



Python

เรียนรู้ Concept และฝึกพน Coding

Data Science

- ❖ เรียนรู้หลักการและวิธีการ Coding เกี่ยวกับ Data Science เป็นอย่างตั้งต่อไป
- ❖ แนะนำการ Coding กับ NumPy, Pandas, MongoDB, Matplotlib และ PandasGUI
- ❖ หากมีพื้นฐาน Python เป็นอย่างต้นมาแล้ว สามารถศึกษาด้วยตนเอง Step-by-Step ได้อย่างรวดเร็ว
- ❖ หมาย: สำหรับนักศึกษาและผู้สนใจเพิ่ม Skill การ Coding เกี่ยวกับ Data Science



ไฟล์ตัวอย่างภายในเล่ม
<https://serazu.com/>
9786164872394

ศุภชัย สมพาณิช

สารบัญ

บทที่ 1 เตรียมความพร้อมก่อนเข้าสู่โลกของข้อมูล.....	1
ภาษา Python กับงานด้านข้อมูล	2
การดาวน์โหลดและติดตั้ง Python SDK	2
การเขียนโค้ดภาษา Python ด้วย PyCharm	5
การดาวน์โหลดและติดตั้งโปรแกรม PyCharm	5
เริ่มต้นเขียนโค้ดภาษา Python ในโปรแกรม Pycharm	7
การรันภาษา Python ในโปรแกรม Pycharm	9
การเขียนโค้ดภาษา Python ด้วย Visual Studio Code	11
การดาวน์โหลดและติดตั้งโปรแกรม Visual Studio Code และส่วนขยายสำหรับภาษา Python	11
เริ่มต้นเขียนโค้ดภาษา Python ใน Visual Studio Code	13
การรันโค้ดภาษา Python ในโปรแกรม Visual Studio Code	14
การ Config โปรแกรม Visual Studio Code ให้สามารถใช้ Package ของ Python เพิ่มเติม	16
การเขียนโค้ดภาษา Python ในรูปแบบ Jupyter Notebook	18
บทที่ 2 พื้นฐานการทำงานกับข้อมูลด้วย NumPy	21
การดาวน์โหลดและติดตั้ง NumPy	21
การใช้งาน NumPy กับโครงสร้างข้อมูลแบบอาร์เรย์ (Array)	22
การตรวจสอบชนิดข้อมูลพื้นฐานในอาร์เรย์ของ NumPy	26
การหาค่าเฉลี่ยเลขคณิต, ค่าสูงสุด และค่าต่ำสุดของ NumPy	29
พื้นฐานการสุ่มตัวเลขทศนิยม	30
พื้นฐานการสุ่มตัวเลขจำนวนเต็มด้วยฟังก์ชัน randint()	31
พื้นฐานการสุ่มตัวเลขจำนวนเต็มด้วยฟังก์ชัน randrange()	31
การสุ่มตัวเลขจำนวนเต็มหลายจำนวน	33
พื้นฐานการใช้งานอาร์เรย์ 2 มิติ ของ NumPy	34
การหาผลรวมในอาร์เรย์ด้วยฟังก์ชัน sum()	35



การสร้างอาร์เรย์ด้วยฟังก์ชัน zeros() แบบมีข้อมูลเลข 0 เท่านั้น	38
การสร้างอาร์เรย์ด้วยฟังก์ชัน ones() แบบมีข้อมูลเลข 1 เท่านั้น	41
การเลือกข้อมูลแบบเป็นช่วงจากอาร์เรย์แบบ 1 มิติ ของ NumPy.....	44
การเลือกข้อมูลแบบเป็นช่วงจากอาร์เรย์แบบ 2 มิติ ของ NumPy.....	46
การเลือกข้อมูลจากอาร์เรย์แบบมีเงื่อนไขด้วยฟังก์ชัน where()	49
การรวมอาร์เรย์ด้วยฟังก์ชัน append()	51
บทที่ 3 พื้นฐานการทำงานกับข้อมูลด้วย Pandas	55
การดาวน์โหลดและติดตั้ง Pandas	55
การแสดงข้อมูลจากอาร์เรย์ 2 มิติ แบบตารางด้วยฟังก์ชัน DataFrame()	56
การถอดคอลัมน์หรือแถวออกจาก DataFrame ด้วยฟังก์ชัน drop()	57
การถอดค่าซ้ำออกจาก DataFrame ด้วยฟังก์ชัน drop_duplicates()	59
การเปลี่ยนชื่อคอลัมน์ใน DataFrame ด้วยฟังก์ชัน rename()	61
การสร้างฟังก์ชันทำงานกับข้อมูลโดยอาศัยฟังก์ชัน apply()	62
การสร้างแลมด้า (Lambda) ทำงานกับข้อมูลโดยอาศัยฟังก์ชัน apply()	64
การทำงานกับข้อมูลตามแกน X หรือแกน Y	65
การทำงานกับข้อมูลตามแนวคอลัมน์ด้วยฟังก์ชัน applymap()	66
โครงสร้างข้อมูลแบบ Series ของ Pandas	67
Series แบบกำหนดค่า Index เอง	70
การสร้าง Series จากโครงสร้างข้อมูล Dictionary (dict)	71
การอ่านข้อมูลจาก Series	72
การอ่านข้อมูลจาก Series แบบเป็นช่วงข้อมูล	75
บทที่ 4 DataFrame ของ Pandas	79
พื้นฐานการสร้าง DataFrame ของ Pandas	79
การสร้าง DataFrame แบบหลายคอลัมน์	81
การสร้าง DataFrame จากรายการ list แบบหลายคอลัมน์	83
การตรวจสอบข้อมูลเบื้องต้นของ DataFrame	86
การตรวจสอบชนิดข้อมูลของ DataFrame ด้วยคุณสมบัติ dtypes	89
การสร้าง DataFrame จาก Dictionary (dict)	92
การรวม DataFrame โดยอาศัยฟังก์ชัน append()	95
การรวม DataFrame ด้วยฟังก์ชัน concat()	98

พื้นฐานการอ่านข้อมูลจาก DataFrame ด้วยคุณสมบัติ loc	100
พื้นฐานการจัดกลุ่มข้อมูลใน DataFrame ด้วยฟังก์ชัน groupby()	105
บทที่ 5 การทำงานกับไฟล์ Excel	109
การอ่านข้อมูลจากไฟล์ Excel ด้วย xlrd.....	109
การแสดงรายชื่อคอลัมน์ที่ได้จากไฟล์ Excel.....	113
การเลือกข้อมูลเป็นช่วงจากไฟล์ Excel	114
การค้นหาข้อมูลด้วยฟังก์ชัน match() หรือฟังก์ชัน contains() ของ Pandas	118
การสร้างไฟล์ Excel จาก DataFrame ด้วย xlsxwriter.....	121
การ Export ข้อมูลจาก DataFrame ไปสู่โครงสร้างข้อมูลแบบอื่นๆ	123
การสร้างไฟล์ Excel และเพิ่มข้อมูลด้วย openpyxl	129
การใส่ข้อมูลแบบมีเงื่อนไข	132
บทที่ 6 ทำงานกับข้อมูลในไฟล์ CSV	135
พื้นฐานการอ่านข้อมูลจากไฟล์ CSV ด้วยฟังก์ชัน read_csv() ของ Pandas	135
การระบุเครื่องหมายอื่นๆ ในไฟล์ CSV.....	137
การเลือกบางคอลัมน์หรือบางແղວในไฟล์ CSV	138
การเปลี่ยนชนิดข้อมูลคอลัมน์	143
ชนิดข้อมูลพื้นฐานของไฟรอนกับ Pandas	146
การเปลี่ยนเป็นชนิดข้อมูลจัดกลุ่ม category ของ Pandas	146
การเลือกข้อมูลจากไฟล์ CSV ด้วยคุณสมบัติ iloc.....	148
บทที่ 7 ทำงานกับข้อมูลในฐานข้อมูล SQL Server	153
การดาวน์โหลดและติดตั้งฐานข้อมูล SQL Server	154
การสร้างตารางเก็บรายการสินค้า	160
การติดตั้ง ODBC เชื่อมต่อไฟรอนเข้ากับฐานข้อมูล SQL Server	165
การทำหนัดให้ฐานข้อมูล SQL Server เข้ารหัสภาษาไทย	168
การจัดการข้อมูล (CRUD) ในฐานข้อมูล SQL Server ด้วยไฟรอน	169
การอ่านข้อมูลจากฐานข้อมูล SQL Server	169
การเพิ่มข้อมูลใหม่ลงในฐานข้อมูล SQL Server	172
การแก้ไขข้อมูลในฐานข้อมูล SQL Server	174
การลบข้อมูลในฐานข้อมูล SQL Server (แบบตารางเดียว)	176



บทที่ 8 พื้นฐานการใช้งานระบบฐานข้อมูล MongoDB	179
ระบบฐานข้อมูล NoSQL แบบ MongoDB คืออะไร	179
การดาวน์โหลดและติดตั้งฐานข้อมูล MongoDB	180
พื้นฐานการเชื่อมต่อกับฐานข้อมูล MongoDB ด้วย MongoDB Compass	182
การสร้างฐานข้อมูล MongoDB ใหม่	183
การดาวน์โหลดและติดตั้ง pymongo	189
การจัดการข้อมูล (CRUD) ในฐานข้อมูล MongoDB	189
การเพิ่มข้อมูลใหม่เข้าไปในฐานข้อมูล MongoDB	190
การอัปเดตข้อมูลในฐานข้อมูล MongoDB	193
การลบข้อมูลในฐานข้อมูล MongoDB	195
บทที่ 9 การทำงานกับข้อมูลสูญหายและค่าผิดปกติ	197
บัญหาของข้อมูลสูญหาย	197
การถอดข้อมูลสูญหายออกด้วยฟังก์ชัน dropna()	198
การถอดค่าข้อมูลสูญหายแบบใช้ตัวแปรเดิม	199
การยินยอมให้มีค่าสูญหาย	200
การถอดค่าแบบกำหนดแกน X หรือแกน Y	201
การคำนวณปริมาณข้อมูลสูญหาย	202
การคำนวณปริมาณข้อมูลสูญหาย (หน่วยเบอร์เซ็นต์)	203
การตรวจสอบข้อมูลสูญหายด้วยฟังก์ชัน isnull()	204
พื้นฐานการเติมค่าที่สูญหายด้วยฟังก์ชัน interpolate()	205
การเติมค่าสูญหายด้วยฟังก์ชันfillna()	206
ทำความสะอาดกับข้อมูลที่มีค่าผิดปกติ (Outliers)	210
การคำนวณค่าเฉลี่ยเลขคณิตแบบตัดค่าสูญหายด้วยฟังก์ชัน nanmean()	212
บทที่ 10 การใช้งาน Pandas แบบมีส่วนแสดงผล (PandasGUI)	215
การดาวน์โหลดและติดตั้ง PandasGUI	216
พื้นฐานการแสดงข้อมูลจาก DataFrame ด้วย PandasGUI	216
การกรองข้อมูล (แท็บ Filters)	218
การ Import ข้อมูลจากไฟล์ CSV และ Excel	220
การ Export ข้อมูลออกเป็นไฟล์ CSV	220

บทที่ 11 ค่ากลางของข้อมูล (Measures of Central Tendency) 223

การหาค่าเฉลี่ยเลขคณิต (Arithmetic Mean) ด้วยฟังก์ชัน mean() ของ NumPy	223
การหาค่าเฉลี่ยเลขคณิตใน DataFrame ด้วยฟังก์ชัน mean() ของ Pandas.....	225
การหาค่าเฉลี่ยเลขคณิตแบบจัดกลุ่มข้อมูลใน DataFrame ของ Pandas	227
การหาค่าเฉลี่ยเลขคณิตกับตัวเลขศนิยมด้วยฟังก์ชัน fmean() ของ statistics.....	230
ค่าเฉลี่ยเรขาคณิต (Geometric Mean)	231
การหาค่าเฉลี่ยถ่วงน้ำหนัก (Weight Mean) ด้วยฟังก์ชัน average() ของ NumPy.....	234
การหาค่าเฉลี่ยถ่วงน้ำหนักใน DataFrame ของ Pandas.....	236
การหาค่าเฉลี่ย-armonic (Harmonic Mean).....	238
การหาค่ามัธยฐาน (Median).....	239
ฐานนิยม (Mode).....	242
การคำนวณค่ามัธยฐานแบบตัดค่าสูญหายด้วยฟังก์ชัน nanmedian()	243
ฐานนิยมแบบหลายค่าด้วยฟังก์ชัน multimode() ของ statistics.....	244

บทที่ 12 การหาตำแหน่งของข้อมูล (Measures of Position) 247

การหาตำแหน่งแบบควอร์ไทล์ (Quartile).....	247
การหาตำแหน่งแบบเดไซล์ (Decile)	251
การหาตำแหน่งแบบเปอร์เซ็นไทล์ (Percentile).....	253
วิธีการคำนวณค่าผิดปกติ Outliers	256

บทที่ 13 การวัดการกระจายของข้อมูล (Measures of Dispersion) 261

เป้าหมายของการวัดการกระจายของข้อมูล.....	261
การวัดการกระจายแบบสัมบูรณ์ (Absolute Variation).....	262
พิสัย (Range).....	263
ส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation)	264
ส่วนเบี่ยงเบนเฉลี่ย (Mean Deviation)	266
ส่วนเบี่ยงเบนควอร์ไทล์ (Quartile Deviation).....	270
การวัดการกระจายสัมพัทธ์ (Relative Variation)	273
สัมประสิทธิ์ของพิสัย (Coefficient of Range).....	273
สัมประสิทธิ์ของการแปรผัน (Coefficient of Variation).....	275
สัมประสิทธิ์ของส่วนเบี่ยงเบนเฉลี่ย (Coefficient of Average Deviation)	280



สัมประสิทธิ์ของส่วนเบี่ยงเบนควอตีไทล์ (Coefficient of Quartile Deviation).....	283
การคำนวณค่าทางสถิติเบื้องต้นด้วยพังก์ชัน describe() ของ Pandas	286
การแสดงค่าทางสถิติเบื้องต้นของ PandasGUI.....	288
บทที่ 14 พื้นฐานการแสดงข้อมูลแบบกราฟด้วย Matplotlib.....	291
การดาวน์โหลดและติดตั้ง Matplotlib.....	291
พื้นฐานการสร้างกราฟจุด (Scatter Plot).....	292
การสร้างกราฟจุดจากข้อมูลหลายชุด	294
การทำหนวดขนาดจุด	297
การทำหนดความเข้มของจุดข้อมูล.....	298
การสร้างกราฟแท่ง (Bar Graph)	299
การสร้างกราฟวงกลม (Pie Graph)	302
การแบ่งสัดส่วนข้อมูลออกจากส่วนของกราฟวงกลม.....	303
พื้นฐานการสร้างกราฟเส้น (Line Graph).....	304
การทำหนดข้อมูล 1 แกน.....	308
การสร้างกราฟเส้นและ Mark จุดข้อมูล.....	309
การปรับแต่งกราฟเป็นเส้นประ	310
การปรับแต่งรูปแบบกราฟเส้นด้วย linestyle.....	313
กราฟเส้นกับข้อมูลหลายชุด.....	315
การแยกกราฟเส้นจากข้อมูลแต่ละชุด	316
การสร้างกราฟ BoxPlot	319
บทที่ 15 การสร้างกราฟของ PandasGUI.....	325
การสร้างกราฟแท่ง (Bar Graph)	326
การสร้างกราฟเส้น (Line Graph).....	329
การสร้างกราฟกระจายของข้อมูล หรือกราฟจุด (Scatter Graph)	330
การสร้างกราฟวงกลม (Pie Graph)	331

การเลือกข้อมูลจากอาร์เรย์แบบมีเงื่อนไขด้วยฟังก์ชัน where()

โครงสร้างข้อมูลแบบอาร์เรย์ของ NumPy มีความสามารถเลือกดูข้อมูลแบบมีเงื่อนไขติดตัวอยู่แล้ว เช่น มีข้อมูลเป็นตัวเลขจำนวนเต็ม 5 ตัว เก็บอยู่ในตัวแปร arr หากกำหนดเงื่อนไขว่า ต้องการข้อมูลที่มีค่ามากกว่าหรือเท่ากับ 30 ผลที่ได้คือ 30, 40 และ 50 มีโค้ดดังต่อไปนี้

Python

```
import numpy as np

arr = np.array([10, 20, 30, 40, 50])

result = arr[arr>=30]
print('ข้อมูล : ', result)
```

```
UsingNumpy.py <
UsingNumpy.py > ...
1 import numpy as np
2
3 arr = np.array([10, 20, 30, 40, 50])
4
5 result = arr[arr>=30]
6 print('ข้อมูล : ', result)
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Projects> cd 'c:\Projects'; & 'C:\Program Files (x86)\Python39\python' 'c:\Projects\UsingNumpy.py'
ข้อมูล : [30 40 50]
PS C:\Projects>
```

รูปที่ 2-33 กรณีมีข้อมูลตรงกับเงื่อนไข

ต่อมาเปลี่ยนเงื่อนไขใหม่ ต้องการข้อมูลที่มีค่ามากกว่าหรือเท่ากับ 100 ถือเป็นเงื่อนไขที่ไม่ตรงกับข้อมูลที่มีอยู่ ผลที่ได้คือ เป็นค่าว่าง ดังรูปที่ 2-34

Python

```
result = arr[arr>=100]
```



การรวม DataFrame ด้วยฟังก์ชัน concat()

ฟังก์ชัน **concat()** ทำหน้าที่เชื่อมต่อโครงสร้างข้อมูล DataFrame เข้าด้วยกัน การรวมข้อมูลด้วยวิธีนี้มีข้อดีตรงที่ผู้อ่านสามารถกำหนดค่า Index ได้ด้วยตัวเอง

ตัวอย่างที่ 4-8 การรวม DataFrame ด้วยฟังก์ชัน concat()

มีข้อมูลเก็บอยู่ในโครงสร้าง DataFrame ที่แตกต่างกันอยู่ 2 ชุด เก็บอยู่ในตัวแปร data1 กับ data2 ตามลำดับ หากต้องการรวมข้อมูลเข้าด้วยกัน มีโค้ดดังต่อไปนี้

โค้ดไฟรอนที่ 4-8 การรวม DataFrame ด้วยฟังก์ชัน concat() (DataFrameConcat.ipynb)

```
import pandas as pd

data1 = [
    ['สมชาย', 185, 16000], ['วีระชัย', 175, 17500],
    ['พรชัย', 182, 20000], ['วีระ', 183, 19900]
]
data2 = [
    ['ดวงใจ', 165, 17800, 'กรุงเทพ'], ['วีไล', 163, 19300, 'เชียงใหม่']
]

colsname1 = ['ชื่อ', 'ส่วนสูง', 'เงินเดือน']
colsname2 = ['ชื่อ', 'ส่วนสูง', 'เงินเดือน', 'ที่อยู่']

df1 = pd.DataFrame(data1, columns=colsname1, index=[3, 4, 5, 6])
df2 = pd.DataFrame(data2, columns=colsname2, index=[1, 2])

df = pd.concat([df1, df2])
df
```

	ชื่อ	ส่วนสูง	เงินเดือน	ที่อยู่
3	สมชาย	185	16000	NaN
4	วีระชัย	175	17500	NaN
5	พรชัย	182	20000	NaN
6	วีระ	183	19900	NaN
1	ดวงใจ	165	17800	กรุงเทพ
2	วีไล	163	19300	เชียงใหม่

รูปที่ 4-21 แสดงโครงสร้างข้อมูลจาก DataFrame 2 ชุด

จากรูปที่ 4-21 พบร่วมกับโครงสร้างข้อมูลที่แตกต่างกัน ถูกเติมด้วยข้อมูลสูญหาย NaN



การเลือกข้อมูลเป็นช่วงจากไฟล์ Excel

การเรียกดูข้อมูลจากไฟล์ Excel ผู้เขียนเลือกเขียนໂດดแบบ Jupyter Notebook เพื่อให้ส่วนแสดงผลชัดเจนยิ่งขึ้นดังนี้

- การระบุช่วงแຄวที่ต้องการ เช่น ต้องการข้อมูลแถวที่ 1 ถึง 3 ผู้อ่านต้องระบุช่วง [0:3]

Python – Jupyter Notebook

```
import pandas as pd

df = pd.read_excel('Product.xlsx')
df[0:3]
```

The screenshot shows a Jupyter Notebook interface with a single cell containing Python code. The code imports pandas, reads an Excel file named 'Product.xlsx', and selects rows 0 to 3. The resulting DataFrame is displayed as a table:

	รหัสสินค้า	ชื่อสินค้า	ต้นทุน	ราคาขาย
0	A0001	การใช้งานภาษา C#	290	325
1	A0002	พัฒนาเว็บแอปด้วย ASP.NET Core MVC	270	299
2	A0003	การใช้งานฐานข้อมูล SQL Server 2019	300	359

รูปที่ 5-8 แสดงข้อมูล 3 แถว

- การระบุคอลัมน์และระบุແຄວ ทำได้โดยการระบุชื่อคอลัมน์และระบุช่วงແຄວที่ต้องการ เช่น ต้องการข้อมูลคอลัมน์รหัสสินค้าและชื่อสินค้า จำนวน 3 ແຄວແຮກ มีໄດ້ดังຕ่อປັ້ນ

Python – Jupyter Notebook

```
import pandas as pd

df = pd.read_excel('Product.xlsx')
df[['รหัสสินค้า', 'ชื่อสินค้า']][0:3]
```



บทที่
6

ทำงานกับข้อมูลในไฟล์ CSV

ไฟล์ CSV มีนามสกุลไฟล์ .csv ถือเป็นแหล่งข้อมูลอิเล็กทรอนิกส์ประเภทหนึ่งที่มีความนิยมสูงมาก ข้อมูลที่อยู่ในไฟล์ CSV มีรูปแบบที่เป็นมาตรฐานชัดเจน ส่งผลให้ได้ข้อมูลที่มีคุณภาพสูงอีกรูปแบบหนึ่ง

พื้นฐานการอ่านข้อมูลจากไฟล์ CSV ด้วยฟังก์ชัน `read_csv()` ของ Pandas

ไฟล์ CSV โดยปกติแล้วใช้เครื่องหมาย , คั่นข้อมูลแต่ละส่วน ส่งผลให้เกิดการแบ่งส่วนของข้อมูลอย่างชัดเจน จึงทำให้ข้อมูลมีรูปแบบแน่นอนเช่นเดียวกับไฟล์ Excel

การทำงานกับไฟล์ประเทก CSV ด้วยภาษา Python ผู้อ่านสามารถใช้ฟังก์ชันที่ชื่อว่า `read_csv()` ของ Pandas เข้ามาทำหน้าที่อ่านข้อมูลที่ได้จากไฟล์ CSV

โปรแกรม Visual Studio Code สามารถเปิดไฟล์ CSV ได้โดยตรง เพราะว่าเป็นไฟล์ข้อความธรรมดា ในกรณีนี้ไฟล์เก็บข้อมูลสินค้าที่ชื่อว่า Product.csv เก็บอยู่ในโฟลเดอร์เดียวกับไฟล์โค้ดภาษา Python เป็นไฟล์ที่มีข้อมูลเป็นภาษาไทยร่วมอยู่ด้วย ดังรูปที่ 6-1

ลำดับ	รายละเอียด
1	รหัสสินค้า, ชื่อสินค้า, ต้นทุน, ราคาขาย
2	A0001, การใช้งานภาษา C#, 290, 325
3	A0002, พัฒนาเว็บด้วย ASP.NET Core MVC, 270, 299
4	A0003, การใช้งานฐานข้อมูล SQL Server 2019, 300, 359
5	A0004, พื้นฐานการเขียนโปรแกรมภาษา Python, 270, 299
6	A0005, การเขียนโปรแกรมเว็บด้วย C#, 250, 279
7	A0006, การพัฒนาเว็บด้วย ASP.NET Core MVC, 299, 315
8	A0007, การเขียนโปรแกรมเบื้องต้นด้วย C++, 245, 299

รูปที่ 6-1 แสดงข้อมูลสินค้าที่อยู่ในไฟล์ Product.csv

การจัดการข้อมูล (CRUD) ในฐานข้อมูล SQL Server ด้วย Python

การทำงานพื้นฐานกับข้อมูลประกอบด้วย 4 อย่าง คือ การเพิ่ม (Create), การอ่านข้อมูล (Read), การแก้ไขข้อมูล (Update) และการลบข้อมูล (Delete) เรียกว่า **CRUD**

ในทางปฏิบัติแล้ว ข้อมูลที่เราต้องทำงานด้วยไม่ได้มาจากการไฟล์ของโปรแกรมต่างๆ เพียงอย่างเดียว แต่ยังรวมถึงข้อมูลที่ถูกจัดเก็บอยู่ในระบบฐานข้อมูลด้วย เราต้องรู้วิธีจัดการข้อมูลเหล่านี้ ด้วยภาษา Python ก่อน

การอ่านข้อมูลจากฐานข้อมูล SQL Server

การทำงานในระดับพื้นฐานที่สุดคือ เราต้องรู้จักวิธีการ Import ข้อมูลจากตาราง Product เข้ามาทำงานในโปรเจกต์ของเราให้ได้ก่อน กล่าวไห้อีกนัยหนึ่งว่า เราต้องการอ่านข้อมูล (Read) จากตารางในฐานข้อมูลนั้นเอง

ตัวอย่างที่ 7-1 การอ่านข้อมูลจากฐานข้อมูล SQL Server

ผู้เขียนต้องการอ่านรายการสินค้าจากตาราง Product แบบไม่มีเงื่อนไข มีโค้ดดังต่อไปนี้

โค้ด Python ที่ 7-1 การอ่านข้อมูลจากฐานข้อมูล SQL Server (CRUD_Read.py)

```
import pyodbc

strConn = 'driver=ODBC Driver 17 for SQL Server;server=.\SQLExpress;database=thaivbBiz;trusted_connection=yes;charset=UTF8'
sql = 'SELECT ProductID, ProductName, ProductCost, ProductPrice FROM Product'

print('รหัสสินค้า' + '\tชื่อสินค้า' + '\t\tต้นทุน' + '\t\tราคาขาย')
print('-----')
with pyodbc.connect(strConn) as Conn:
    data = Conn.execute(sql)
    for item in data:
        print(str(item[0]).ljust(10) + '\t' + str(item[1]).ljust(20)
              + str(item[2]) + '\t' + str(item[3]))

print(type(data))
```



การเพิ่มข้อมูลใหม่เข้าไปในฐานข้อมูล MongoDB

ตัวอย่างที่ 8-1 การเพิ่มข้อมูลในฐานข้อมูล MongoDB

ผู้เรียนต้องการเพิ่มรายการสินค้าใหม่เข้าไปในcollectionชื่อ product ที่มีว่า product ด้วยภาษา Python รหัส 1006 จำนวน 1 เล่ม ดังได้ดังต่อไปนี้

โค้ด Python ที่ 8-1 การเพิ่มข้อมูลในฐานข้อมูล MongoDB (CRUD_Create.py)

```
from pymongo import MongoClient

client = MongoClient()
client = MongoClient('mongodb://localhost:27017/')

mydb = client['thaivbbiz']
mycol = mydb['product']

data = {
    'ProductID' : '1006',
    'ProductName' : 'พัฒนาเว็บแอปด้วย Vue',
    'ProductPrice' : 315,
    'ProductCost' : 279
}

mydb.product.insert_one(data)

for item in mydb.product.find():
    print(item)
    print(type(item))
```

```
File Edit Selection View Go Run Terminal Help
CRUD
CRUD_Create.py x
CRUD_Create.py > ...
1 from pymongo import MongoClient
2
3 client = MongoClient()
4 client = MongoClient('mongodb://localhost:27017/')
5
6 mydb = client['thaivbbiz']
7 mycol = mydb['product']
8
9 data = {
10     'ProductID' : '1006',
11     'ProductName' : 'พัฒนาเว็บแอปด้วย Vue',
12     'ProductPrice' : 315,
13     'ProductCost' : 279
14 }
15
16 mydb.product.insert_one(data)
17
18 for item in mydb.product.find():
19     print(item)
20     print(type(item))
```



บทที่
9

การทำงานกับข้อมูลสูญหายและค่าผิดปกติ

ความต้องการสูงสุดของข้อมูล นั่นคือ ข้อมูลที่อยู่ในสถานะสมบูรณ์แบบ, ไม่มีข้อมูลผิดพลาด, ข้อมูลครบถ้วน และข้อมูลลูก)t้อง เช่น อายุของพนักงานต้องมีทุกคน มีค่าติดลบไม่ได้, อายุเกิน 150 ปีไม่ได้ เป็นต้น เห็นได้ว่า ปัญหาข้อมูลสูญหายถือเป็นปัญหาพื้นฐานข้อแรกที่ผู้อ่านหลีกเลี่ยงไม่ได้ เราจะมาดูกันว่า เราสามารถจัดการปัญหานี้อย่างไรได้บ้าง

ปัญหาของข้อมูลสูญหาย

มีข้อมูลพนักงาน 4 คน ประกอบด้วย Id, Name, Salary และ Age เก็บอยู่ในตัวแปรที่ชื่อว่า data พบว่า ข้อมูลชุดนี้มีปัญหาข้อมูลสูญหายที่เกิดมาจากค่า np.nan, pd.NaT และ None ดังโดยต่อไปนี้

Python – Jupyter Notebook

```
import pandas as pd
import numpy as np

data = {
    'Id' : [100, 200, 300, 400],
    'Name' : ['ศุภชัย', 'สมใจ', 'ศักดิ์ชัย', 'พรชัย'],
    'Salary' : [20000, 22000, np.nan, pd.NaT],
    'Age' : [35, 36, np.nan, None]
}

df = pd.DataFrame(data)
```



การ Import ข้อมูลจากไฟล์ CSV ॥และ Excel

ในกรณีที่ผู้อ่านต้องการ Import ข้อมูลจากไฟล์ CSV หรือ Excel ก็สามารถทำได้เช่นกัน โดยการคลิกเมนู Edit > Import ดังรูปที่ 10-6

The screenshot shows the PandasGUI application window. At the top, there's a menu bar with 'Edit' and 'Debug' selected. A red box labeled '1' indicates the 'Edit' menu. Below it, a sub-menu is open with 'Import' highlighted by a red box labeled '2'. To the right of the sub-menu, a tooltip says 'เลือก' (Select). The main workspace displays a DataFrame with the following data:

	Name	Shape
0	df	5 X 4
1	Product.xlsx - รายชื่อสินค้า	7 X 4

Below the DataFrame, there are tabs for 'DataFrame', 'Filters', 'Statistics', 'Grapher', and 'Reshaper'. The 'DataFrame' tab is active. The main area shows a table with 7 rows and 5 columns. The columns are labeled 'index', 'รหัสลับค่า', 'ชื่อลับค่า', 'จำนวน', and 'ราคาราย'. The data includes various product details and their counts and prices.

รูปที่ 10-6 แสดงข้อมูลที่ได้จากไฟล์ Excel

จากรูปที่ 10-6 พบว่าเราทำงานกับข้อมูล 2 ชุด คือ ข้อมูลที่ได้จากตัวแปร df และข้อมูลที่ได้จากไฟล์ Excel

การ Export ข้อมูลออกเป็นไฟล์ CSV

ผู้อ่านสามารถ Export ข้อมูลของผู้อ่านออกเป็นไฟล์ CSV ได้อีกด้วย ในกรณีนี้ต้องการ Export ข้อมูลที่อยู่ในตัวแปร df เป็นไฟล์ CSV ผลที่ได้คือ PandasGUI จะใช้เครื่องหมาย , คั่นข้อมูลระหว่าง คอลัมน์ ดังรูปที่ 10-7

The screenshot shows the PandasGUI application window. At the top, there's a menu bar with 'Edit' and 'Debug' selected. A red box labeled '1' indicates the 'Edit' menu. Below it, a sub-menu is open with 'Export' highlighted by a red box labeled '2'. To the right of the sub-menu, a tooltip says 'เลือก' (Select). The main workspace displays a DataFrame with the following data:

	Name	Shape
0	df	5 X 4
1	Product.xlsx - รายชื่อสินค้า	7 X 4



การหาค่าเฉลี่ยเลขคณิตกับตัวเลขทศนิยมด้วยฟังก์ชัน fmean() ของ statistics

ในกรณีที่ผู้อ่านมีข้อมูลที่เป็นตัวเลขทศนิยม ผู้อ่านสามารถหาค่าทางสถิติเบื้องต้น โดยอาศัยกลุ่มฟังก์ชันของ Package ที่ชื่อว่า **statistics** เข้ามาทำงานด้านนี้โดยเฉพาะเลยก็ได้

ให้ผู้อ่านพิมพ์คำสั่งต่อไปนี้ใน TERMINAL ของโปรแกรม Visual Studio Code เพื่อดownload และติดตั้ง statistics ก่อน

TERMINAL

```
pip install statistics
```

ภาพหน้าจอแสดง Terminal ใน Visual Studio Code. ผู้ใช้พิมพ์คำสั่ง `pip install statistics` แล้วกดปุ่ม Enter. ผลลัพธ์แสดงว่า package 'statistics' ถูก download และ install 成 công.

```
PS C:\Projects> pip install statistics
Defaulting to user installation because normal site-packages is not writeable
Collecting statistics
  Downloading statistics-1.0.3.5.tar.gz (8.3 kB)
Collecting docutils>=0.3
  Downloading docutils-0.16-py2.py3-none-any.whl (548 kB)
    ██████████ | 548 kB 2.2 MB/s
Using legacy 'setup.py install' for statistics, since package 'wheel' is not installed.
Installing collected packages: docutils, statistics
  Running setup.py install for statistics ... done
Successfully installed docutils-0.16 statistics-1.0.3.5
PS C:\Projects>
```

รูปที่ 11-7 แสดงการดาวน์โหลดและติดตั้ง statistics

ใน statistics มีฟังก์ชันที่ชื่อว่า **fmean()** ทำหน้าที่คำนวณค่าเฉลี่ยเลขคณิตกับข้อมูลที่เป็นตัวเลขทศนิยมโดยเฉพาะ ดังโค้ดต่อไปนี้

Python

```
import statistics as st

data = [25.10, 40.68, 35.99, 23.4]
result = st.fmean(data)

print('{0:.3f}'.format(result))
```



ตัวอย่างที่ 13-3 การคำนวณส่วนเบี่ยงเบนเฉลี่ยใน NumPy

ในการนี้ที่ผู้อ่านต้องการใช้งาน NumPy เข้ามาคำนวณส่วนเบี่ยงเบนเฉลี่ย ต้องใช้ 2 พังก์ชัน คือ

- พังก์ชัน `mean()` สำหรับคำนวณค่าเฉลี่ยเลขคณิต
- พังก์ชัน `absolute()` สำหรับตัดเครื่องหมายลบออกจากผลต่าง ดังได้ดังต่อไปนี้

ตัวอย่างที่ 13-3 การคำนวณส่วนเบี่ยงเบนเฉลี่ยใน NumPy (md.py)

```
from numpy import mean, absolute
```

```
data = [10, 15, 16, 20, 17]
```

```
result = mean(absolute(data - mean(data)))
print(result)
```

The screenshot shows a code editor window for a file named 'md.py'. The code is as follows:

```
md.py    ×
.vscode > md.py > ...
1  from numpy import mean, absolute
2
3  data = [10, 15, 16, 20, 17]
4
5  result = mean(absolute(data - mean(data)))
6  print(result)
```

Below the code editor is a terminal window showing the execution of the script:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
TERMINAL
PS D:\Project> d;; cd 'd:\Project'; & 'C:\Program Files\Python3
1\pythonFiles\lib\python\debugpy\launcher' '50449' '--' 'd:\Proj
2.48
PS D:\Project>
```

รูปที่ 13-3 แสดงส่วนเบี่ยงเบนเฉลี่ยของ NumPy

อธิบายการทำงานของโค้ด

- เริ่มต้นให้หาผลต่างของข้อมูลแต่ละตัวกับค่าเฉลี่ยเลขคณิต (`mean(data)`) ก่อน แล้วใช้พังก์ชัน `absolute()` เข้ามาทำหน้าที่ตัดเครื่องหมายลบออก

md.py

```
absolute(data - mean(data))
```



ตัวอย่างที่ 13-9 การคำนวณค่าสัมประสิทธิ์ของส่วนเบี่ยงเบนเฉลี่ยใน NumPy มีโค้ดดังต่อไปนี้

โค้ดไฟรอนที่ 13-9 การคำนวณค่าสัมประสิทธิ์ของส่วนเบี่ยงเบนเฉลี่ยใน NumPy (ca.py)

```
from numpy import mean, absolute

a = [20, 11, 12, 10, 9]
b = [15, 16, 17, 18, 17]

mean_a = mean(a)
md_a = mean(absolute(a - mean(a)))
cv_a = md_a/mean_a

mean_b = mean(b)
md_b = mean(absolute(b - mean(b)))
cv_b = md_b/mean_b

print('ค่าเฉลี่ย A : ', mean_a)
print('ส่วนเบี่ยงเบนเฉลี่ย A : ', md_a)
print('สัมประสิทธิ์ส่วนเบี่ยงเบนเฉลี่ย A : ', cv_a)
print('-----')
print('ค่าเฉลี่ย B : ', mean_b)
print('ส่วนเบี่ยงเบนเฉลี่ย B : ', md_b)
print('สัมประสิทธิ์ส่วนเบี่ยงเบนเฉลี่ย B : ', cv_b)
```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
			PS D:\Project> d;; cd 'd:\Project'; & 'C:\Program 61\pythonFiles\lib\python\debugpy\launcher' '49980 ค่าเฉลี่ย A : 12.4 ส่วนเบี่ยงเบนเฉลี่ย A : 3.04 สัมประสิทธิ์ส่วนเบี่ยงเบนเฉลี่ย A : 0.24516129032258063 ----- ค่าเฉลี่ย B : 16.6 ส่วนเบี่ยงเบนเฉลี่ย B : 0.8799999999999997 สัมประสิทธิ์ส่วนเบี่ยงเบนเฉลี่ย B : 0.05301204819277106 PS D:\Project> █

รูปที่ 13-10 พลการคำนวณสัมประสิทธิ์ของส่วนเบี่ยงเบนเฉลี่ย

อธิบายการทำงานของโค้ด

- ที่ข้อมูลชุด A คำนวณค่าเฉลี่ยเลขคณิตและส่วนเบี่ยงเบนเฉลี่ยก่อน กล่าวคือ
 - ค่าเฉลี่ยเลขคณิต** ใช้ฟังก์ชัน `mean()` เก็บไว้ที่ตัวแปร `mean_a`
 - ส่วนเบี่ยงเบนเฉลี่ย** ใช้ฟังก์ชัน `mean()` ร่วมกับฟังก์ชัน `absolute()` เก็บไว้ที่ตัวแปร `md_a`
 - สัมประสิทธิ์ส่วนเบี่ยงเบนเฉลี่ย** โดยการนำตัวแปร `md_a` หารด้วยตัวแปร `mean_a`



การกำหนดความเข้มของจุดข้อมูล

ในกรณีที่ผู้อ่านมีข้อมูลจำนวนมาก มีโอกาสสูงมากที่ข้อมูลจะอยู่ในตำแหน่งทับซ้อนกัน ทำให้การสื่อสารข้อมูลอาจจำไม่ชัดเจน สื่อสารไม่ครบถ้วน วิธีการแก้ไขปัญหานี้ก็คือ ผู้อ่านสามารถกำหนดให้ข้อมูลแสดงผลด้วยความเข้มของแต่ละจุด

ตัวอย่างที่ 14-4 การกำหนดความเข้มของจุดข้อมูล

ผู้เขียนกำหนดให้ข้อมูลชุดที่ 1 แสดงผลตามปกติ ส่วนข้อมูลชุดที่ 2 กำหนดให้แสดงผล 50% (จากกว่าปกติหรือกึ่งไปร่องใส) โดยอาศัยพารามิเตอร์ที่ชื่อว่า **alpha** กำหนดค่า 0.5 ดังโค้ดต่อไปนี้

โค้ดไฟรอนที่ 14-4 การกำหนดความเข้มของจุดข้อมูล (MultipleData.py)

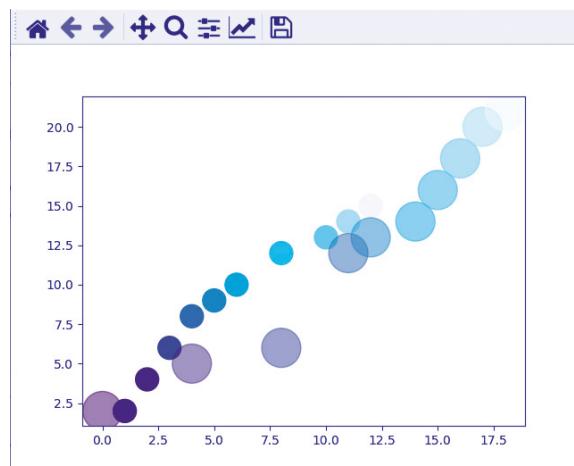
```
import matplotlib.pyplot as plt
import numpy as np

x1 = [1, 2, 3, 4, 5, 6, 8, 10, 11, 12]
y1 = [2, 4, 6, 8, 9, 10, 12, 13, 14, 15]

x2 = [0, 4, 8, 11, 12, 14, 15, 16, 17, 18]
y2 = [2, 5, 6, 12, 13, 14, 16, 18, 20, 21]

colors = np.array([0, 10, 20, 30, 40, 50, 60, 70, 80, 90])

plt.scatter(x1, y1, c=colors, s=350)
plt.scatter(x2, y2, c=colors, s=1000, alpha=0.5)
plt.show()
```



รูปที่ 14-7 แสดงข้อมูลที่กับซ้อนกันอยู่

จากรูปที่ 14-7 เมื่อข้อมูลทั้ง 2 ชุดมีความเข้มของแต่ละจุด ทำให้การดูข้อมูลชัดเจนยิ่งขึ้น

การสร้างกราฟแท่ง (Bar Graph)

กราฟแท่ง (Bar Graph) ใช้ในการแสดงความแตกต่างของข้อมูลได้อย่างชัดเจน เป็นกราฟอีกรูปแบบหนึ่งที่เราคุ้นเคยกันเป็นอย่างดี เป็นหน้าที่ของฟังก์ชันที่ชื่อว่า **bar()**

ตัวอย่างที่ 14-5 พื้นฐานการสร้างกราฟแท่ง

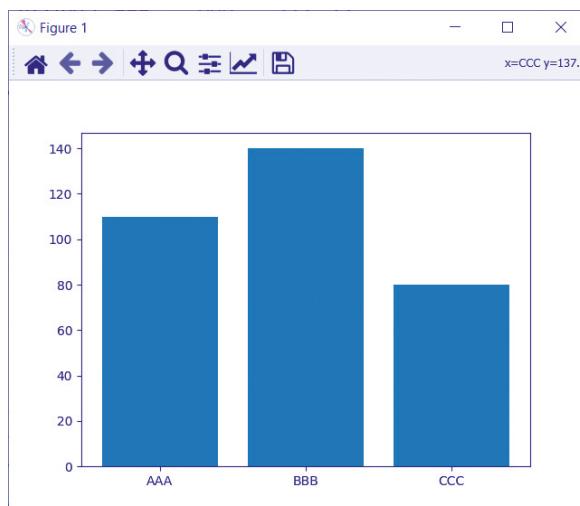
มีข้อมูล 3 ชุดของนาย AAA, BBB และ CCC ต้องการแสดงข้อมูลในรูปแบบกราฟแท่ง ดังได้ดังต่อไปนี้

ได้ด้วย命令ที่ 14-5 พื้นฐานการสร้างกราฟแท่ง (UsingBar.py)

```
import matplotlib.pyplot as plt  
import numpy as np  
  
name = np.array(["AAA", "BBB", "CCC"])  
data = np.array([110, 140, 80])  
  
plt.bar(name, data)  
plt.show()
```

โดยปกติแล้ว เราจะได้กราฟแท่งตามแนวตั้ง โดยที่

- **แกน x** แสดงชื่อ ได้จากตัวแปร name
- **แกน y** แสดงค่าของข้อมูลแต่ละตัว ได้จากตัวแปร data ดังรูป 14-8



รูปที่ 14-8 แสดงกราฟแท่งแนวตั้ง

การแบ่งสัดส่วนข้อมูลออกจากส่วนของกราฟวงกลม

ในกรณีที่ผู้อ่านต้องการเน้นสัดส่วนของข้อมูลบางตัว ให้โดดเด่นมากกว่าส่วนอื่นๆ ก็สามารถกำหนดให้แบ่งส่วนดังกล่าว แสดงผลลอกห่างจากกราฟวงกลมโดยการกำหนดค่า `explode` เช่น ต้องการกำหนดให้ข้อมูลตัวที่ 1 (A) กับตัวที่ 3 (C) แบ่งส่วนออกไป โดยการสร้างตัวแปรที่ชื่อว่า `data_explode` ค่าที่กำหนดขึ้นมาอยู่มีค่ามากทำให้มีระยะห่างมากขึ้น จากนั้นนำไปกำหนดค่าให้กับพารามิเตอร์ `explode` ดังโค้ดต่อไปนี้

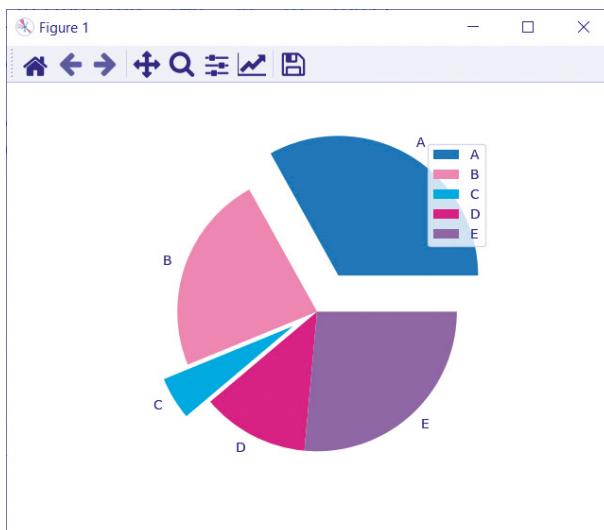
ตัวอย่างที่ 14-7 การแบ่งสัดส่วนข้อมูลออกจากส่วนของกราฟวงกลม

โค้ดไฟรอนที่ 14-7 การแบ่งสัดส่วนข้อมูลออกจากส่วนของกราฟวงกลม (UsingPie.py)

```
import matplotlib.pyplot as plt
import numpy as np

data = np.array([200, 140, 30, 75, 160])
data_label = ['A', 'B', 'C', 'D', 'E']
data_explode = [0.3, 0, 0.2, 0, 0]

plt.pie(data, labels = data_label, explode=data_explode)
plt.legend()
plt.show()
```



รูปที่ 14-13 กราฟวงกลมที่มีการแบ่งสัดส่วนออกไป

การสร้างกราฟ BoxPlot

กราฟ BoxPlot ทำหน้าที่บอกผู้อ่านว่า ข้อมูลที่เราสนใจอยู่มีการกระจายตัวมากน้อยเพียงใด ก็จะช่วยทำให้มองข้อมูลได้ครบถ้วนยิ่งขึ้น

ตัวอย่างที่ 14-15 การสร้างกราฟ BoxPlot

เพื่อให้ผู้อ่านเข้าใจวิธีการดูกราฟแบบ BoxPlot ผู้เขียนเลือกใช้ข้อมูลแบบปกติ มีลักษณะเกาเกลี่มกัน กระจายตัวน้อยเป็นข้อมูลที่ไม่มีค่ามากเกินไปหรือน้อยเกินไป (Outliers) คือ 42, 40, 45, 48, 50 และ 45 จากนั้นนำข้อมูลชุดนี้มาสร้าง BoxPlot ดังโค้ดต่อไปนี้

โค้ดโปรแกรมที่ 14-15 การสร้างกราฟ BoxPlot (UsingBoxPlot.py)

```
import matplotlib.pyplot as plt
import numpy as np

data = [42, 40, 45, 48, 50, 45]
fig = plt.figure(1, figsize=(10, 6))
ax = fig.add_subplot(111)

ax.boxplot(data, vert=False, manage_ticks=True)
ax.set_xlabel('Values')
ax.set_yticklabels(['Products'])

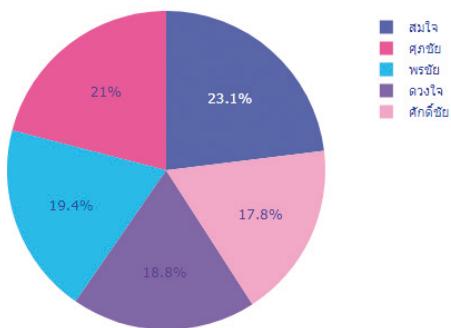
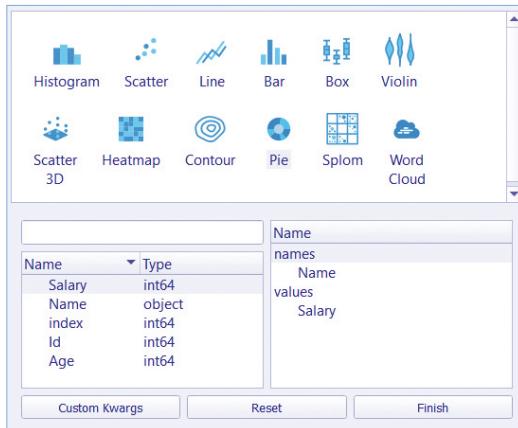
q = np.quantile(data, [0.00, 0.25, 0.50, 0.75, 1.00])
ax.vlines(q, ymin=0, ymax=100, color='red', ls=':')

ax.set_xticks(q)
ax.set_xlim(0.5, 1.5)
plt.show()
```

การสร้างกราฟวงกลม (Pie Graph)

การสร้างกราฟวงกลม (Pie Graph) เป็นกราฟที่กำหนดให้แสดงสัดส่วนข้อมูลเพื่อเปรียบเทียบว่า ข้อมูลใดมีมากน้อยกว่ากัน ต้องการข้อมูล 2 อย่าง คือ

- **names** หมายถึง ชื่อของข้อมูล ในกรณีนี้คือ ชื่อพนักงาน Name
- **values** หมายถึง ข้อมูลที่ต้องการแสดงค่า ในกรณีนี้คือ เงินเดือน Salary



รูปที่ 15–9 แสดงกราฟวงกลม

สรุปหัวยี่เล่ม

ในการทำงานกับข้อมูลด้วยภาษาไพธอน ที่ผู้เขียนเลือกมาดำเนินการในหนังสือเล่มนี้ มีเจตนา ต้องการให้ผู้อ่านทราบว่า ข้อมูลที่ผู้อ่านกำลังสนใจอยู่ ล้วนแล้วแต่มีสิ่งที่ถูกปกปิดซ่อนอยู่ภายในความสามารถใช้ประโยชน์จากสิ่งที่ซ่อนอยู่ ยอมได้เบรียบกว่าการไม่ทราบอะไรเลย อาจจะเป็นเพียง เนื้อหาในขั้นต้น ไม่สามารถนำเสนอได้ครบถ้วนทุกแง่มุม แต่อย่างน้อยก็อยากให้เป็นรากฐานไปสู่ ในระดับต่อไป... สวัสดีครับ