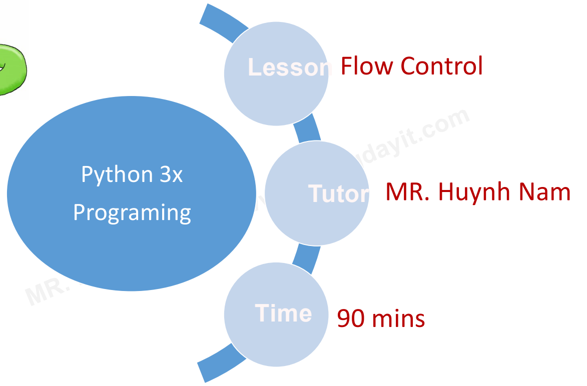


Flow Control

MR. Huynh Nam

90 mins

Content

- Condition Flow
  - Python if Statement Syntax
  - Syntax of if...else
  - Syntax of if...elif...else
- Loop Flow
  - While Loop
  - For Loop
  - Break and Continue
- Python List Comprehension
- Python Directory and Files Management
- Exceptions
  - Python Errors and Built-in Exceptions
  - Python Exception Handling - Try, Except and Finally

Python if Statement Syntax

```
if test expression:
    statement(s)
```

Using **and**, **or** keyword to combine multiple condition in test expression

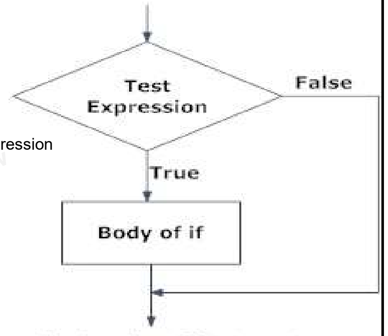
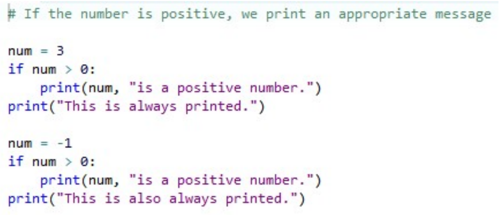
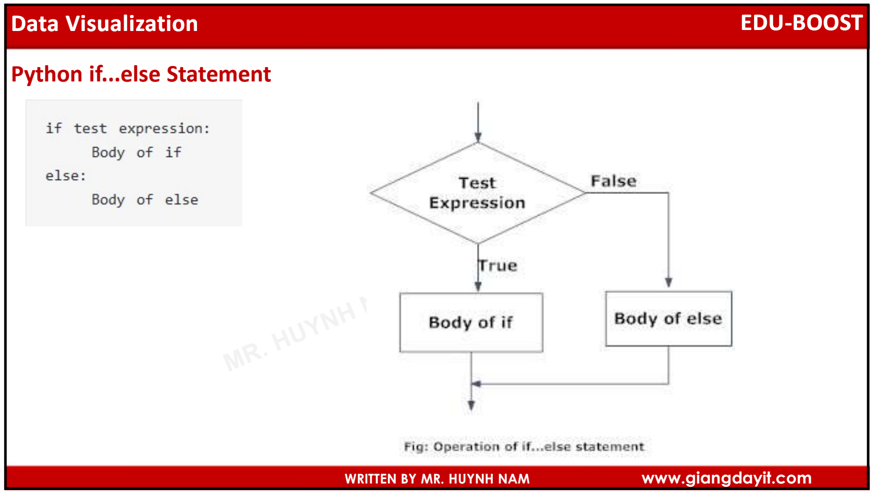


Fig: Operation of if statement

Example



```
# If the number is positive, we print an appropriate message

num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")

num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```



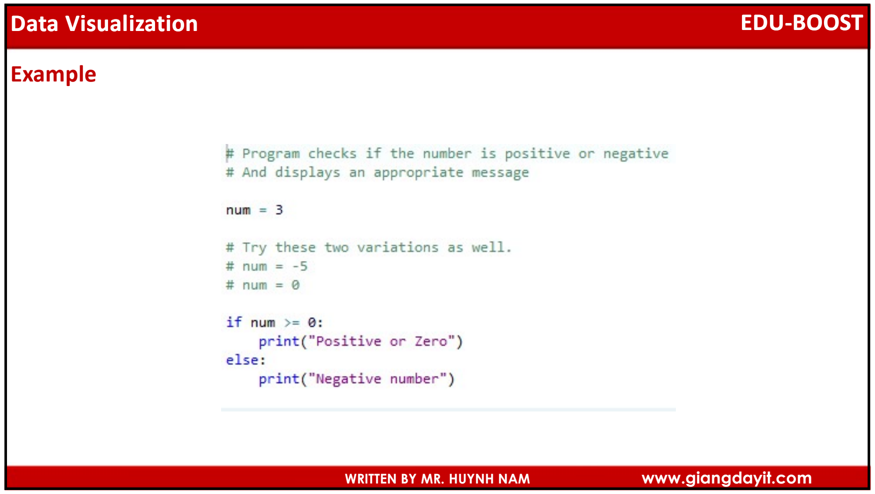
Data Visualization EDU-BOOST

Python if...else Statement

```
if test expression:
    Body of if
else:
    Body of else
```

Fig: Operation of if...else statement

5

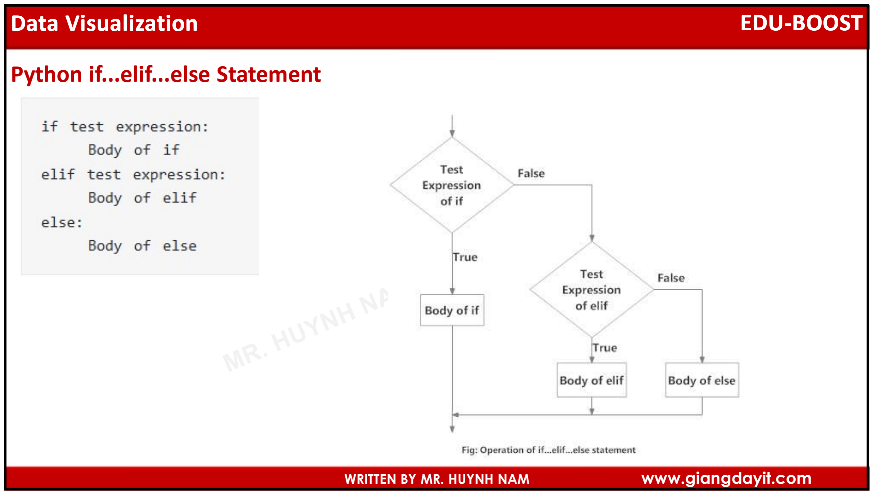


Data Visualization EDU-BOOST

Example

```python
# Program checks if the number is positive or negative
# And displays an appropriate message

num = 3

# Try these two variations as well.
# num = -5
# num = 0

if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

6



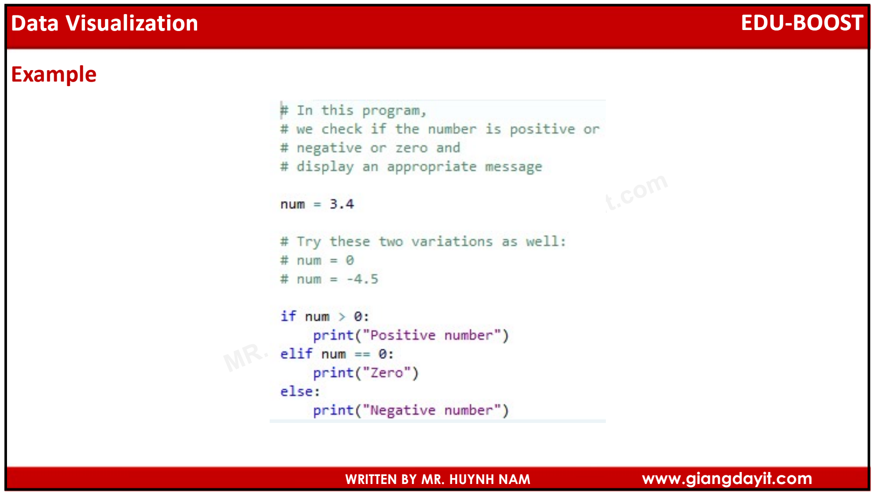
Data Visualization EDU-BOOST

Python if...elif...else Statement

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```

Fig: Operation of if...elif...else statement

7



Data Visualization EDU-BOOST

Example

```python
# In this program,
# we check if the number is positive or
# negative or zero and
# display an appropriate message

num = 3.4

# Try these two variations as well:
# num = 0
# num = -4.5

if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

8

More examples

```
# In this program, we input a number
# check if the number is positive or
# negative or zero and display
# an appropriate message
# This time we use nested if

num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

Output 1

```
Enter a number: 5
Positive number
```

Output 2

```
Enter a number: -1
Negative number
```

Output 3

```
Enter a number: 0
Zero
```

Exercise if | if...else | if...elif...else

Check out these examples to learn more:

- Python Program to Check if a Number is Positive, Negative or 0
- Python Program to Check if a Number is Odd or Even
- Python Program to Check Leap Year

Syntax of while Loop in Python

```
while test_expression:
    Body of while
```

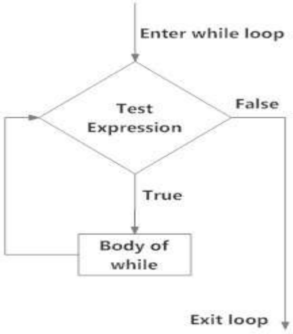


Fig: operation of while loop

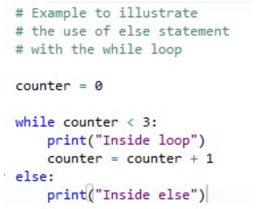
Example

```
# Program to add natural
# numbers upto
# sum = 1+2+3+...+n

# To take input from the user,
# n = int(input("Enter n: "))

n = 10

# initialize sum and counter
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1    # update counter

# print the sum
print("The sum is", sum)
```

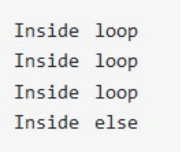
```
Enter n: 10
The sum is 55
```

Data Visualization **EDU-BOOST**

while loop with else



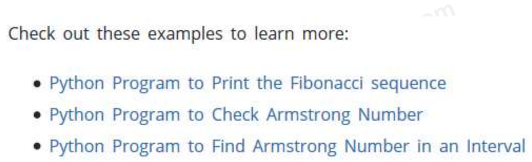
```
# Example to illustrate
# the use of else statement
# with the while loop

counter = 0

while counter < 3:
    print("Inside loop")
    counter = counter + 1
else:
    print("Inside else")
```



```
Inside loop
Inside loop
Inside loop
Inside else
```

Data Visualization **EDU-BOOST**

Exercise while loop

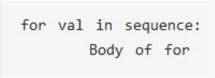


Check out these examples to learn more:

- Python Program to Print the Fibonacci sequence
- Python Program to Check Armstrong Number
- Python Program to Find Armstrong Number in an Interval

Data Visualization **EDU-BOOST**

Syntax of for Loop



```
for val in sequence:
        Body of for
```

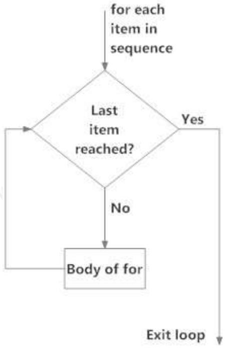
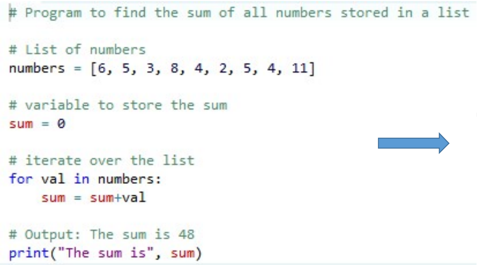


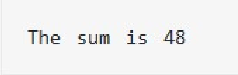
Fig: operation of for loop

Data Visualization **EDU-BOOST**

Example



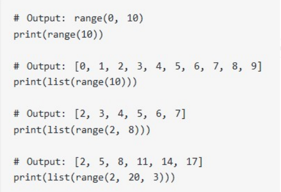
```
# Program to find the sum of all numbers stored in a list

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
    sum = sum+val

# Output: The sum is 48
print("The sum is", sum)
```



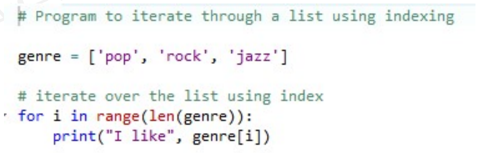
```
The sum is 48
```

The range() function

- Generate a sequence of numbers using range() function.
  - Example: range(10) will generate numbers from 0 to 9 (10 numbers).
- Also define the start, stop and step size as range(start,stop,step size).
  - Step size defaults to 1 if not provided.
- Function does not store all the values in memory, it would be inefficient. So it remembers the start, stop, step size and generates the next number on the go
  - To force this function to output all the items, we can use the function list()

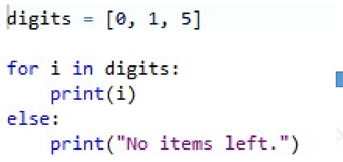


```
# Output: range(0, 10)
print(range(10))

# Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(list(range(10)))

# Output: [2, 3, 4, 5, 6, 7]
print(list(range(2, 8)))

# Output: [2, 5, 8, 11, 14, 17]
print(list(range(2, 20, 3)))
```

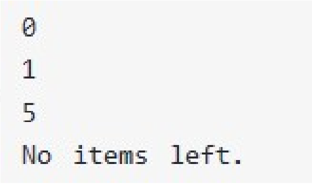


```
# Program to iterate through a list using indexing

genre = ['pop', 'rock', 'jazz']

# iterate over the list using index
for i in range(len(genre)):
    print("I like", genre[i])
```

for loop with else

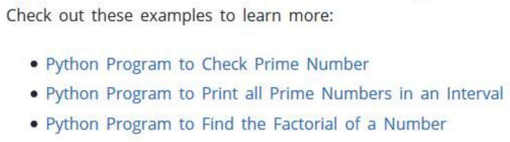


```
digits = [0, 1, 5]

for i in digits:
    print(i)
else:
    print("No items left.")
```



```
0
1
5
No items left.
```

Exercise for loop

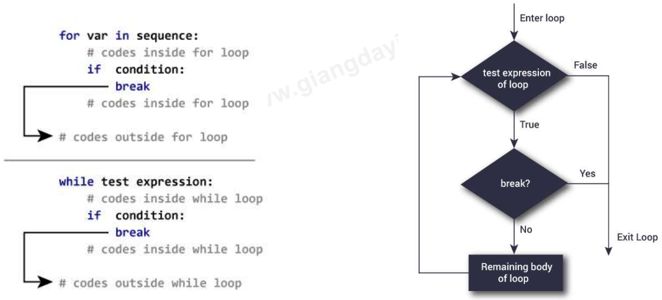


Check out these examples to learn more:

- Python Program to Check Prime Number
- Python Program to Print all Prime Numbers in an Interval
- Python Program to Find the Factorial of a Number

Python break and continue

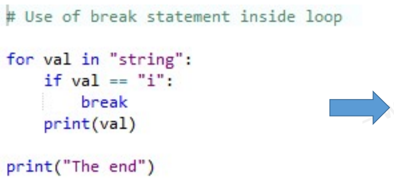
- Break and continue statements can alter the flow of a normal loop
  - Loops iterate over a block of code until test expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without checking test expressi--



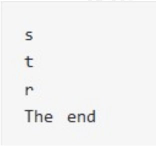
```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```

Example

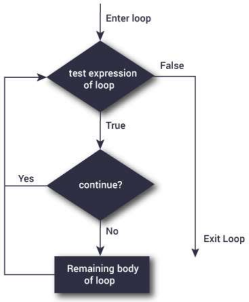


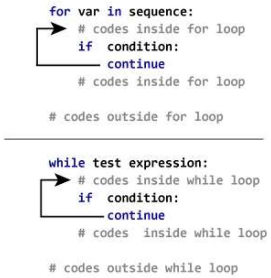
```
# Use of break statement inside loop

for val in "string":
    if val == "i":
        break
    print(val)

print("The end")
```



```
s
t
r
The end
```

Syntax of Continue

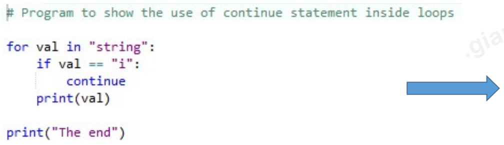




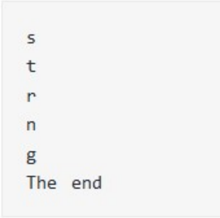
```
for var in sequence:
    # codes inside for loop
    if condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if condition:
        continue
    # codes  inside while loop

# codes outside while loop
```

Example

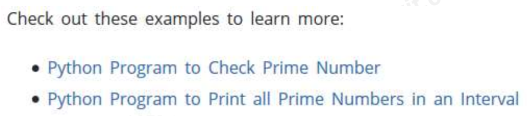


```
# Program to show the use of continue statement inside loops

for val in "string":
    if val == "i":
        continue
    print(val)

print("The end")
```



```
s
t
r
n
g
The end
```

Exercise break & continue



Check out these examples to learn more:

- Python Program to Check Prime Number
- Python Program to Print all Prime Numbers in an Interval

Overview

- Condition If…else

```
if <expression>:
    <statements>
elif <expression>:
    <statements>
else:
    <statements>
```

- Loop while | for

```
while <expression>:
    <statements>
else:
    <statements>
```

```
for <name> in
<container>:
    <statements>
else:
    <statements>
```

- Commands: break, continue, pass

25

Programing Fundamentals with Python

- Example

REMEMBER Indentation

```
x = 149
y = 100
if x > y:
    print(x,"is greater than",y)
else:
    print(x,"is less than", y)
```

```
149 is greater than 100
```

```
x = 0
while x < 10:
    print(x),
    x+=1
```

```
a = [‘bo’, ‘me’, ‘con’]
for x in a:
        print(x),
```

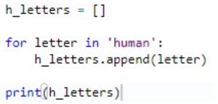
```
0 1 2 3 4 5 6 7 8 9
```

```
bo me con
```

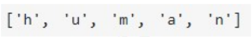
26

Python List Comprehension

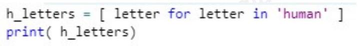
- Iterating through a string Using for Loop



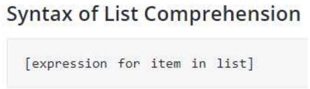
```
h_letters = []

for letter in 'human':
    h_letters.append(letter)

print(h_letters)
```



```
['h', 'u', 'm', 'a', 'n']
```



```
h_letters = [ letter for letter in 'human' ]
print( h_letters)
```



Syntax of List Comprehension

```
[expression for item in list]
```

[expression for item in list]

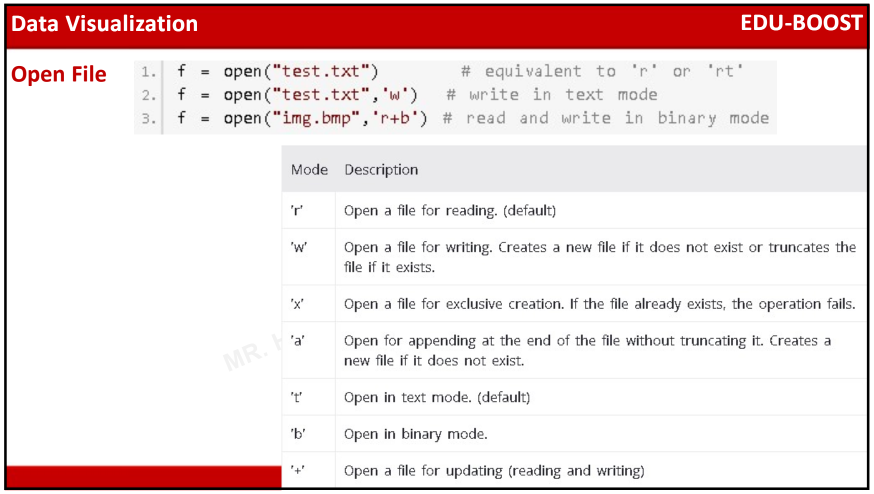
[letter for letter in 'human']

27

Python File I/O

- File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk).
- A file operation takes place in the following order.
    1. Open a file
    2. Read or write (perform operation)
    3. Close the file

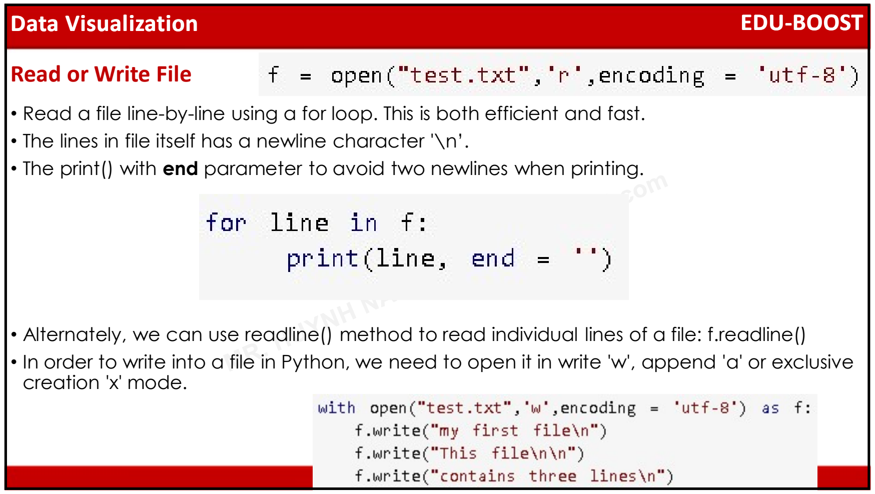
28



Open File

```
f = open("test.txt")      # equivalent to 'r' or 'rt'
f = open("test.txt",'w')  # write in text mode
f = open("img.bmp",'r+b') # read and write in binary mode
```

Mode	Description
'r'	Open a file for reading. (default)
'w'	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
'x'	Open a file for exclusive creation. If the file already exists, the operation fails.
'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode. (default)
'b'	Open in binary mode.
'+'	Open a file for updating (reading and writing)



Read or Write File

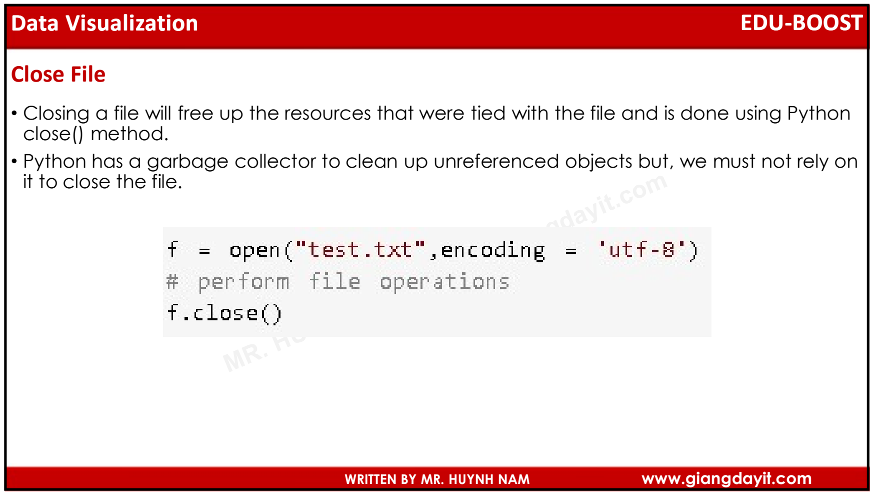
```
f = open("test.txt",'r',encoding = 'utf-8')
```

- Read a file line-by-line using a for loop. This is both efficient and fast.
- The lines in file itself has a newline character '\n'.
- The print() with **end** parameter to avoid two newlines when printing.

```
for line in f:
    print(line, end = '')
```

- Alternately, we can use readline() method to read individual lines of a file: f.readline()
- In order to write into a file in Python, we need to open it in write 'w', append 'a' or exclusive creation 'x' mode.

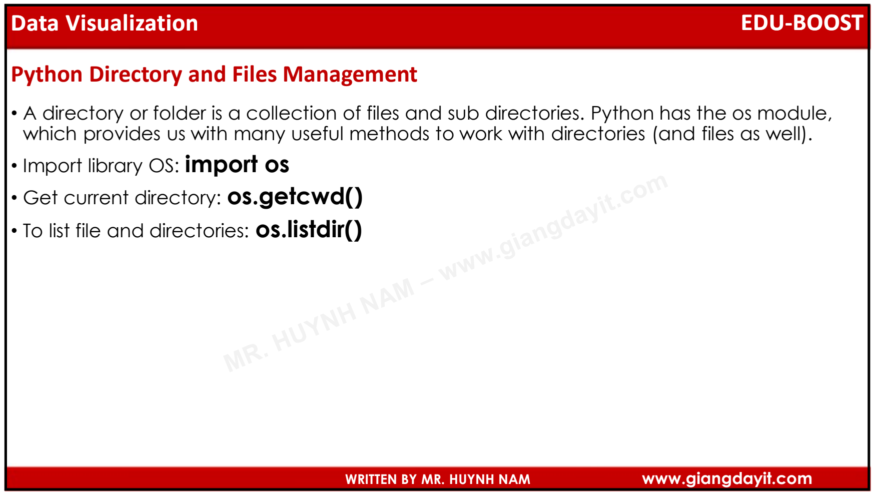
```
with open("test.txt",'w',encoding = 'utf-8') as f:
   f.write("my first file\n")
   f.write("This file\n\n")
   f.write("contains three lines\n")
```



Close File

- Closing a file will free up the resources that were tied with the file and is done using Python close() method.
- Python has a garbage collector to clean up unreferenced objects but, we must not rely on it to close the file.

```
f = open("test.txt",encoding = 'utf-8')
# perform file operations
f.close()
```

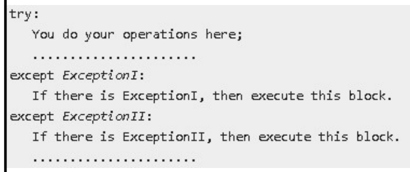


Python Directory and Files Management

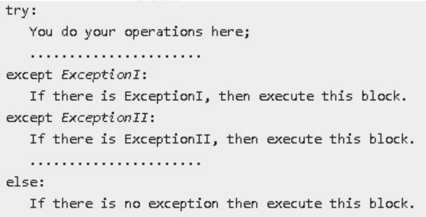
- A directory or folder is a collection of files and sub directories. Python has the os module, which provides us with many useful methods to work with directories (and files as well).
- Import library OS: **import os**
- Get current directory: **os.getcwd()**
- To list file and directories: **os.listdir()**

Python Exception Handling - Try, Except and Finally

- Python has many built-in exceptions which forces your program to output an error when something in it goes wrong.
- When these exceptions occur, it causes the current process to stop and passes it to the calling process until it is handled. If not handled, our program will crash.



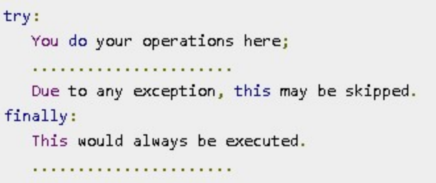
```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
```



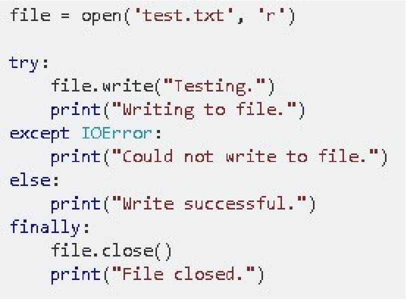
```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

Python Exception Handling - Try, Except and Finally

- You can use a finally: block along with a try: block. The finally block is a place to put any code that must execute, whether the try-block raised an exception or not. The syntax of the try-finally statement is this

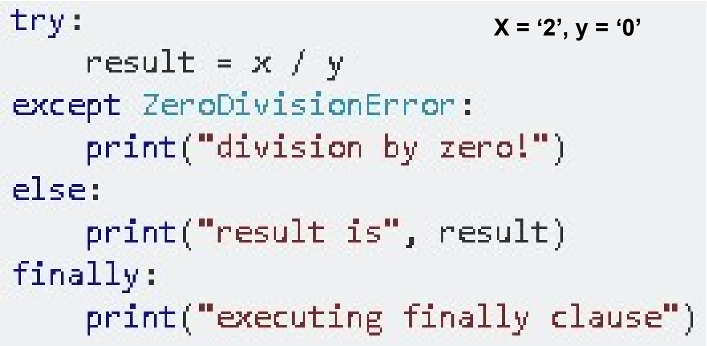


```
try:
   You do your operations here;
   ......................
   Due to any exception, this may be skipped.
finally:
   This would always be executed.
   ......................
```



```
file = open('test.txt', 'r')

try:
    file.write("Testing.")
    print("Writing to file.")
except IOError:
    print("Could not write to file.")
else:
    print("Write successful.")
finally:
    file.close()
    print("File closed.")
```

Python Exception Handling - Try, Except and Finally



```
try:
    result = x / y
except ZeroDivisionError:
    print("division by zero!")
else:
    print("result is", result)
finally:
    print("executing finally clause")
```

X = ‘2’, y = ‘0’

THANK YOU

Q & A