpreter Shell***

Source: เสกสรรค์ สุวรรณมณี แนะนำการเขียนโปรแกรมคอมพิวเตอร์ ด้วยภาษาไพธอน

รูปแบบการเขียนโปรแกรมภาษาไพธอน

- **ฟังก์ชัน Function** คือ โครงสร้างการทำงานที่มีจุดเริ่มต้นและจุดสิ้นสุดในตัว มีลักษณะเป็น โปรแกรมย่อย (subprogram) อยู่ภายในโปรแกรม หรือถูกเรียกใช้งานจากโปรแกรมอื่น หรือส่วนอื่น ๆ มีชื่อเรียกหลายชื่อเช่น procedure, subprogram หรือ routine และฟังก์ชันที่เป็นส่วนประกอบในคลาส มีอีกชื่อเรียกหนึ่งว่า เมธอด (method)
- **โมดูล Module** คือ โปรแกรมไพธอน ที่รวบรวมฟังก์ชัน ข้อมูล และคำสั่งการทำงานต่าง ๆ ทุกโปรแกรมไพธอน (.py) เป็นโมดูล และแต่ละโมดูลสามารถเรียกใช้งานกันได้ โดยการ import หรือนำเข้าโมดูล เข้ามาในโปรแกรม

Source: เสกสรรค์ สุวรรณมณี แนะนำการเขียนโปรแกรมคอมพิวเตอร์ ด้วยภาษาไพธอน

Function Definition การนิยามฟังก์ชัน

- **function** ในโปรแกรมภาษาไพธอน มีการประกาศหรือนิยามก่อน จึงจะเรียกใช้งานได้ โดยใช้ คำหลัก (keyword หรือ reserved word) ของภาษาไพธอน คือ def (define หรือ definiton)

```
def ชื่อฟังก์ชัน ( พารามิเตอร์ ) :
    statement_1
    ....    # คำสั่งการทำงานในฟังก์ชัน เขียนย่อหน้า 1 ระดับ อยู่ในฟังก์ชัน
    statement_2
```

- ฟังก์ชันจะจบการทำงานที่คำสั่งสุดท้าย หรือ ที่คำสั่ง **return**
- คำสั่ง **return** เป็นการจบฟังก์ชันแบบไม่มีการส่งค่าคืนกลับ
- หรือ **return *value*** เป็นการจบฟังก์ชันแบบ ส่งค่าคืนกลับเป็นค่า ***value***

Source: เสกสรรค์ สุวรรณมณี แนะนำการเขียนโปรแกรมคอมพิวเตอร์ ด้วยภาษาไพธอน

ตัวอย่าง การนิยามฟังก์ชัน

```
def func1( ) :
    print(“Hello World”)
    return

def func2( p ) :
    print(“Hello ”*p, end=‘’)
    print(“World”)

def func3( a, b ) :
    c = (a*a + b*b)**0.5
    return c
```

- การตั้งชื่อฟังก์ชัน ใช้หลักการเดียวกับการตั้งชื่อตัวแปร และตั้งชื่อไม่ซ้ำกัน
- หลังชื่อฟังก์ชัน ตามด้วย ( ) เสมอ
- พารามิเตอร์ของฟังก์ชัน อยู่ภายใน ( ) ตั้งชื่อเหมือนตัวแปร อาจจะมีหลายตัว หรือไม่มีก็ได้
- คำสั่ง **return** เป็นจุดจบการทำงานของฟังก์ชัน (อาจมีหลายตำแหน่งได้) ถ้าไม่มีคำสั่ง return ฟังก์ชันทำงานจนจบคำสั่งสุดท้าย

ตัวอย่าง การนิยามฟังก์ชัน

ตัวอย่างการเรียกฟังก์ชัน

```
func1( )
func2(2)
x = 3
func2(x)
y = func3( 3.0, 4.0 )
print(y, func3(6.0, 8.0) )
```

ตัวอย่างผลลัพธ์ของโปรแกรม

```
Hello World
Hello Hello World
Hello Hello Hello World
5.0 10.0
```

- นิยามของฟังก์ชัน **ไม่มีการทำงานเกิดขึ้น ถ้าไม่ได้ถูกเรียกใช้**
- ***Function call*** คือการเรียกฟังก์ชัน หรือการเรียกให้ฟังก์ชันทำงาน
- เรียกฟังก์ชันด้วยชื่อและตามวงเล็บ และค่าในวงเล็บที่ส่งให้กับพารามิเตอร์ของฟังก์ชัน
- เมื่อฟังก์ชันทำงานจบ จะกลับมายังส่วนการเรียกใช้เสมอ อาจจะมีการส่งค่าคืนกลับมา (เช่น func3) หรือ ไม่มี (เช่น func1 func2)

Scope of Functions and Variables

- ไฟล์ Source Code โปรแกรมไพธอน (file.py) ประกอบด้วย ส่วนที่เป็นคำสั่งการทำงาน (script) ส่วนหลักระดับแรกสุด (ไม่ย่อหน้า) ซึ่งจะถูกเริ่มทำงานไปตามลำดับคำสั่ง
- หากมีการนิยามฟังก์ชัน ให้ใส่ไว้ในส่วนต้นโปรแกรม หรือก่อนส่วนคำสั่งการทำงาน (script) เพราะจะได้เรียกใช้งานฟังก์ชันที่ประกาศไว้ก่อนหน้าได้
- ฟังก์ชันจะถูกทำงานเมื่อมีการเรียกใช้เท่านั้น
- อาจมีการสร้างฟังก์ชัน main() ไว้เป็นส่วนหลักของโปรแกรม แต่ต้องมีคำสั่งเรียก ฟังก์ชัน main( ) ไว้เพื่อเรียกให้ทำงาน

Source: **เสกสรรค์ สุวรรณมณี** แนะนำการเขียนโปรแกรมคอมพิวเตอร์ ด้วยภาษาไพธอน

โครงสร้างไฟล์โปรแกรม Python (.py)

```
import xxx    # ส่วนการนำเข้าโมดูลต่างๆ จากภายนอก ที่ใช้ในโปรแกรมนี้

def f1(  ) :      # ประกาศ หรือนิยามของฟังก์ชัน
   …
def f2( x ) :     # ประกาศ หรือนิยามของฟังก์ชัน
   …
def main( ) :  # ประกาศฟังก์ชันสำหรับเป็นส่วนหลักของโปรแกรม (ไม่มีก็ได้)
   …
# ถ้าโปรแกรมมีสร้างฟังก์ชัน main ต้องเขียนคำสั่งการเรียกฟังก์ชัน main( ) ไว้ด้วย

…
a = 12
main()
…
```

Modules

- โปรแกรมไพธอน (filename.py) หนึ่ง ๆ เรียกว่าเป็นโมดูล
- โปรแกรมไพธอนสามารถนำเข้าโปรแกรมไพธอน อื่น ๆ ด้วยคำสั่ง

  import *ชื่อโมดูล* (ชื่อไฟล์เท่านั้น ไม่ต้องมี .py)
- ไฟล์โมดูลที่ถูกเรียกใช้ เก็บอยู่ในตำแหน่งเดียวกับกับไฟล์โปรแกรม (หรือมีการกำหนด path ไว้แล้ว)
- *โมดูลมาตรฐานต่าง ๆ หรือ Python Packages ที่ติดตั้งเพิ่มเติม ถูกเรียกใช้งานได้โดยไพธอน จะไปค้นหาไฟล์โมดูลในตำแหน่ง sys.path ที่กำหนดไว้แล้ว (หรือ system environment PYTHONPATH ของระบบ)*

Import การนำเข้า Modules และเรียกใช้ฟังก์ชัน

- โปรแกรมไพธอน (md.py) เป็นโมดูล มีฟังก์ชัน f1 และ f2

ไฟล์โมดูล md.py

```
# md.py
def f1(n) :
    x = n+100
    return x
def f2(x,y) :
    z = x+2*y
    return z
```

ไฟล์โมดูล md.py (ต่อ)

```
def main() :
    print('md')
    x=f1(2); y=f2(x,3)
    print(y,'\nEnd of md')

main()
```

Source: เสกสรรค์ สุวรรณมณี แนะนำการเขียนโปรแกรมคอมพิวเตอร์ ด้วยภาษาไพธอน

Import การนำเข้า Modules และเรียกใช้ฟังก์ชัน

- โปรแกรมไพธอน (caller.py) เรียกใช้ import โมดูล md และสามารถเรียกใช้ฟังก์ชัน md.f1( ) และ md.f2( ) ได้ มี *ชื่อโมดูล.* นำหน้าฟังก์ชัน

ไฟล์โมดูล `caller.py`

```
import md
def main() :
    print('caller')
    x= md.f1(2)
    y= md.f2(x,3)
    print('caller:', y)

main()
```

ผลการรันโปรแกรม `caller.py`

```
md
108
End of md
caller
caller: 108
# ส่วน main ของ md.py จะถูกรันก่อน
```

Import การนำเข้า Modules และเรียกใช้ฟังก์ชัน

ถ้าต้องการให้ส่วนหลักของโมดูล ***ไม่ถูกเรียกรัน*** เมื่อถูก import ไปยังโปรแกรมอื่น ๆ

ไฟล์โมดูล md.py

```
# md.py
def f1(n) :
    x = n+100
    return x
def f2(x, y) :
    z = x+2*y
    return z
```

ไฟล์โมดูล md.py (ต่อ)

```
def main() :
    print('md')
    x=f1(2); y=f2(x,3)
    print(y,'\nEnd of md')

if __name__ == "__main__" :
    main()
# ส่วนนี้จะเรียกรันเมื่อ md.py เป็นโปรแกรมหลักเท่านั้น
```

Source: เสกสรรค์ สุวรรณมณี แนะนำการเขียนโปรแกรมคอมพิวเตอร์ ด้วยภาษาไพธอน

Import การนำเข้า Modules และเรียกใช้ฟังก์ชัน

- โปรแกรมไพธอน (caller.py) เรียกใช้ import โมดูล md1 และสามารถเรียกใช้ฟังก์ชัน md.f1( ) และ md.f2( ) ได้ มี *ชื่อโมดูล.* นำหน้าฟังก์ชัน

ไฟล์โมดูล caller.py

```
import md
def main() :
    print('caller')
    x= md.f1(2)
    y= md.f2(x,3)
    print('caller:', y)

main()
```

ผลการรันโปรแกรม caller.py

```
caller
caller: 108


# ส่วน main ของ md.py จะไม่ถูกรัน
# ถ้ามีการใส่โค้ดข้างล่างนี้ ไว้ใน md.py
  if __name__ == "__main__" :
     main()
```

Convert Python Script to .exe File

- PyInstaller
  - https://python.land/deployment/pyinstaller
  - https://www.geeksforgeeks.org/python/convert-python-script-to-exe-file/

> pip install pyinstaller

> pyinstaller --onefile -w 'filename.py'

- Additional PyInstaller Options
  - --onefile: Bundles the script and all dependencies into a single executable.
  - --windowed: Suppresses the console window (useful for GUI applications).
  - --icon: Allows you to specify an icon for the executable.
- For more information, see https://medium.com/@moraneus/crafting-a-standalone-executable-with-pyinstaller-f9a99ea24432


PRINCE OF
SONGKLA
UNIVERSITY