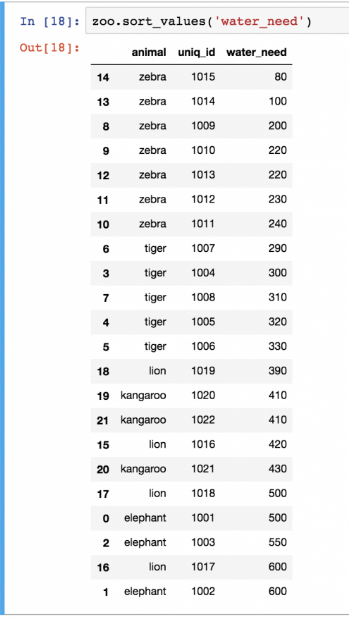
**Pandas**

**การเรียงข้อมูล (Data Sorting) และ การเติมข้อมูล (Data Munging)**

การเรียงข้อมูล (Data Sorting) มีวัตถุประสงค์เพื่อจัดลำดับข้อมูลที่มีอยู่ให้ง่ายต่อการใช้งาน การเรียงลำดับสามารถเรียงข้อมูล (จำพวกตัวเลข) จากมากไปน้อย หรือน้อยไปมาก คำสั่งใน Pandas คือ .sort\_values( ) เช่น



1. **จากตัวอย่างด้านบนให้นักเรียนโหลดข้อมูล zoo.csv และให้ลองคำสั่ง .sort\_values( ) ด้านบน**

จาก exercise ที่ผ่านมาถ้านักเรียนสังเกตจะพบว่าตัวแปร (parameter) ที่กำหนดในคำสั่ง .sort\_values( ) คือชื่อ column ที่เราต้องการ sort ในที่นี้คือ water\_need เราสามารถ sort ข้อมูลโดยกำหนดตัวแปรได้หลาย column ซึ่งมี syntax ดังต่อไปนี้ (เราสามารถใช้ syntax ‘by’ กับ column เดียวก็ได้)

zoo.sort\_values(by = ['animal', 'water\_need'])

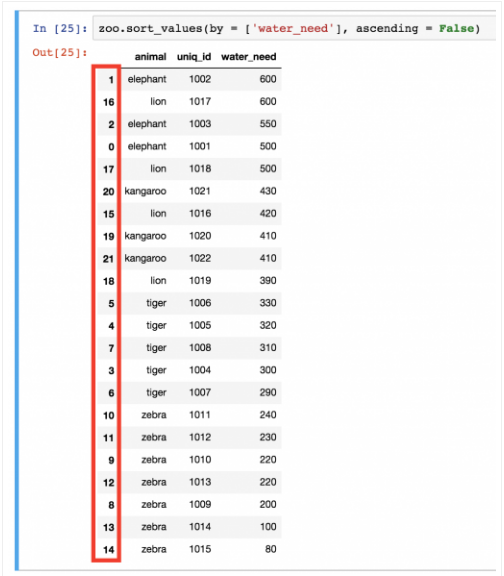
1. **ให้นักเรียนลองคำสั่งด้านบน และดูผลลัพธ์ว่าการเรียงลำดับของข้อมูลเป็นอย่างไร**
2. **ให้นักเรียน sort ข้อมูลโดยกำหนดให้ water\_need อยู่ก่อน animal และดูผลลัพธ์ของการเรียงข้อมูลว่าต่างจาก exercise ก่อนหน้าอย่างไร**

คำสั่ง sort\_values( ) โดยค่า default แล้วจะเรียงข้อมูลจากน้อยไปมาก (ascending order) หากเราต้องการเรียงข้อมูลจากค่ามากไปน้อย (descending order) คำสั่งจะมี syntax ดังต่อไปนี้

zoo.sort\_values(by = ['water\_need'], ascending = False)

1. **ให้นักเรียนลองคำสั่งด้านบน และดูผลลัพธ์ว่าการเรียงลำดับของข้อมูลเป็นอย่างไร**

นักเรียนจะสังเกตได้ว่าข้อมูลเรียงลำดับจากมากไปน้อยตาม water\_need แต่ช้าก่อน ถ้าสังเกตุ column แรกซึ่งคือ index ของแต่ละ record จะพบว่าเลข index ไม่ได้ถูกเรียงลำดับให้เหมาะสม ซึ่งจะส่งผลให้การอ้างถึงหรือการแสดงข้อมูลในรูปแบบ graph อาจผิดพลาดได้



ดังนั้นในบางกรณี เช่น การทำ Visualization การทำ Machine Learning) เราจำเป็นต้องกำหนดค่า index (การทำ re-index) ใหม่ ซึ่งมี syntax ดังต่อไปนี้

zoo.sort\_values(by = ['water\_need'], ascending = False)**.reset\_index()**

1. **ให้นักเรียนลองคำสั่งด้านบน และดูผลลัพธ์**

จากผลลัพธ์นักเรียนจะสังเกตเห็นว่าข้อมูล (Dataframe) ยังเก็บ index เก่าไว้ด้วย ซึ่งตอนนี้มีชื่อ column คือ index หากนักเรียนต้องการเอา column นี้ออก สามารถทำได้โดยใช้คำสั่ง

zoo.sort\_values(by = ['water\_need'], ascending = False).reset\_index**(drop = True)**

1. **ให้นักเรียนลองคำสั่งด้านบน และดูผลลัพธ์**

จาก worksheet ครั้งก่อนในหัวข้อเกี่ยวกับการรวมข้อมูล ถ้านักเรียนยังจำได้เมื่อเราทำการรวมข้อมูล zoo.csv และ zoo\_eats.csv จะมีข้อมูลบาง record ที่มีค่า NaN ปรากฏอยู่ ซึ่งอาจทำให้เรารู้สึกรำคาญได้ หรือในกรณีที่เราต้องวิเคราะห์ข้อมูล หรือทำ Machine Learning เราจำเป็นต้องมีข้อมูลที่ครบถ้วน ซึ่งเราสามารถเติมข้อมูลได้โดยใช้คำสั่ง .fillna( )

1. **ให้นักเรียนโหลดข้อมูล zoo.csv และ zoo\_eats.csv ที่ทำก่อนหน้านี้ และทำการ merge ข้อมูลเข้าด้วยกัน (เลือก option เป็น left หลังจากทำเสร็จแล้วควรจะมีข้อมูลทั้งหมด 22 records)**

ในการเติมข้อมูลถือเป็นศาสตร์อย่างหนึ่ง เนื่องจากข้อมูลที่เติมเข้าไปจำเป็นต้องมีความหมายหรือค่าที่เหมาะสม ถ้าเป็นทั่วๆ ไปแล้ว สำหรับค่าที่เป็นตัวเลขเราอาจเติมค่า 0 หรือค่าเฉลี่ยของ Column นั้น ในกรณีที่เป็นตัวอักษรเราก็ต้องมีตัวอักษรเฉพาะที่เราต้องการเติม เช่น ถ้าเป็น column ที่ระบุว่าเป็น kid หรือ adult แล้วมี record ที่มีค่า NaN อยู่ เราอาจหาว่าระหว่าง kid กับ adult มีค่าไหนมากกว่ากัน แล้วเราก็เติมค่าที่มากกว่าให้กับ record ที่มีค่า NaN (ประมาณว่าพวกมากลากไป)

อย่างไรก็ตามเราสามารถใช้คำ unknown ได้ ซึ่งจะทำให้มีข้อมูลประเภทที่สามขึ้นมา เช่น kid adult และ unknown ในกรณีที่เราไม่ต้องการข้อมูลประเภทใหม่ เราอาจใช้เทคนิคที่ได้อธิบายแล้วด้านบน

1. **ให้นักเรียนลองคำสั่งต่อไปนี้ และดูผลลัพธ์ (ใน record ของ lion ค่า NaN จะต้องเปลี่ยนเป็น unknown)**

zoo.merge(zoo\_eats, how = 'left')**.fillna('unknown')**

จากข้อมูลนักเรียนลองสังเกตุว่าจะมีสัตว์กินพืช (Herbivore) และสัตว์กินเนื้อ (carnivore) ถ้าเราไม่ต้องการเติม unknown ให้กับ lion นักเรียนคิดว่าควรจะเติมค่าอะไร

1. **ให้นักเรียนลองเติมค่าที่เหมาะสมให้กับ lion**

กลับมาที่ข้อมูล pandas\_tutorial\_read.csv เราจะมาดูกันว่า exercise ที่นักเรียนได้ทำผ่านมา จะนำความรู้มาแก้ปัญหาต่อไปนี้อย่างไร

1. **ให้นักเรียนโหลดข้อมูลจาก pandas\_tutorial\_buy.csv (โหลดได้จาก portal) ไว้ที่ตัวแปร blog\_buy และกำหนด column names ตามลำดับดังนี้ 'my\_date\_time', 'event', 'user\_id', 'amount' และทำการโหลด pandas\_tutorial\_read.csv ไว้ที่ตัวแปร acritical\_read โดยกำหนด column names ตามลำดับดังนี้ 'my\_datetime', 'event', 'country', 'user\_id', 'source', 'topic' (นักเรียนสามารถย้อนดูคำสั่งได้ที่ worksheet1)**
2. **ให้นักเรียน import numpy หลังจาก import pandas ด้วย โดยใช้คำสั่ง import numpy as np**

**ข้อมูลจาก pandas\_tutorial\_read.csv เป็นข้อมูลที่ระบุคนที่อ่านบทความบน blog นั้น ในขณะที่ pandas\_tutorial\_buy.csv เป็นข้อมูลที่ระบุคนที่ซื้อของบางอย่างจาก blog นั้นช่วงเวลาระหว่าง 2018-01-01 และ 2018-01-07**

เราจะมาลองหาค่าเฉลี่ยของรายได้ revenue ที่เกิดจากคนที่ซื้อของจาก blog ระหว่างช่วงเวลา 2018-01-01 และ 2018-01-07 (ซึ่งมีขั้นตอนดังนี้)

1. **ให้นักเรียนทำการ merge ข้อมูลจาก blog\_buy ไปยัง article\_read โดยเลือก how = 'left' และกำหนด key column จากทั้งสอง dataframes เป็น user\_id (กำหนดโดยใช้ left\_on และ right\_on) ไว้ที่ตัวแปร step\_1**
2. **ให้นักเรียนเลือก column ชื่อ amount เพราเป็น column ที่เก็บจำนวนเงิน (เราต้องการรู้เฉพาะรายได้) จาก step\_1 ไว้ที่ตัวแปร step\_2**
3. **ให้นักเรียนเติมข้อมูล NaN ด้วยค่า 0 (ทำไมเป็นค่า 0 เพราะคนคนนั้นถึงไม่มีการซื้อของ แต่เมื่อเราต้องการหาค่า mean เราก็ต้องนำข้อมูลมาในการคำนวณด้วย) จากตัวแปร step\_2 ไว้ที่ step\_3**
4. **ให้นักเรียนหาค่าเฉลี่ย จากตัวแปร step\_3 ไว้ที่ตัวแปร result และเรียกตัวแปร result เพื่อดูค่า (ถ้านักเรียนทำถูกต้องควรจะได้คำตอบเท่ากับ 1.0852)**

ต่อไปเราจะมาลองหาประเทศสามประเทศที่มีค่า revenue สู่งสุด ระหว่างช่วงเวลา 2018-01-01 และ 2018-01-07 (ซึ่งมีขั้นตอนดังนี้)

1. **ให้นักเรียน import pandas as pd และ import numpy as np พร้อมทั้งโหลดข้อมูลจาก pandas\_tutorial\_read.csv และ pandas\_tutorial\_buy.csv โดยมี column names ตาม exercise 10**
2. **ให้นักเรียนทำการ merge ข้อมูลจาก blog\_buy ไปยัง article\_read โดยเลือก option how = 'left' และกำหนด key column จากทั้งสอง dataframes เป็น user\_id (กำหนดโดยใช้ left\_on และ right\_on) ไว้ที่ตัวแปร step\_1**
3. **ให้นักเรียนเติมข้อมูล NaN ด้วยค่า 0**
4. **ให้นักเรียนจัดกลุ่มตาม country (คำสั่ง .groupby( )) และหาผลรวม .sum( )**
5. **ให้นักเรียนเลือกเฉพาะ amount**
6. **ให้นักเรียน sort จากมากไปน้อย (จะได้ดูค่าสูงสุดได้)**
7. **ให้นักเรียนแสดง 3 records แรก ด้วย คำสั่ง .head( )**