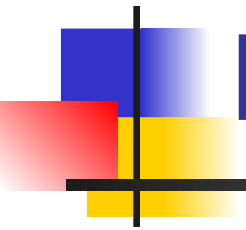
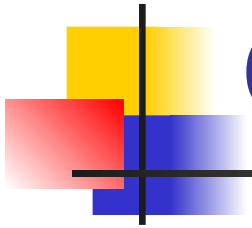


Python Coding part3

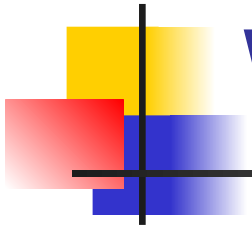
Dr. Bernard Chen

Troy University



Overall Outline

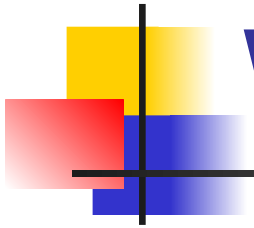
- While Loop
- File I/O



While loop

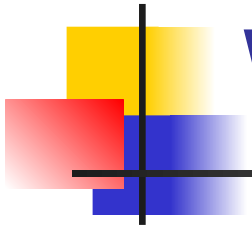
- A **while loop** or **indefinite loop** is a set of instructions that is repeated as long as the associated logical expression is true.

```
while <logical expression>:
    # Code block to be repeated until logical statement is false
    code block
```



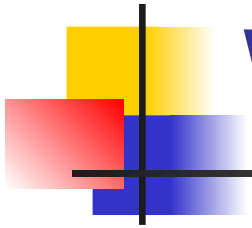
While loop

- When Python reaches a while loop block, it first determines if the logical expression of the while loop is **true** or **false**.
- If the expression is **true**, the code block will be executed, and after it is executed, the program returns to the logical expression at the beginning of the while statement.
- If it is **false**, then the while loop will terminate.



While loop

```
i = 0
n = 8

while n>=1:
    n=n/2
    i+=1
    print (i,n)
```



While loop

- How about this code:

```
n = 0
while n > -1:
    n += 1
```

While loop

EXAMPLE: Which while-loop causes an infinite loop?

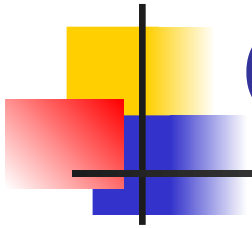
```
# Example 1
n = 1
while n > 0:
    n /= 2
```

```
# Example 2
n = 2
while n > 0:
    if n % 2 == 0:
        n += 1
    else:
        n -= 1
```

Class Practice: Pyramid

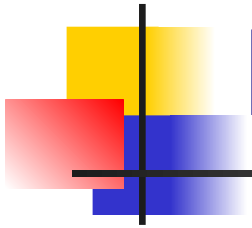
- Now use **while** loop to print the Pyramid looks like: (User can input the height of the pyramid)

```
*
***
*****
*******
*********
***********
*************
```



Overall Outline

- While Loop
- File I/O



File I/O

- Storing data and the results of your programming efforts is important for working over multiple sessions and sharing your results with collaborators.

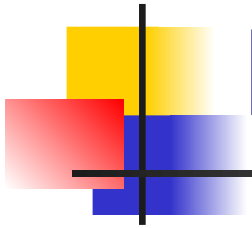
- So far, we used *print* function to display the data to the screen. But there are many ways to store data onto your disk and share it with other program or colleagues.

File I/O

- A **text** file, many times with an extension **.txt**, is a file containing only plain text.

- To work with text files, we need to use *open* function which returns a *file object*. It is commonly used with two

```
f = open(filename, mode)
```



File I/O

- ‘r’, this is the default mode, which opens a file for reading
- ‘w’, this mode opens a file for writing, if the file does not exist, it creates a new file.

File I/O

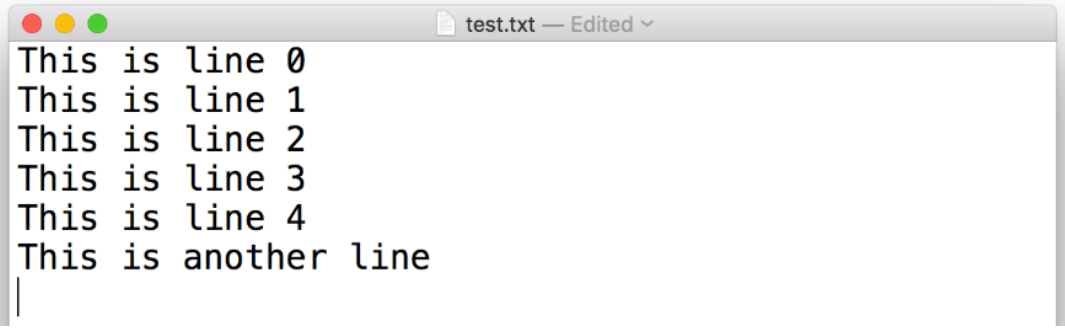
```
file = open('test2.txt','w')
for i in range(10):
    file.write("This is line" + str(i)+'\n')


file.close()
```

File I/O

Append to a file

Now, let's append some string to the *test.txt* file. It is very similar to how we write the file, with only one difference - change the mode to 'a' instead.

```
f = open('test.txt', 'a')
f.write(f"This is another line\n")
f.close()
```



File I/O

```
file = open('test2.txt','w')
for i in range(10):
    file.write("This is line" + str(i)+'\n')


file.close()



f=open('./test2.txt','r')
content=
f.close()
print(content)


input=[]
for line in content:
    input.append(line)
```

File I/O

```
file = open('test2.txt','w')
for i in range(10):
    file.write("This is line" + str(i)+'\n'


file.close()



f=open('./test2.txt','r')
content=f.readlines()
f.close()
print(content)


input=[]
for line in content:
    input.append(line)
```

File I/O

To show full file extensions in Windows (so you can see `.txt`, `.py`, `.xlsx`, etc.):

Windows 11

Method 1 (Quickest)

1. Open **File Explorer**
2. Click **View**
3. Select **Show**
4. Check **File name extensions**

File I/O

```
file=open('min_max.csv','r')
content=file.readlines()
fulldata=[]
for i in range(1,len(content)):
    fulldata.append(content[i].split(','))
```

File I/O

```
file=open('min_max.csv','r')
content=file.readlines()
fulldata=[]
for i in range(1,len(content)):

fulldata.append(content[i].split(','))
```

Min-max normalization

- Min-max normalization: to [new_min$_A$, new_max$_A$]

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

  - Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$

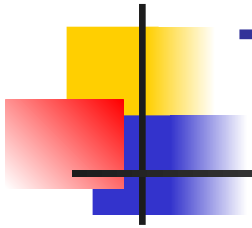
Z-score normalization

- Z-score normalization (μ: mean, σ: standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - Ex. Let μ = 54,000, σ = 16,000. Then

$$\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$$



Take home assignment

- Perform min-max normalization on the first 3 columns in min_max.csv file