Awesome Jupyter Notebook

Session 2

Python Workshop for Benjama EEP

Objectives

1. bool - True, False
2. If-else (04 If Statements.ipynb)
3. While (05 While Loops and User Input.ipynb)
4. For
5. Function (07 Introduction to Functions.ipynb)
6. Object Oriented Programming (Object Oriented Programming.ipynb)
7. Awesome Data Science Notebooks (https://github.com/jakevdp/PythonDataScienceHandbook)

List - ลำดับ

```
speakers = [ 'K', 'Kai', 'Or', 'Paul', 'Tom', 'Woot' ]
heights = [ 180, 164, 165, 165, 180, 170 ]
pop = [ 3.25, 3.9, 4, 3, 3.5, 3.44 ]
```

เข้าถึงข้อมูลในลำดับ

```
print(speakers[0])
print(speakers[1])
print(speakers[2])
```

การกำหนดค่าลำดับ

```
speakers[0] = 'Kriengsak'
print(str(speakers))
speakers[0] = 'K'
print(str(speakers))
```

การหาค่าต่ำสุด-สูงสุด-ผลรวม ของลำดับ

```
print( max(heights) )
print( min(pop) )
print( sum(heights) )
```

bool - **ค่าจริง-เท็จ**

```
speakers = [ 'K', 'Kai', 'Or', 'Paul', 'Tom', 'Woot' ]
heights = [ 180, 164, 165, 165, 180, 170 ]
popularity = [ 3.25, 3.9, 4, 3, 3.5, 3.44 ]

print( 'K' in speakers )
print( 'Justin Bieber' in speakers )
print( speakers[0] == 'K' )
print( speakers[3] == 'Wichit' )
```

If-else

เงื่อนไขเดียว

```
if ค่าจริงเท็จ:
    คำสั่งที่จะทำเมื่อเป็นจริง
else:
    คำสั่งที่จะทำเมื่อเป็นเท็จ
```

ตัวอย่าง

```
if 'Justin Bieber' in speakers:
    print('นักร้องดังนานาชาติมา')
else:
    print('ไม่มีนักร้องดังมางาน')
```

หลายเงื่อนไข

```
if เงื่อนไข1:
    คำสั่งที่จะทำเมื่อเงื่อนไข1เป็นจริง
elif เงื่อนไข2:
    คำสั่งที่จะทำเมื่อเงื่อนไข2เป็นจริง
elif เงื่อนไข3:
    คำสั่งที่จะทำเมื่อเงื่อนไข3เป็นจริง
else:
    คำสั่งที่จะทำเมื่อเงื่อนไขทั้งหมดเป็นเท็จ
```

While

```
while เงื่อนไข:
    คำสั่งที่จะทำเมื่อตรวจสอบเงื่อนไขในแต่ละรอบแล้วเป็นจริง
```

ตัวอย่าง

```
i = 0
while i < 10:
    print('i = '+str(i))
    i = i+1
```

For

```
for สมาชิก in ลำดับ:
    คำสั่งที่จะทำสำหรับสมาชิกแต่ละตัว
```

ตัวอย่าง

```
speakers = [ 'K', 'Kai', 'Or', 'Paul', 'Tom', 'Woot' ]
for speaker in speakers:
    print(speaker)
```

Function - **การเขียนฟังก์ชัน**

```
def ชื่อฟังก์ชัน(รายการข้อมูลที่รับมา):
    คำสั่งภายในฟังก์ชัน
```

ตัวอย่าง

```
def statsummary(a):
    import statistics
    print('ลำดับ: '+str(a))
    print('ค่าเฉลี่ยของลำดับ: '+str(statistics.mean(a)))
    print('ค่าเบี่ยงเบนมาตรฐานของลำดับ: '+str(statistics.stdev(a)))

heights = [ 180, 164, 165, 165, 180, 170 ]
statssummary(heights)
```

Dictionary/json - **ชนิดข้อมูลที่นิยมใช้แลกเปลี่ยนข้อมูล**

- อัตราแลกเปลี่ยน https://api.fixer.io/lastest (https://api.fixer.io/lastest)
- ข้อมูลผู้ใช้บน github https://api.github.com/users (https://api.github.com/users)
- Google: YouTube Data API, Live Stream, Analytics https://developers.google.c /youtube/v3/ (https://developers.google.com/youtube/v3/)
- Facebook: Graph API (https://developers.facebook.com/docs/graph-api/)
- The Movie Database (https://www.themoviedb.org/)

ข้อมูลจากเครื่องวิทยากร

แบบ json

- http://หมายเลขไอพีเครื่องวิทยากร/api/info (http://หมายเลขไอพีเครื่องวิทยากร/api/inf
- http://หมายเลขไอพีเครื่องวิทยากร/api/exchange (http://หมายเลขไอพีเครื่องวิทยากร/ /exchange)

แบบข้อความธรรมดา

- http://หมายเลขไอพีเครื่องวิทยากร/speakers/txt (http://หมายเลขไอพีเครื่อง วิทยากร/speakers/txt)
- http://หมายเลขไอพีเครื่องวิทยากร/height/txt (http://หมายเลขไอพีเครื่องวิทยากร/heig /txt)
- http://หมายเลขไอพีเครื่องวิทยากร/pop/txt (http://หมายเลขไอพีเครื่องวิทยากร/pop/tx

รูปแบบข้อมูล dictionary

```
speakers = {
  'K': 'เกรียงศักดิ์',
  'Woot': 'วราวุฒิ',
  'Tom': 'ไพชยนต์'
}
```

เนื่องจาก str ใน python ใช้ได้ทั้ง '' และ ""

```
speakers = {
  "K": "เกรียงศักดิ์",
  "Woot": "วราวุฒิ",
  "Tom": "ไพชยนต์"
}
```

> ตัวอย่าง info

```
{
    "K": {
        "ชื่อ-สกุล": "เกรียงศักดิ์ ตรีประพิณ",
        "e-mail": "kriengsak.t@ubu.ac.th",
        "picture":  "image/K.jpg"
    },
    "Kai": {
        "ชื่อ-สกุล": "ปิยนันท์ พนากานต์",
        "e-mail": "piyanan.p@ubu.ac.th",
```

การรับส่งข้อมูล json

```
import requests
r = requests.get('http://api.fixer.io/latest')
x = r.json()
print( type(x) )
print(x)
```

More on import

คำสั่งเพื่อสร้างหน้าต่างขึ้นมา

```
import sys
import PyQt5.QtWidgets

app = PyQt5.QtWidgets.QApplication(sys.argv)
w = PyQt5.QtWidgets.QWidget()
w.resize(250, 150)
w.move(300, 300)
w.show()

sys.exit(app.exec_())
```

เราสามารถทำการย่อชื่อโดยใช้คำสั่ง import ช่วยให้สั้นลงได้ดังนี้

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget

app = QApplication(sys.argv)
w = QWidget()
w.resize(250, 150)
w.move(300, 300)
w.show()

sys.exit(app.exec_())
```

แนวคิดเชิงวัตถุ และการสืบทอดคุณสมบัติ

เราสามารถเขียนนิยามหน้าต่างของเราเองได้ดังนี้

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget
from PyQt5.QtGui import QIcon


class MyWindow(QWidget):

    def __init__(self):
        super().__init__()
        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('Icon')
        self.setWindowIcon(QIcon('icon.png'))
        self.show()


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

In [ ]: