

Persistence & Exception





Learning Outcome

- Explain the concepts of persistence
- Explain the concepts of exception handling
- Implement programs that persist data and handle exceptions

Exceptions

Error that occurs while a program is running

What are the potential exceptions?

Enter a number: 3
Enter another number: 0

Exceptions

```
def main():
    num1 = int(input('Enter a number: '))
    num2 = int(input('Enter another number: '))
    result = num1 / num2
    print('{} divided by {} is {}'.format(num1, num2, result))

main()
```

ZeroDivisionError

```
Traceback (most recent call last):
  File "c:\python_examples\exceptions\zero_division_test.py", line 7, in <module>
    main()
  File "c:\python_examples\exceptions\zero_division_test.py", line 4, in main
    result = num1 / num2
ZeroDivisionError: division by zero
```

What are the potential exceptions?

Enter a number: a

Exceptions

```
def main():
    num1 = int(input('Enter a number: '))
    num2 = int(input('Enter another number: '))
    result = num1 / num2
    print('{} divided by {} is {}'.format(num1, num2, result))

main()
```

```
Traceback (most recent call last):
  File "c:\python_examples\exceptions\value_error_test.py", line 7, in <module>
    main()
  File "c:\python_examples\exceptions\value_error_test.py", line 2, in main
    num1 = int(input('Enter a number: '))
ValueError: invalid literal for int() with base 10: 'a'
```

ValueError

Using Exception Handler

```
def main():
    try:
        num1 = int(input('Enter a number: '))
        num2 = int(input('Enter another number: '))
        result = num1 / num2
        print('{} divided by {} is {}'.format(num1, num2, result))
    except ValueError:
        print('Please enter a valid number')
    except ZeroDivisionError:
        print('Second number cannot be 0')

main()
```

Exception Handler
using the try/except
statement to handle
the exceptions
gracefully

Division by zero triggers the ZeroDivisionError exception.

Using Exception Handler

```
def main():
    try:
        num1 = int(input('Enter a number: '))
        num2 = int(input('Enter another number: '))
        result = num1 / num2
        print('{} divided by {} is {}'.format(num1, num2, result))
    except ValueError:
        print('Please enter a valid number')
    except ZeroDivisionError:
        print('Second number cannot be 0')
main()
```

Program jumps to exception handler code that is after the except ZeroDivisionError line.

Using Exception Handler

```
def main():
    try:
        num1 = int(input('Enter a number: '))
        num2 = int(input('Enter another number: '))
        result = num1 / num2
        print('{} divided by {} is {}'.format(num1, num2, result))
    except ValueError:
        print('Please enter a valid number')
    except ZeroDivisionError:
        print('Second number cannot be 0')

main()
```

User enters an invalid number and triggers the ValueError exception.

Program jumps to exception handler code that is after the except ValueError line.

User enters an invalid filename and triggers the IOError exception.

Program jumps to exception handler code that is after the except IOError line.

IOError

```
def main():
    try:
        user_file = open('users.txt', 'r')
        contents = user_file.read()
        print(contents)
        user_file.close()
    except IOError:
        print('An error occurred trying to read')

main()
```



Types of Exception

ValueError

Exception occurred when value cannot be assigned to an object

ZeroDivisionError

Exception occurred when divide by 0

IOError

Exception occurred when program is unable to process the input/output

How to know and catch all exceptions?

Using one except clause to catch all

```
def main():
    try:
        user_file = open('users.txt', 'r')
        contents = user_file.read()
        print(contents)
        user_file.close()
    except IOError:
        print('An error occurred trying to read')
    except:
        print('An unknown error has occurred')

main()
```

Use one except clause
to catch all the
exceptions that are
raised in a try suite.

Using the else Clause

```
def main():
    try:
        num1 = int(input('Enter a number: '))
        num2 = int(input('Enter another number: '))
        result = num1 / num2
    except ValueError:
        print('Please enter a valid number')
    except ZeroDivisionError:
        print('Second number cannot be 0')
    else:
        print("{} divided by {} is {}".format(num1, num2, result))
main()
```

The statements in else block are executed after the statements in try block, only if no exceptions is raised.

Using the finally Clause

```
def main():
    try:
        user_file = open('users.txt', 'r')
        contents = user_file.read()
        print(contents)
    except IOError:
        print('An error occurred trying to read')
    finally:
        user_file.close()

main()
```

The finally clause must appear after all the except clauses and it will always be executed after the try or exceptions handlers have executed

The Phone Stall Ltd

We Buy & Sell Phones

Phones Unlocked



CASH PAID FOR PHONES

Opening Hours
Mon - Sat: 9.30 - 5pm
Tel: 078868 845226

HTC
BlackBerry
NOKIA
LG
SAMSUNG
MOBILE PHONE REPAIR CENTRE

HTC
BlackBerry
NOKIA
LG
SAMSUNG
MOBILE PHONE

Activity

Practical Question 1

Persistence

Persistent storage



Types of Files



Binary

Data not encoded as text,
not human readable

01001001
00100000
01001100
01001111
01010110
01000101
00100000
01011001
01001111
01010101

Text

Data encoded as text as a
series of characters,
human readable

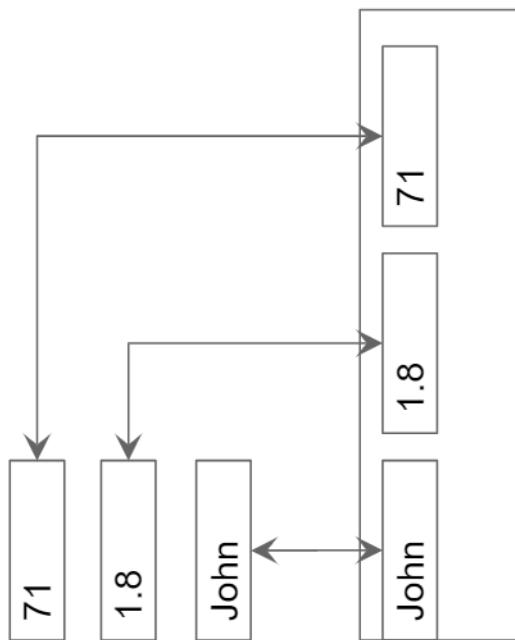


File Input and Output

When a program needs to save data for later use, it writes the data in a file.

The data can then be read from the file at a later time.

weight
height
name





Read Write data

Open the file

Creates a connection between the file and the program

Process the file

Data is either written to or read from the file

Close the file

The file is closed to disconnect the file from the program

File Access Methods

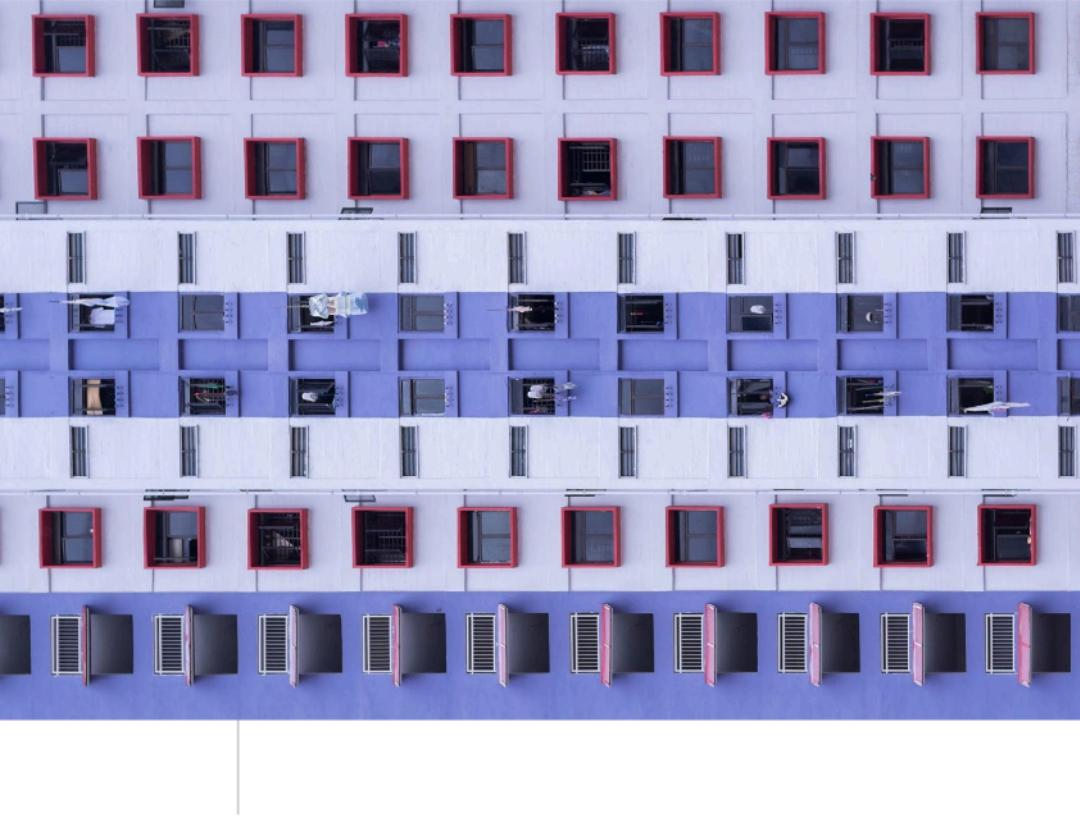


Sequential Access

Access data from the beginning of the file to the end of the file

Direct Access

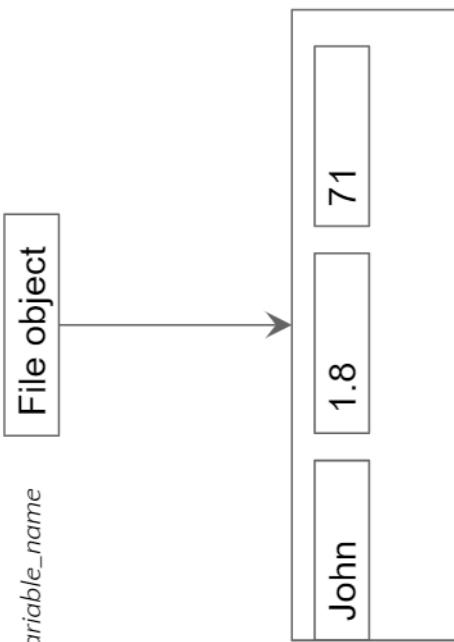
Jump directly to any piece of data in the file without reading data that comes before it



Filenames and File Objects



A file object is an object that is associated with a specific file and provides a way for the program to work with the file



Open a file

```
# open a file for reading only  
user_file = open('users.txt', 'r')
```

Use the **open** function to open a file, given its file name and the operation mode.

```
# open a file for writing only  
sales_file = open('sales.txt', 'w')  
log_file = open('logs.txt', 'a')
```

Use the **r** prefix to specify that the string is a raw string.

```
# specify location of file  
temp_file = open('c:\temp\temp.txt', 'w')
```

'r'

Read mode, file cannot be changed

'w'

Write mode, new file will be created

'a'

Append mode, data will be appended

Write Data to a File

```
# open a file for writing
user_file = open('users.txt', 'w')

# writing a string to the file
user_file.write('John, 1.8, 71')

# close the file
user_file.close()
```

Use the **write** method
to write a string to the
file.

Use the **close** method
to close the file object.

Use the **read** method to read the entire content from the file.

Remember to **close** the file after the operation.

Read Data from a File

```
# open a file for reading
user_file = open('users.txt', 'r')

# Read the file's contents
content = user_file.read()

# close the file
user_file.close()
```

Read Data line by line

```
# open a file for reading
user_file = open('users.txt', 'r')

# Read the file's contents line by line
line1 = user_file.readline() # read John\n
line2 = user_file.readline() # read David\n
line2 = user_file.readline() # read Peter\n

# close the file
user_file.close()
```

Use the **readline** method to read line by line from the file.

The read position will automatically move to the next line after each readline.

John\nDavid\nPeter\n

Read position

True or False

When you open an existing file using the 'w' mode, the contents of the existing file will be erased.



When an input file is opened, its read position is initially set to the first item in the field.



The 'a' mode appends new data to the end of a file. But if the file is nonexistent, it does not create a new one.



If you use readline() to read beyond the end of file, a runtime exception will occur.



To write numbers into a file, the numbers need to be converted to a string before using write() to write into the file.



Call the readline
method just before
entering the while loop
so it can be tested by
the loop for empty
string.

How to detect end of file

```
users_file = open('users.txt', 'r')
line = users_file.readline()
# line read is empty string at end of file
while line != '':
    print(line)
    line = users_file.readline()
users_file.close()
```

Using a for loop that automatically reads line in a file without testing for end of file.

Normally we store the lines read from file into a list for future processing.

Using Python's for loop to read lines

```
lines = []
users_file = open('users.txt', 'r')
for line in users_file:
    lines.append(line)
users_file.close()
```

The Phone Stall Ltd

We Buy & Sell Phones

Phones Unlocked



CASH PAID FOR PHONES

Opening Hours
Mon - Sat: 9.30 - 5pm
Tel: 078868 845226

HTC
BlackBerry
NOKIA
LG
SAMSUNG
MOBILE PHONE REPAIR CENTRE

HTC
BlackBerry
NOKIA
LG
SAMSUNG
MOBILE PHONE REPAIR CENTRE

Activity

Practical Question 2



Shelve

Shelve is Python object persistence.

The shelve module in Python's standard library is for persistent data storage.

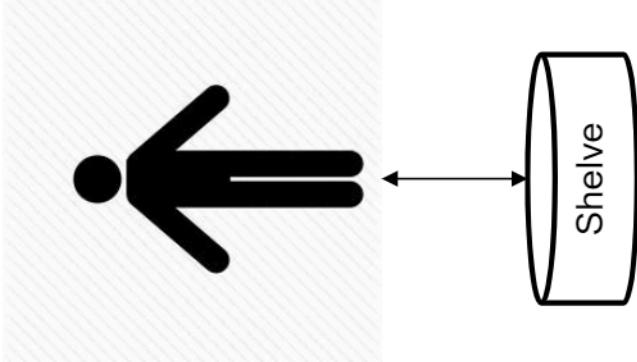
A shelf object is a dictionary-like object.

Student.py

Student

--adminNo
--gpa

__init__(adminNo, gpa)
#accessor, mutator and methods



Shelve

```
db = shelve.open('filename', flag='c', protocol=None, writeback=False)
```

The filename specified the base filename for the underlying database.

More than 1 file may be created with an extension added to the filename

The underlying database is opened for read and write by default.

```
data = db['Students']
```

Retrieve a copy of data at key (e.g 'Students').

It will raise KeyError if no such key in the underlying database

```
db['Students'] = data
```

Store data at key in the underlying database. The old data will be overwritten if using an existing key

```
db.close()
```

Close the underlying database



Recap: Dictionary

Unordered collection written with curly brackets and data is stored as a key/value pair

```
dict = { key:value }
```

```
dict = {'S1234567A': 'Amy Lim', 'T9876543B': 'Tan  
Min Min', 'S5634278Y': 'Tay Ping Sin'}
```

Add item to the dictionary

```
dict['T2345980U'] = 'Karen Pang'
```

Modify value of an item

```
dict['T2345980U'] = 'Karen Zhang'
```

Remove item from the dictionary

```
dict.pop('S1234567A')
```

Delete the dictionary

```
del dict
```



Recap: Dictionary

```
dict = {'S1234567A': 'Amy Lim', 'T9876543B': 'Tan  
Min Min', 'S5634278Y': 'Tay Ping Sim'}
```

use a for loop to loop through a dictionary

Print the key

```
for x in dict:  
    print(x)
```

Print the value

```
for x in dict:  
    print(dict[x])
```

Use Key to get the value

```
name = dict.get('T2345980U')
```

Check if key is present in the dictionary

```
if 'T2345980U' in dict:  
    print(dict['T2345980U'])
```

```
# Class definition stored in Student.py
import shelve, Student
```

Create Student object

```
s1 = Student.Student('201234L', 4.0)
s2 = Student.Student('209876T', 3.2)
```

```
# Open persistent storage for reading and
writing. Creating it if it does doesn't exist
```

```
studDict = {}
db = shelve.open('storage.db', 'c')
```

```
try:
    if 'Students' in db: #is key exist?
        studDict = db['Students'] #retrieve data
    else:
        db['Students'] = studDict #start with empty
except:
    print("Error in opening storage.db.")
```

Add object to the studDict dictionary

```
studDict[s1.get_adminNo()] = s1
studDict[s2.get_adminNo()] = s2
```

Retrieve and print object from the studDict dictionary

```
stud = studDict['209876T']
print(stud.get_adminNo(), stud.get_gpa())
```

Modify object value

```
stud = studDict.get('209876T')
stud.set_gpa(3.5)
```

Delete object from the studDict dictionary

```
studDict.pop('209876T')
```

Update studDict dictionary to the persistent storage and
close the shelve database object when it is not in use

```
db['Students'] = studDict
db.close()
```

The Phone Stall Ltd

We Buy & Sell Phones

Phones Unlocked



CASH PAID FOR PHONES

Opening Hours
Mon - Sat: 9.30 - 5pm
Tel: 078868 845226

HTC
NOKIA
LG
APPLE
SAMSUNG
REPAIR CENTRE

HTC
NOKIA
LG
APPLE
SAMSUNG
REPAIR CENTRE

NOKIA
MOBILE PHONE

Activity

Practical Question 4



Summary

- Explain the concepts of persistence
- Explain the concepts of exception handling
- Implement programs that persist data and handle exceptions