# Basic File I/O

# พื้นฐานการนำเข้าและส่งออกข้อมูลจากไฟล์

- การนำเข้าและส่งออกข้อมูลจากไฟล์เป็นส่วนที่สำคัญที่ใช้สำหรับการพัฒนาเว็ปแอฟพลิเคชั่น และงานประยุกต์ (application) ขนาดใหญ่
- ภาษา Python มีฟังก์ชันสำหรับนำเข้าและส่งออกข้อมูลจากไฟล์หลายฟังก์ชัน เช่น การสร้าง (creating) การอ่าน (reading) การเขียน (writing) การเขียนต่อท้าย (appending) และการลบ (deleting)

# ฟังก์ชัน open()

- ฟังก์ชันหลักในการทำงานกับไฟล์ในภาษา Python คือฟังก์ชัน open()
- ฟังก์ชัน open() รับค่าพารามิเตอร์ 2 ค่า คือ filename และ mode
- โดย filename คือ ชื่อไฟล์ที่เราต้องการทำงานด้วย

# ฟังก์ชัน open() ต่อ

- ส่วน mode คือวิธีการสำหรับการเปิดไฟล์ มีรายละเอียดคือ
  - r หมายถึง Read เป็นการเปิดไฟล์เพื่อ ‘อ่าน’ หากไม่มีไฟล์ชื่อตามที่ระบุใน filename จะส่งข้อมูลว่าเกิดข้อผิดพลาด โดยภาษา Python หากไม่กำหนดค่า mode จะตั้งให้เป็น r โดยอัตโนมัติ
  - a หมายถึง Append เป็นการเปิดไฟล์เพื่อ ‘เขียนต่อท้าย’ หากไม่มีไฟล์ชื่อตามที่ระบุใน filename จะ ‘สร้าง’ ไฟล์ใหม่แทน

# ฟังก์ชัน open() ต่อ

- ส่วน mode คือวิธีการสำหรับการเปิดไฟล์ มีรายละเอียดคือ
  - w หมายถึง Write เป็นการเปิดไฟล์เพื่อ ‘เขียนทับ’ หากไม่มีไฟล์ชื่อตามที่ระบุใน filename จะ ‘สร้าง’ ไฟล์ใหม่แทน
  - x หมายถึง Create เป็นการ ‘สร้าง’ ไฟล์ขึ้นมาใหม่ ให้มีชื่อตามที่ระบุใน filename หากไม่มีไฟล์ชื่อตามที่ระบุใน filename จะส่งข้อมูลว่าเกิดข้อผิดพลาด

# ไวยากรณ์สำหรับฟังก์ชัน open()

- ฟังก์ชัน open() มีรูปแบบคำสั่งดังนี้

  ชื่อตัวแปร = open(filename, mode)

  โดย filename และ mode ต้องเขียนภายในเครื่องหมาย ‘’ หรือ “”

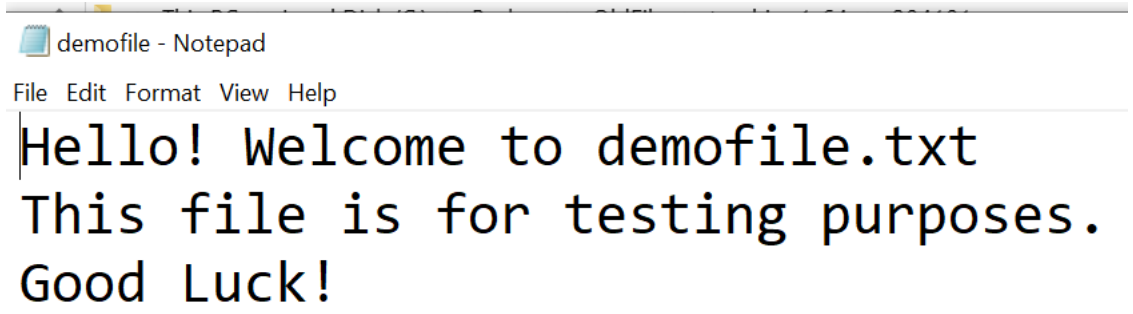
- เช่น f = open(“demofile.txt”, “r”) จะเป็นการเปิดไฟล์ชื่อ demofile.txt เพื่ออ่าน

# การอ่านไฟล์

- method read() ใช้สำหรับอ่านข้อมูล (content) ที่อยู่ในไฟล์
- เช่น เมื่อพิมพ์คำสั่ง 2 คำสั่งคือ

```
f = open("demotext.txt", "r")

print(f.read())
```




ข้อมูลที่เก็บในไฟล์ชื่อ demotext.txt

```
====== RESTART: C:\Backup\OldFile\te
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
>>>
```

ผลลัพธ์ที่ได้เมื่อรัน 2 คำสั่งข้างต้น

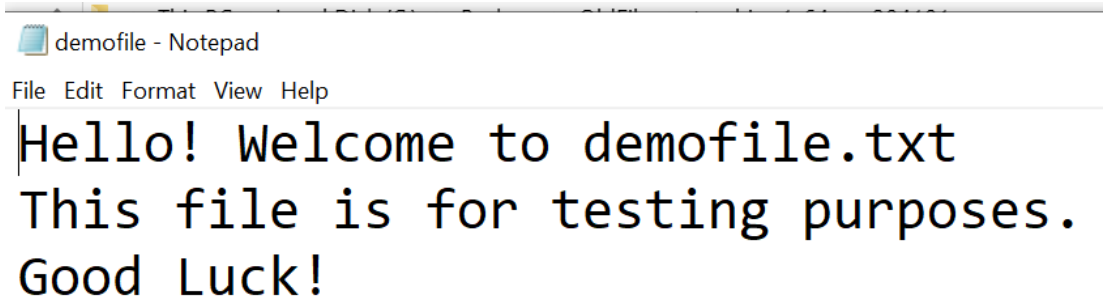
# การอ่านไฟล์ ต่อ

- โดยปกติ method read() จะอ่านข้อมูลทั้งหมดที่มีอยู่ในไฟล์ ใช้ร่วมกับ mode r
- ทั้งนี้เราสามารถอ่านข้อมูลในไฟล์ได้ตามจำนวนตัวอักขระ (character) ที่ระบุด้วย method read(int) โดย int คือจำนวนตัวอักขระที่ต้องการอ่านจากไฟล์
- เช่น เมื่อพิมพ์คำสั่ง 2 คำสั่งคือ

```
f = open("demotext.txt", "r")

print(f.read(5))
```

# การอ่านไฟล์ ต่อ




ข้อมูลที่เก็บในไฟล์ชื่อ demotext.txt

```
>>>
======= RESTART: C:\Backup
Hello
>>>
```

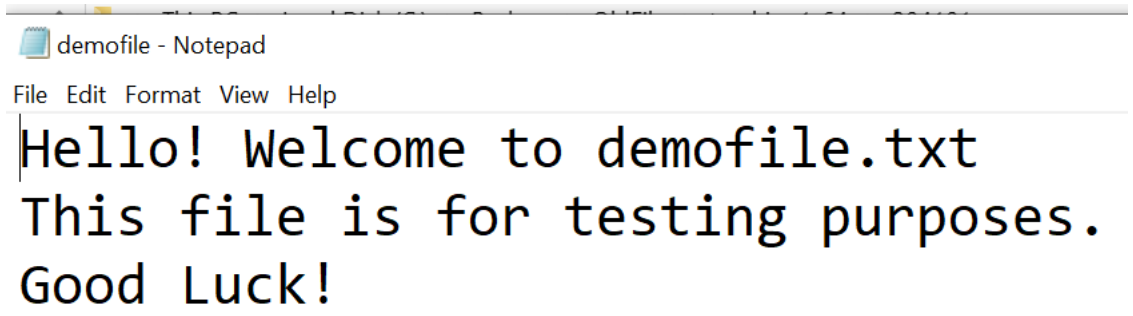
ผลลัพธ์ที่ได้เมื่อรัน 2 คำสั่งในสไลด์ก่อนหน้า

# การอ่านไฟล์ ต่อ

- นอกจากนี้เรายังสามารถอ่านข้อมูลในไฟล์ทีละบรรทัดได้ด้วย method readline()
- เช่น เมื่อพิมพ์คำสั่ง 2 คำสั่งคือ

```
f = open("demotext.txt", "r")

print(f.readline())
```




ข้อมูลที่เก็บในไฟล์ชื่อ demotext.txt

```
>>>
====== RESTART: C:\Backup\OldFil
Hello! Welcome to demofile.txt

>>>
```

ผลลัพธ์ที่ได้เมื่อรัน 2 คำสั่งข้างต้น

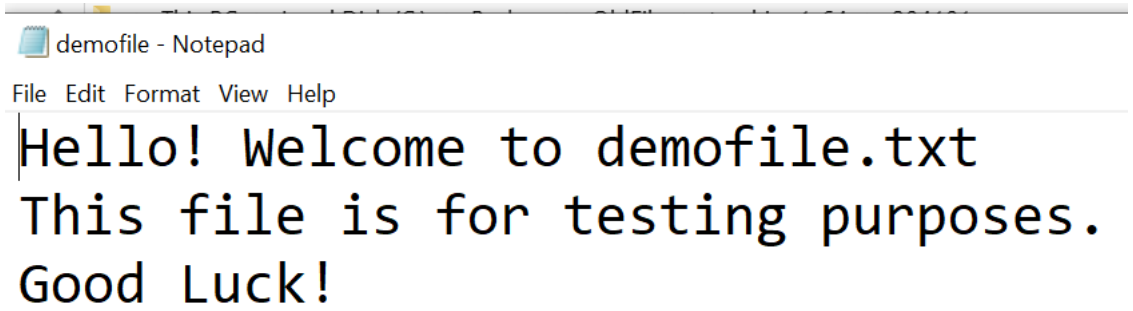
# การอ่านไฟล์ ต่อ

- นอกจากนี้เรายังสามารถอ่านข้อมูลทั้งหมดในไฟล์ด้วยลูป for ได้ด้วย
- เช่น เมื่อพิมพ์คำสั่ง 3 คำสั่งคือ

```
f = open("demotext.txt", "r")

for x in f:

    print(x)
```




ข้อมูลที่เก็บในไฟล์ชื่อ demotext.txt

```
>>>
====== RESTART: C:\Backup\OldFile\teac
Hello! Welcome to demofile.txt

This file is for testing purposes.

Good Luck!
>>>
```

ผลลัพธ์ที่ได้เมื่อรัน 3 คำสั่งข้างต้น

# การปิดไฟล์

- โดยปกติ method close() จะปิดไฟล์ที่เราใช้งานเสร็จ
- ถึงแม้ว่าผลลัพธ์ที่ได้จาก method close() จะไม่แสดงให้เห็นทางจอภาพ แต่เป็นเรื่องที่ดีที่เราควรฝึกหัดไว้ว่าเราควรปิดไฟล์เสมอ เมื่อเราทำงานกับไฟล์นั้นเสร็จแล้ว
- ตัวอย่างการใช้งาน method close() เช่น

```
f = open(“demotext.txt”, “r”)

print(f.readline())

f.close()
```

# การเขียนต่อท้ายไฟล์

- method write() จะเขียนสตริงที่ระบุ ลงไปในไฟล์ที่ระบุชื่อไว้ใน filename ของฟังก์ชัน open() ใช้งานร่วมกับ mode a และ mode w
- เช่น f = open("demofile2.txt", "a")

```
f.write("Now the file has more content!")

#open and read the file after the appending:

f = open("demofile2.txt", "r")

print(f.read())

f.close()
```

# การเขียนต่อท้ายไฟล์ ต่อ

- ทั้งนี้หากเรายัง “**ไม่มี**” ไฟล์ชื่อ demofile2.txt อยู่ คำสั่งในสไลด์ก่อนหน้า จะสร้างไฟล์ชื่อ demofile2.txt ขึ้นมาใหม่ แล้วเขียนสตริงที่กำหนดลงไป

```
>>>
====== RESTART: C:\Backup\OldFile\tea
Now the file has more content!
>>>
```

ผลลัพธ์ที่ได้เมื่อรันคำสั่งในสไลด์ก่อนหน้า

# การเขียนต่อท้ายไฟล์ ต่อ

- แต่หากเรายัง “**มี**” ไฟล์ชื่อ demofile2.txt อยู่แล้ว คำสั่งในสไลด์ก่อนหน้า จะเขียนสตริงที่กำหนดต่อท้ายข้อความที่มีอยู่ในไฟล์ demofile2.txt
- เช่น หากเรารันคำสั่งในสไลด์หน้า 13 อีกครั้ง

```
>>>
====== RESTART: C:\Backup\OldFile\teaching1_64\204101\file ma
Now the file has more content!Now the file has more content!
>>>
```

ผลลัพธ์ที่ได้

# การเขียน (ทับ) ไฟล์

- ลองเปลี่ยน method write() มาใช้กับ mode w บ้าง
- เช่น f = open("demofile3.txt", “w")

```
f.write("Woops! I have deleted the content!")

#open and read the file after the appending:

f = open("demofile3.txt", "r")

print(f.read())

f.close()
```

# การเขียน (ทับ) ไฟล์ ต่อ

- ทั้งนี้หากเรายัง “**ไม่มี**” ไฟล์ชื่อ demofile3.txt อยู่ คำสั่งในสไลด์ก่อนหน้า จะสร้างไฟล์ชื่อ demofile3.txt ขึ้นมาใหม่ แล้วเขียนสตริงที่กำหนดลงไป

```
>>>
====== RESTART: C:\Backup\OldFile\tea
Woops! I have deleted the content!
>>>
```

ผลลัพธ์ที่ได้เมื่อรันคำสั่งในสไลด์ก่อนหน้า

- ข้อควรระวัง!!! หาก “**มี**” ไฟล์ชื่อ demofile3.txt อยู่แล้ว คำสั่งในสไลด์ก่อนหน้า จะ “เขียนสตริงที่ระบุทับ” ลงไปในข้อความเดิม

# การสร้างไฟล์ใหม่

- ลองเปลี่ยน method write() มาใช้กับ mode x บ้าง
- เช่น f = open("myfile.txt", "x")

```
f.write(“Content of file from x mode!")

#open and read the file after the appending:

f = open(" myfile.txt", "r")

print(f.read())

f.close()
```

# การสร้างไฟล์ใหม่ ต่อ

- ทั้งนี้หากเรายัง “**ไม่มี**” ไฟล์ชื่อ myfile.txt อยู่ คำสั่งในสไลด์ก่อนหน้า จะสร้างไฟล์ชื่อ myfile.txt ขึ้นมาใหม่ แล้วเขียนสตริงที่กำหนดลงไป

```
>>>
====== RESTART: C:\Backup\OldFile\tea
Content of file from x mode!
>>>
```

ผลลัพธ์ที่ได้เมื่อรันคำสั่งในสไลด์ก่อนหน้า

# การสร้างไฟล์ใหม่ ต่อ

- แต่หากเรา “**มี**” ไฟล์ชื่อ myfile.txt อยู่แล้ว คำสั่งในสไลด์ 18 จะแสดงข้อความผิดพลาดขึ้น

```
>>>
====== RESTART: C:\Backup\OldFile\teaching1_64\204101\fi
Traceback (most recent call last):
  File "C:\Backup\OldFile\teaching1_64\204101\file manip
 <module>
    f = open("myfile.txt", "x")
FileExistsError: [Errno 17] File exists: 'myfile.txt'
>>>
```

ผลลัพธ์ที่ได้

# การลบไฟล์

- การลบไฟล์ เราต้อง import module OS ก่อน หลังจากนั้นใช้ฟังก์ชัน os.remove()
- เช่น import os
- os.remove("myfile.txt")
- ผลลัพธ์ที่ได้ จะเป็นการลบไฟล์ชื่อ myfile.txt

# การตรวจสอบว่ามีไฟล์ชื่อตามระบุอยู่หรือไม่

- เพื่อเป็นการป้องกันการผิดพลาด การทำงานกับไฟล์ควรตรวจสอบด้วยว่ามีไฟล์ที่เราต้องการทำงานด้วยหรือไม่
- เช่น import os

```
if os.path.exists("myfile.txt"):
        os.remove("myfile.txt")
else:
        print("The file does not exist")
```

# การตรวจสอบว่ามีไฟล์ชื่อตามระบุอยู่หรือไม่ ต่อ

- ทั้งนี้หากเรา “**มี**” ไฟล์ชื่อ myfile.txt อยู่แล้ว คำสั่งจากสไลด์ก่อนหน้าจะลบไฟล์ชื่อ myfile.txt
- แต่หากยัง “**ไม่มี**” ไฟล์ชื่อ myfile.txt อยู่ คำสั่งในสไลด์ก่อนหน้า จะได้ผลลัพธ์

```
>>>
====== RESTART: C:\Backup\OldFile\teac
The file does not exist
>>>
```

ผลลัพธ์ที่ได้เมื่อรันคำสั่งในสไลด์ก่อนหน้า

# References

- https://www.w3schools.com/python/python_file_handling.asp