# เอกสารประกอบการบรรยาย

# Data Science with Python

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

โดย ผู้ช่วยศาสตราจารย์ ดร.วสิศ ลิ้มประเสริฐ

# บทที่ 2: การจัดการและเตรียมข้อมูล (Data Manipulation & Preparation)

**Main Story (บทที่ 2):** "ในบทที่ 1 เราทำงานกับข้อมูล 'ทีละรายการ' (List, Dict) แต่ในโลกจริง ข้อมูลมาเป็น 'ล้านรายการ' ในไฟล์ CSV/Excel ในบทนี้ เราจะใช้ 'Pandas' ซึ่งเป็นเครื่องมือที่ทรงพลังที่สุดในการ 'โหลด', 'ทำความสะอาด', และ 'รวม' ข้อมูล Transaction และ Performance ทั้งหมดเข้าด้วยกัน"

## 2.1 รู้จักกับ Pandas (Introduction to Pandas) (1.0 ชม.)

* **บรรยาย:** DataFrame (ตาราง Excel) และ Series (คอลัมน์เดียว) คืออะไร?
* **Main Story Link:** "เปลี่ยนจาก List ของ Dict (ในบทที่ 1) มาเป็น DataFrame"
* **Demo (From Chapter 1):**  
  import pandas as pd  
    
  # ข้อมูลจากบทที่ 1  
  customer\_list = [  
   {'cif': 12345, 'name': 'Mr. A', 'dpd': 15},  
   {'cif': 12346, 'name': 'Ms. B', 'dpd': 91},  
   {'cif': 12347, 'name': 'Mr. C', 'dpd': 0}  
  ]  
    
  # สร้าง DataFrame จาก List of Dicts  
  df\_from\_list = pd.DataFrame(customer\_list)  
  print(df\_from\_list)  
  # นี่คือโครงสร้างหลักที่เราจะใช้ใน Pandas
* **Demo (Create Series):**  
  # Series (คอลัมน์เดียว)  
  s\_dpd = pd.Series([0, 15, 91, 3], name="DPD\_Days")  
  print(s\_dpd.mean()) # -> 27.25

## 2.2 การนำเข้าและสำรวจข้อมูล (Data Acquisition & Exploration) (1.5 ชม.)

### 2.2.1 การอ่านข้อมูล (Data Input)

* **Demo:** pd.read\_csv() และ pd.read\_excel()
* **Workshop 2.1 (Loading):**  
  # 1. โหลด Transaction Data  
  df\_tran = pd.read\_csv('Transection\_20240731.csv')  
    
  # 2. โหลด Performance Data  
  df\_perf = pd.read\_csv('Performance.csv')  
  # (ตัวอย่าง) ถ้าเป็น Excel: df\_perf = pd.read\_excel('Performance.xlsx', sheet\_name='Sheet1')  
    
  # 3. โหลด Data Dictionary (คำอธิบายฟิลด์)  
  df\_dict = pd.read\_csv('transaction FieldName.csv')

### 2.2.2 การสำรวจข้อมูล (Exploration)

* **Main Story Link:** "เราต้องเป็น 'นักสืบ' ก่อน ข้อมูลที่ได้มาเป็นอย่างไร?"
* **Demo (Basic):** head(), tail(), shape  
  print(df\_tran.shape) # (จำนวนแถว, จำนวนคอลัมน์)  
  print(df\_tran.head(3)) # ดู 3 แถวแรก
* **Demo (Deep Dive):** info(), describe()
* **Workshop 2.2 (Exploration):**
  + **df\_tran.info()**:
    - ดูว่ามีกี่แถว (Entries), กี่คอลัมน์
    - ดูว่าคอลัมน์ไหนมีค่าว่าง (Missing) บ้าง? (ดูที่ Non-Null Count) **เช่น FNPLFDTE จะมีค่าน้อยกว่าแถวทั้งหมด**
    - ดูประเภทข้อมูล (Dtype): FRPDATE เป็น int64 (ต้องแก้), FPRINCAM เป็น float64 (ถูกต้อง)
  + **df\_tran.describe()**:
    - ดูค่าสถิติของคอลัมน์ตัวเลข (FPRINCAM, FDPDUE01)
    - mean (ค่าเฉลี่ย), std (ส่วนเบี่ยงเบน)
    - min, 25%, 50% (median), 75%, max
    - **ข้อสังเกต:** FDPDUE01 มี max 98 วัน (สมมติ) แต่ min เป็น 0 หรือเปล่า? หรือเป็นค่าลบ (ถ้ามี)?
  + **df\_tran['FPRODTY'].value\_counts()**:
    - ดูว่ามี Product Type อะไรบ้าง อย่างละกี่บัญชี? (เช่น 'PP', 'PL\_UN')
    - (ขั้นสูง) df\_tran['FPRODTY'].value\_counts(normalize=True) (ดูเป็นสัดส่วน %)

## 2.3 การทำความสะอาดข้อมูล (Data Cleaning) (1.5 ชม.)

### 2.3.1 การจัดการข้อมูลที่หายไป (Missing Data)

* **Demo:** dropna() (ลบแถว) vs fillna() (เติมค่า)
* **Main Story Link:** "ค่า FDPDUE01 ที่เป็นค่าว่าง (NaN) หมายความว่าอะไร? (อาจจะ=0) เราต้องตัดสินใจทางธุรกิจและเติมค่า"
* **Workshop 2.3 (Cleaning - fillna):**  
  # สมมติ DPD ว่าง หมายถึงไม่ค้างชำระ  
  print(f"DPD missing count before: {df\_tran['FDPDUE01'].isnull().sum()}")  
  df\_tran['FDPDUE01'].fillna(0, inplace=True)  
  print(f"DPD missing count after: {df\_tran['FDPDUE01'].isnull().sum()}")  
    
  # FNPLFDTE (วันที่เป็น NPL) ว่าง หมายถึงไม่เป็น NPL (ยังไม่เติม)  
  # เราอาจจะเติม "Not NPL" หรือปล่อยว่างไว้ก็ได้

### 2.3.2 การแปลงประเภทข้อมูล (Type Conversion)

* **Demo:** astype(), pd.to\_datetime(), pd.to\_numeric()
* **Main Story Link:** "คอลัมน์วันที่ (FRPDATE, FNPLFDTE) ถูกอ่านเป็นตัวเลข/ข้อความ เราต้องแปลงมันให้เป็น 'วันที่' จริงๆ"
* **Workshop 2.4 (Type Conversion):**  
  # 1. แปลง Date (ที่เป็นตัวเลข 20240731)  
  df\_tran['FRPDATE'] = pd.to\_datetime(df\_tran['FRPDATE'], format='%Y%m%d')  
    
  # 2. แปลง FNPLFDTE (อาจมีค่า 0 หรือค่าที่แปลงไม่ได้)  
  # errors='coerce' จะเปลี่ยนค่าที่แปลงไม่ได้ (เช่น 0) ให้เป็น NaT (Not a Time)  
  df\_tran['FNPLFDTE'] = pd.to\_datetime(df\_tran['FNPLFDTE'], format='%Y%m%d', errors='coerce')  
    
  # 3. (สมมติ) ถ้า FPRINCAM ถูกอ่านเป็น string (เช่น '5,000,000')  
  # df\_tran['FPRINCAM'] = df\_tran['FPRINCAM'].str.replace(',', '')  
  # df\_tran['FPRINCAM'] = pd.to\_numeric(df\_tran['FPRINCAM'])

### 2.3.3 การลบข้อมูลที่ไม่จำเป็น (Duplicates & Dropping)

* **Demo:** drop\_duplicates(), drop()
* **Workshop 2.5 (Drop):**  
  # 1. ลบคอลัมน์ที่ไม่ใช้ออก (เช่น Watchlist Flag ทั้ง 30 คอลัมน์)  
  cols\_to\_drop = [f'FWLSTF{i:02d}' for i in range(1, 31)]   
  df\_tran.drop(columns=cols\_to\_drop, inplace=True)  
    
  # 2. ลบแถวที่ซ้ำซ้อน (เช็คทั้งแถว)  
  df\_tran.drop\_duplicates(inplace=True)  
    
  # 3. (ถ้าจำเป็น) ลบแถวที่ซ้ำซ้อนเฉพาะคอลัมน์หลัก (เช่น FACCNO)  
  # df\_tran.drop\_duplicates(subset=['FACCNO'], keep='first', inplace=True)

## 2.4 การรวมและจัดรูปข้อมูล (Data Transformation) (2.0 ชม.)

### 2.4.1 การเลือกและกรองข้อมูล (Selecting & Filtering)

* **Demo (Selecting):** การเลือกคอลัมน์ (Bracket [])  
  # เลือก 1 คอลัมน์ (ได้ Series)  
  s\_principal = df\_tran['FPRINCAM']  
    
  # เลือกหลายคอลัมน์ (ได้ DataFrame)  
  df\_subset = df\_tran[['FCUSNO', 'FACCNO', 'FPRINCAM', 'FDPDUE01']]
* **Demo (Filtering):** การกรองแถว (Boolean Masking)
* **Workshop 2.6 (Filtering):**  
  # 1. กรองแถวที่ DPD > 0 (ค้างชำระ)  
  df\_overdue = df\_tran[df\_tran['FDPDUE01'] > 0]  
    
  # 2. กรองแถวที่ DPD > 90 (NPL)  
  df\_npl = df\_tran.loc[df\_tran['FDPDUE01'] > 90]  
    
  # 3. กรองแบบซับซ้อน (และ &)  
  # กรองเฉพาะ FPRODTY == 'PP' และ FPRINCAM > 1,000,000  
  condition\_1 = df\_tran['FPRODTY'] == 'PP'  
  condition\_2 = df\_tran['FPRINCAM'] > 1000000  
  df\_high\_value\_pp = df\_tran[condition\_1 & condition\_2]  
    
  # 4. กรองแบบซับซ้อน (หรือ |)  
  # กรอง Product 'PP' หรือ 'PL\_UN'  
  df\_pp\_or\_pl = df\_tran[  
   (df\_tran['FPRODTY'] == 'PP') | (df\_tran['FPRODTY'] == 'PL\_UN')  
  ]  
  # (วิธีง่ายกว่า)  
  # df\_pp\_or\_pl = df\_tran[df\_tran['FPRODTY'].isin(['PP', 'PL\_UN'])]

### 2.4.2 การรวมข้อมูล (Combining DataFrames)

* **บรรยาย:** pd.concat() (การต่อกันแนวตั้ง - เช่น รวมไฟล์ 12 เดือน)  
  # (สมมติเรามี df\_tran\_jan และ df\_tran\_feb)  
  # df\_all\_months = pd.concat([df\_tran\_jan, df\_tran\_feb])
* **บรรยาย:** pd.merge() (การ Join - เทียบเท่า VLOOKUP/SQL JOIN)
* **Main Story Link:** "ข้อมูลสำคัญ (STAGE\_CIF) อยู่ในไฟล์ Performance เราจะ VLOOKUP ข้อมูลนี้มาใส่ใน Transaction"
* **Workshop 2.7 (Merging):**
  + **โจทย์:** รวม df\_tran (ซ้าย) กับ df\_perf (ขวา)
  + df\_merged = pd.merge(df\_tran, df\_perf, left\_on='FCUSNO', right\_on='CIF', how='left')
  + **ตรวจสอบ:** df\_merged.info() ดูว่ามี STAGE\_CIF เพิ่มเข้ามาหรือไม่?
  + **วิเคราะห์ผล:**
    - how='left': (Default) เอา df\_tran เป็นตัวตั้ง ถ้า FCUSNO ไหนไม่มีใน df\_perf ค่า STAGE\_CIF จะเป็น NaN
    - how='inner': เอาเฉพาะ FCUSNO ที่มีใน df\_tran **และ** df\_perf (ข้อมูลบางส่วนอาจหายไป)

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

*(ส่วนนี้ยังคงเดิม)*

**ภารกิจ:** จับคู่ "สิ่งที่เราอยากทำ" (ข้อ 1-6) กับ "โค้ด Pandas ที่ถูกต้อง" (ข้อ A-F)

**สิ่งที่เราอยากทำ (Tasks):**

1. โหลดไฟล์ Transection\_20240731.csv เข้ามาใน Python
2. รวมตาราง df\_tran กับ df\_perf โดยใช้ FCUSNO และ CIF เป็นคีย์ (เหมือน VLOOKUP)
3. ดูค่าสถิติเบื้องต้น (เช่น ค่าเฉลี่ย, max, min) ของคอลัมน์ที่เป็นตัวเลขทั้งหมด
4. เลือกมาเฉพาะแถวที่ FDPDUE01 (วันค้างชำระ) "มากกว่า 0"
5. เติม "0" ลงในช่องว่าง (NaN) ของคอลัมน์ FDPDUE01
6. แปลงคอลัมน์ FRPDATE ที่เป็นตัวเลข (เช่น 20240731) ให้เป็นข้อมูลประเภทวันที่

โค้ด Pandas (Code Snippets):

A. df\_tran[df\_tran['FDPDUE01'] > 0]

B. df\_tran['FDPDUE01'].fillna(0, inplace=True)

C. pd.merge(df\_tran, df\_perf, left\_on='FCUSNO', right\_on='CIF', how='left')

D. df\_tran.describe()

E. pd.to\_datetime(df\_tran['FRPDATE'], format='%Y%m%d')

F. pd.read\_csv('Transection\_20240731.csv')

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