Python Programming

COMP 8347

Usama Mir

usamamir@uwindsor.ca

Python Basics

- Topics:
  - If statements
  - Loops
  - Exception handling
  - Functions
  - File Input/Output
  - Modules

Control Flow

- Conditional branching
  - if statements
- Looping
  - while
  - for … in
- Exception handling
- Function or method call

if Statements

- *suite*: a block of code, i.e. a sequence of one or more statements
- Syntax:

```
if bool_expression1:
    suite1
elif bool_expression2:
suite2
…
elif bool_expressionN:
suiteN
else:
else_suite
```

- No parenthesis or braces
- Use *indentation* for block structure

```
if a < 10:
    print("few")
elif a < 25:
    print("some")
else:
    print("many")
```

Loops = What is it?

- Clear starting condition
- Clear finishing condition
- Some statement that leads the loops from its start to the end

Loops: Python vs. Java

- Python:
  - while loops
  - for loops …. for i in range (5)
- Java:
  - while loops
  - do .. while
  - for loops (int i = 0…..)

while Statements

- Used to execute a suite 0 or more times
  - number of times depends on while loop's Boolean expression.
  - Syntax:

    while bool_expression:

    suite
  - Example:

```
x, sum = 0,0
    while x < 10:
        sum += x
        x += 2
    print(sum, x)   #What is final value of sum and x
```

Answer: sum=20, x=10

for … in Statements

- Syntax:

  for variable in iterable:

  suite

- Example: fruits = [‘apple’, ‘pear’, ‘plum’, ‘peach’]

```
for item in fruits:
    print(item)
```

- Alternatively

```
for i in range(len(fruits)):
    print(fruits[i])
```

for … in Statements - Enumerate

```
#####for loop for a list of days
days = ["mon", "tue", "wed", "thu", "friday", "sat", "sun"]
for i,d in enumerate(days):
    print(i,d)
```

Output

```
0 mon
1 tue
2 wed
3 thu
4 friday
5 sat
6 sun
```

Break and Continue

ex2.py - C:\Users\Arunita\OneDrive - University of Windsor\8347\slidesF20\ex2.py (3.8.3)

File Edit Format Run Options Window Help

```python
# Example using break and continue
# Continue: control returns to top of current loop
# Break: exit from current loop

for num in [11, 8, 3, 25, 9, 16]:
    if num > 20:
        print('exiting loop')
        break   # exit the loop completely
    elif num%2 == 0:
        continue # immediately start next iteration
    print(num)

```

Ln: 7 Col: 28

Iter#	num	num>20?	Num%2==0?	print(num)
0	11	n	n	11
1	8	n	y	
2	3	n	n	3
3	25	y		

Exception Handling

- Functions or methods indicate errors or other important events by ***raising exceptions***.
- Syntax (simplified):

```
try:
    try_suite
except exception1 as variable1:
    exception_suite1
…
except exceptionN as variableN
finally:
    # cleanup
```

- variable part is optional

Exception Handling – Example 1

Example:

```
s = input('Enter number: ')
try:
    n = float(s)
    print(n, ' is valid. ')
except ValueError as err:
    print(err)
```

- If user enters ‘8.6’ output is: 8.6 is valid
- If user enters ‘abc’, output is:
  ValueError: could not convert string to float: ‘abc'

Exception Handling – Class Exercise

- Part 1
  - Create a list with the following values 5, 10, 30, 40, 9.9
  - Run a for loop till the size of the list
  - Take a variable item and store each index of the list as a power of two. E.g., 1st value in item is 0, then 1, then 4, then 9, and so on
  - Print the index and the item. This program will give error for the index values which are out of the above list's range
- Part 2
  - For the above program, use try and except statements to print the index error when the index is out of bound/range. Your output should look like this:

```
index= 0 item 5
index= 1 item 10
index= 4 item 9.9
list index out of range
9 is out of range
list index out of range
16 is out of range
```

Exception Handling – Class Exercise – Part 1

exceptionhandling.py - C:/Users/manch/OneDrive/Desktop/U of Windsor/Comp 8347/New/All slides/Usama Slides/2/exceptionhandling.py (3.10.1)

File Edit Format Run Options Window Help

```
mylist = [5, 10, 15, 20, 9.9]
for i in range (5):
    item = mylist[i**2]
    print("index=", i**2, "item", item)
```

IDLE Shell 3.10.1

File Edit Shell Debug Options Window Help

```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/manch/OneDrive/Desktop/U of Windsor/Comp 8347/New/All slides/Usama Slides/2/exceptionhandling.py
index= 0 item 5
index= 1 item 10
index= 4 item 9.9
Traceback (most recent call last):
  File "C:/Users/manch/OneDrive/Desktop/U of Windsor/Comp 8347/New/All slides/Usama Slides/2/exceptionhandling.py", line 3, in <module>
    item = mylist[i**2]
IndexError: list index out of range
>>>
```

Exception Handling – Class Exercise – Part 2

exceptionhandling.py - C:/Users/manch/OneDrive/Desktop/U of Windsor/Comp 8347/New/All slides/Usama Slides/2/exceptionhandling.py (3.10.1)

File Edit Format Run Options Window Help

```
mylist = [5, 10, 15, 20, 9.9]
for i in range (5):

    try:
        item = mylist[i**2]
        print("index=", i**2, "item", item)

    except IndexError as err:
        print(err)
        print(i**2, "is out of range")
```

```
================================
index= 0 item 5
index= 1 item 10
index= 4 item 9.9
list index out of range
9 is out of range
list index out of range
16 is out of range
>>>
```

Functions

- Function definition:

A block of reusable code that is used to perform a single action

Ex. A code without a function →

function example.py - C:\Users\manch\OneDrive\Desktop\function example.py (3.10.1)

File Edit Format Run Options Window Help

```
#other code here

name = input("enter your name:")
time = input("enter the preferred time")
print("Good" + time + "," + name)

#other code here

name = input("enter your name:")
time = input("enter the preferred time")
print("Good" + time + "," + name)
```

Functions – Example 1

- **Same code implemented with a function**

```
#function starts here

def greet ():
    name = input("enter your name:")
    time = input("enter the preferred time")
    print("Good" + time + "," + name)

greet()
#other code here

greet()
#other code here
```

Functions – Example 2

```
# function with two arguments
def add_numbers(num1, num2):
        sum = num1 + num2
       print("Sum: ",sum)

# function call with two values
add_numbers(5, 4)

# Output: Sum: 9
```

Functions - Example 3 - With Variable Arguments

```
# *args for variable number of arguments
def myFun(*argv):
    for arg in argv:
        print(arg)


myFun('Hello', 'Welcome', 'to', 'GeeksforGeeks')
```

Output:

```
Hello
Welcome
to
GeeksforGeeks
```

File Input/Output

- Open a file
  - f = open(filename, mode)
  - mode is optional; possible values:
    - ‘w’ = write
    - ‘r’ = read (default)
    - ‘a’ = append
    - ‘rb’(‘wb’) = read (write) in binary
  - Example:
    - f = open(“text.txt”)

File Methods

- Read data from a file
  print(f.read())

- Writing data on a file
  - f = open(“test.txt”, ‘w’)
  - f.write(“Hello Python”)

- *f.read(n)*: reads at most n bytes from f
- *f.readline()*: reads only one line
- *f.readlines()*: reads all the lines to the end of file and return them as a list
- *f.close()*: closes a file and free up the resources

Use strip() Function

- Python's strip( ) function is used to remove unwanted characters from a string.

Ex.

```
str5 = '++++++Python Tutorial****** $$$$$'
print ("\n Given string is = ", str5)
str6 = str5. strip ( ' $*+' )
print (" Stripping the '+', '*' and '$' symbols on both sides of the string is = ", str6)
```

Output:

```
Given string is =  ++++++Python Tutorial****** $$$$$
Stripping the '+', '*' and '$' symbols on both sides of the string is =  Python Tutorial
```

Modules

- *Modules*: Contain additional functions and custom data types.
  - Ex. import calendar
- Other examples:
  - import os
  - import math
  - import datetime

Module Example – Calendar

```
# import module
import calendar

yy = 2017
mm = 11

# display the calendar
print(calendar.month(yy, mm))
```

Output:

```
    November 2017
Mo Tu We Th Fr Sa Su
        1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

References

- Slides from Dr. Jaekel and Dr. Saja
- https://www.softwaretestinghelp.com/python/input-output-python-files/
- https://www.tutorialspoint.com/difference-between-for-loop-and-while-loop-in-python
- Programming in Python 3 A complete introduction to the python language (2nd Ed) by Mark Summerfield. Addison Wesley 2010.
- https://www.w3schools.com/python/
- https://www.tutorialsteacher.com/python/error-types-in-python
- https://www.geeksforgeeks.org/python-functions/
- https://www.geeksforgeeks.org/python-calendar-module/