# เอกสารประกอบการบรรยาย

# Data Science with Python

บริษัท แลนด์ แอนด์ เฮ้าส์ จำกัด (มหาชน) | Land & Houses

โดย ผู้ช่วยศาสตราจารย์ ดร.วสิศ ลิ้มประเสริฐ

# บทที่ 1: พื้นฐาน Python สำหรับงาน Data Science (Python Foundations)

**Main Story (บทที่ 1):** "ก่อนที่เราจะสร้างระบบอัตโนมัติ เราต้องเรียนรู้ 'ภาษา' ที่ใช้สั่งงานคอมพิวเตอร์ก่อน ในบทนี้ เราจะปูพื้นฐาน Python ทั้งหมด ตั้งแต่การคำนวณง่ายๆ (เทียบเคียง Excel) ไปจนถึงการเขียน Logic ที่ซับซ้อน เพื่อเตรียมพร้อมสำหรับการจัดการข้อมูลขนาดใหญ่"

## 1.1 การเตรียมความพร้อมและเครื่องมือ (Setup & Tools) (1.0 ชม.)

* **บรรยาย:** แนะนำคอร์ส, Main Story, และภาพรวมของ Data Science Workflow (Load -> Clean -> Transform -> Analyze -> Visualize -> Automate)
* **Demo:** การติดตั้ง Anaconda, VS Code (Extensions: Python, Jupyter), และการใช้งาน Jupyter Notebook เบื้องต้น (การรัน Cell, Markdown vs Code)

## 1.2 การเขียนโปรแกรม Python เบื้องต้น (Python Basics) (2.0 ชม.)

### 1.2.1 ตัวแปรและประเภทข้อมูล (Variables & Data Types)

* **บรรยาย:** int (เลขจำนวนเต็ม), float (เลขทศนิยม), str (ข้อความ), bool (True/False)
* **ตัวอย่าง (Contextual):**  
  # ข้อมูลจาก Case Study (Transaction file)  
  customer\_name = "Mr. A" # str  
  account\_no = "1234567890" # str (ถึงจะเป็นตัวเลข แต่เราไม่นำไปคำนวณ)  
  principal = 5000000.0 # float  
  dpd\_days = 35 # int  
  is\_overdue = True # bool  
  loan\_type = 'L' # str

### 1.2.2 โครงสร้างข้อมูลพื้นฐาน (Data Structures)

* **List (ลิสต์):** เก็บข้อมูลเป็น "ลำดับ" (แก้ไขได้, ใช้ [])
  + **ตัวอย่าง:**  
    # รายการ DPD ของลูกหนี้ 3 คน  
    dpd\_list = [0, 15, 92]  
    print(f"DPD ของลูกค้ารายที่สอง: {dpd\_list[1]}") # Access by index (เริ่มที่ 0) -> 15  
      
    # เพิ่มข้อมูล  
    dpd\_list.append(3)   
    print(f"ข้อมูลใหม่: {dpd\_list}") # -> [0, 15, 92, 3]  
      
    # รายการ Product Types  
    product\_types = ['PP', 'PL\_UN', 'CORPORATE']
* **Dict (Dictionary):** เก็บข้อมูลเป็นคู่ Key:Value (แก้ไขได้, ใช้ {})
  + **ตัวอย่าง:**  
    # ข้อมูลลูกหนี้ 1 คน (มีโครงสร้าง)  
    customer\_profile = {  
     'cif': 12345,   
     'name': 'Mr. A',   
     'dpd': 15,  
     'fprodt': 'PP'  
    }  
    print(f"DPD ของ Mr. A: {customer\_profile['dpd']}") # Access by key  
      
    # เพิ่ม/แก้ไข Key  
    customer\_profile['dpd'] = 20 # แก้ไขค่า  
    customer\_profile['principal'] = 5000000.0 # เพิ่ม key ใหม่  
    print(f"ข้อมูลใหม่: {customer\_profile}")
* **Tuple & Set:** (บรรยาย) ข้อแตกต่างและการใช้งาน (Tuple ใช้ () แก้ไขไม่ได้, Set ใช้ {} เก็บของไม่ซ้ำ)
* **Workshop 1.1 (Data Structures):**
  + สร้าง List ของยอดหนี้ [100000, 250000, 50000] และหาผลรวม (sum())
  + สร้าง Dict ข้อมูลลูกค้า 1 ราย (FCUSNO, FPRINCAM, FDPDUE01) จากข้อมูลตัวอย่าง

## 1.3 คำสั่งควบคุมและฟังก์ชัน (Control Flow & Functions) (2.0 ชม.)

### 1.3.1 คำสั่งควบคุม (Control Flow)

* **if/elif/else (การเลือกทำตามเงื่อนไข):**
  + **Main Story Link:** "เราจะใช้ Logic นี้ในการจัดกลุ่มลูกหนี้"
  + **ตัวอย่าง (Simple):**  
    dpd = 35 # Days Past Due  
    stage = ''  
    if dpd > 90:  
     stage = "3. NPL"  
    elif dpd > 30:  
     stage = "2. Under-performing"  
    else:  
     stage = "1. Performing"  
    print(f"DPD {dpd} days = Stage {stage}")
  + **ตัวอย่าง (Nested):**  
    principal = 6000000  
    if stage == "1. Performing":  
     if principal > 5000000:  
     print("High-Value Performing Customer")  
     else:  
     print("Standard Performing Customer")  
    else:  
     print(f"Non-Performing Customer (Stage {stage})")
* **for (การวนซ้ำ):**
  + **Main Story Link:** "การประมวลผลลูกหนี้ทีละคน"
  + **ตัวอย่าง (Simple List):**  
    loan\_amounts = [100, 200, 50, 80]  
    total\_loan = 0  
    for amount in loan\_amounts:  
     total\_loan = total\_loan + amount # หรือ total\_loan += amount  
    print(f"Total loan amount: {total\_loan}") # -> 430
  + **ตัวอย่าง (List of Dicts):**  
    # จาก Workshop 1.1  
    customer\_list = [  
     {'cif': 12345, 'name': 'Mr. A', 'dpd': 15},  
     {'cif': 12346, 'name': 'Ms. B', 'dpd': 91},  
     {'cif': 12347, 'name': 'Mr. C', 'dpd': 0}  
    ]  
      
    npl\_count = 0  
    for customer in customer\_list:  
     # (นำ Logic if/else มาใส่ใน loop)  
     if customer['dpd'] > 90:  
     print(f"{customer['name']} is NPL.")  
     npl\_count += 1  
     else:  
     print(f"{customer['name']} is Performing.")  
    print(f"Total NPL customers: {npl\_count}")

### 1.3.2 การเขียนฟังก์ชัน (Functions)

* **Main Story Link:** "สร้าง 'เครื่องมือ' ที่ใช้ซ้ำได้ ไม่ต้องเขียนโค้ดเดิมๆ"
* **ตัวอย่าง (Simple Function):**  
  def calculate\_interest(principal, rate\_percent):  
   """คำนวณดอกเบี้ยอย่างง่าย"""  
   interest = principal \* (rate\_percent / 100)  
   return interest  
    
  loan1\_interest = calculate\_interest(5000000, 5)  
  print(f"ดอกเบี้ย: {loan1\_interest}") # -> 250000.0
* **Workshop 1.2 (Function):**
  + สร้างฟังก์ชัน classify\_stage(dpd\_days) ที่รับค่า DPD และ return ค่า Stage (ข้อความ "1. Performing", "2. Under-performing", "3. NPL")
  + นำฟังก์ชันนี้ไปใช้ใน for loop จากข้อ 1.3.1 (ตัวอย่าง List of Dicts)

## 1.4 การประยุกต์ Python กับงาน (เทียบเคียง Excel) (1.0 ชม.)

### 1.4.1 สูตรคำนวณและข้อความ

* **Math Functions (เทียบเท่า SUM, MAX, MIN, COUNT, ROUND)**
  + sum(my\_list), max(my\_list), min(my\_list), len(my\_list)
  + round(123.456, 2) -> 123.46
* **String Functions (เทียบเท่า LEFT, RIGHT, MID, LOWER, UPPER, TRIM)**
  + **Slicing (LEFT, RIGHT, MID):**  
    account\_no = "1234567890"  
    branch\_code = account\_no[0:3] # LEFT(3) -> "123"  
    last\_4\_digits = account\_no[-4:] # RIGHT(4) -> "7890"  
    account\_class = account\_no[3:5] # MID(4, 2) -> "45"
  + **Methods (LOWER, UPPER, TRIM, FIND):**  
    customer\_name = " Mr. A "  
    print(customer\_name.strip()) # TRIM -> "Mr. A"  
    print(customer\_name.lower().strip()) # -> "mr. a"  
    print(account\_no.startswith("123")) # -> True

### 1.4.2 การจัดการ Error (Error Handling)

* **บรรยาย:** try...except
* **Main Story Link:** "จะเกิดอะไรขึ้นถ้าระบบอัตโนมัติของเราเจอข้อมูลผิดพลาด? ต้องดักจับ Error"
* **ตัวอย่าง (TypeError):**  
  data = {'dpd': 'N/A'} # ข้อมูล DPD เป็นข้อความ  
  try:  
   if data['dpd'] > 30: # บรรทัดนี้จะ Error เพราะ 'N/A' > 30 ไม่ได้  
   print("Overdue")  
  except TypeError:  
   print(f"Error: DPD data is invalid ({data['dpd']}). Setting to 0.")  
   data['dpd'] = 0
* **ตัวอย่าง (ZeroDivisionError):**  
  def calculate\_ratio(principal, installment):  
   try:  
   ratio = principal / installment  
   return ratio  
   except ZeroDivisionError:  
   print(f"Error: Installment is zero for principal {principal}. Returning None.")  
   return None # หรือ return 0  
    
  calculate\_ratio(500000, 10000) # ทำงานปกติ  
  calculate\_ratio(500000, 0) # เข้า except

## 1.5 Mini-Game: ทบทวนความรู้ (Review Game)

*(ส่วนนี้ยังคงเดิม)*

**ภารกิจ:** จับคู่ "สิ่งที่เราอยากทำ" (ข้อ 1-6) กับ "โค้ดที่ถูกต้อง" (ข้อ A-F)

**สิ่งที่เราอยากทำ (Concepts):**

1. เก็บข้อมูลลูกหนี้ 1 คน (cif: 12345, dpd: 30)
2. สร้าง "เครื่องมือ" ที่ใช้ซ้ำได้ เพื่อจัดกลุ่ม Stage
3. ประมวลผลลูกค้า "ทุกคน" ในลิสต์ ทีละคน
4. ป้องกันโปรแกรม "พัง" ถ้าเจอข้อมูล DPD เป็นตัวอักษร 'N/A'
5. ตัดสินใจว่า DPD 35 วัน ควรอยู่กลุ่มไหน (>90, >30, หรืออื่นๆ)
6. เก็บ "ลำดับ" ของ DPD ของลูกหนี้หลายคน [0, 15, 92]

โค้ดที่ถูกต้อง (Code Snippets):

A. for customer in customer\_list:

B. my\_list = [0, 15, 92]

C. if dpd > 90: ... elif dpd > 30: ... else: ...

D. my\_dict = {'cif': 12345, 'dpd': 30}

E. try: ... except TypeError: ...

F. def classify\_stage(dpd\_days): ...

*(เฉลย: 1=D, 2=F, 3=A, 4=E, 5=C, 6=B)*