Persistence with Files in Python

Roberto A. Bittencourt

Files in Text Mode

Example: Writing into a file

```
# open file for writing
file1 = open('test.txt', 'w')

# write text into a file
file1.write('Testing writing.')

# close file
file1.close()
```

Example: copying a file into another file

```
reading = open('arq1.txt', 'r')
writing = open('arq2.txt', 'w')

text = reading.read()
writing.write(text)

reading.close()
writing.close()
```

Example: copying a list of numbers into a file

```
lst = []
n = int(input('Enter the list size: '))
print ('Type the list elements: ')
for i in range(n):
    element = int(input())
    lst.append(element)
file1 = open('list1.txt', 'w')
file1.write('%d\n' % n)
for i in range(n):
    file1.write('%d\n' % lst[i])
file1.close()
```

Text file commands in Python

- Text Files in Python
  - Object from `file` type represents a file
- Which file will be opened?
  - `open` function opens a file object, given path + filename and opening mode

```
file1 = open('mytext.txt','w')

file2 = open('c:/doc/file.c','r')
```

File modes

Opening files:

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

Opening modes:

- **`r`** : read-only
- **`w`** : write-only
- **`a`** : read/write, stream at end (append)
- **`r+`** : read/write, stream at start
- **`w+`** : read/write, always creates a new file

File commands in Python

- `fileobj = `**`open`**`(filename, mode)`
  - Opens file
- `fileobj.`**`write(`***`str`***`)`**
  - Similar to *print()*, one parameter only (must be string), does not add newline at the end
- `fileobj.`**`read`**`()`
  - similar to *input()*, reads and returns a string with all the text in the file
- `fileobj.`**`readline`**`()`
  - similar to *read()*, but reads just one line from the file and returns a string with that line
- `fileobj.`**`close()`**
  - Closes file

The **`with`** command

- The **`with`** command simplifies file handling
  - The **`with`** blocks starts with file opening
  - File is closed when the **`with`** block ends

Example:

```
with open('test.txt', 'w') as file:
    file.write('Testing writing.')
```

Example: copying a list of numbers into a file using the **with** command

```
lst = []
n = input('Type list size: ')
print ('Type list elements: ')
for i in range(n):
    element = int(input())
    lst.append(element)
with open('list1.txt', 'w') as file:
    file.write('%d\n' % n)
    for i in range(n):
      file.write('%d\n' % lst[i])
```

Example: reading a list of numbers from a file using the `with` command

```
lst = []
with open('list1.txt', 'r') as file:
    n = int(file.readline())
    for i in range(n):
        element = int(file.readline())
        lst.append(element)
print('Read %d elements:' % n)
for i in range(n):
    print(lst[i])
```

Example: reading a file whose number of lines is unknown

```
# not the best solution
str = input('Type file name: ')
with open(str, 'r') as file:
    line  = file.readline()
    while(line != '')
        print(line)
        line = file.readline()
```

```
# this is a better solution
str = input('Type file name: ')
with open(str, 'r') as file:
    for line in file:
        print(line)
```

Files in Binary Mode

Files in binary mode

- Oriented to bytes instead of characters
- File opening

  ```
  fileobj = open(filename, mode)
  ```
- Opening modes (**b** : indicates binary mode):
  - **`rb`** : read-only in binary mode
  - **`wb`** : write-only in binary mode
  - **`ab`** : read/write in binary mode, stream at end (append)
  - **`rb+`** : read/write in binary mode, stream at start
  - **`wb+`** : read/write in binary mode, always creates a new file
- Example:

  ```
  file = open('data.dat', 'wb')
  ```

pickle Library

- To deal with complex data in Python, there are various alternatives (e.g., pickle, JSON, marshal)
  - One of the simplest solutions is the **pickle** library
- File write command in pickle:

```
dump(object, fileobj)
```

  - Writes the data stored in `object` into the `fileobj` file.
  - `object` may be either a primitive type or a complex type (e.g., list, dictionary, tuple, object)
- File read command in pickle:

```
load(fileobj)
```

  - Reads from `fileobj` the object that is stored in the associated file and returns it

Example: a list

```
from pickle import dump, load

list1 = [7, 40, 12, 0, -5]
print('List:', list1)

with open('file01.dat', 'wb') as file:
  dump(list1, file)

with open('file01.dat', 'rb') as file2:
  list2 = load(file2)

print('Retrieved list:', list2)
```

Example: two lists in sequence

```
from pickle import dump, load
list1 = [7, 40, 12, 0, -5]
list2 = ['house', 'car', 'bike']
print('List 1:', list1)
print('List 2:', list2)

with open('file02.dat', 'wb') as file:
  dump(list1, file)
  dump(list2, file)

with open('file02.dat', 'rb') as file2:
  list3 = load(file2)
  list4 = load(file2)

print('Retrieved list 1:', list3)
print('Retrieved list 2:', list4)
```

Example: two objects in sequence (1)

```
from pickle import dump, load

class Student:
    def __init__(self, name, number, birth_year):
        self.name = name
        self. number = number
        self. birth_year = birth_year

student1 = Student('Marcos Santos',19211175, 1997)
student2 = Student('Mary Shaw',18111130, 1996)
print('Student 1 - Name: %s, Number: %d, Birth Year:
  %d' % (student1.name, student1.number,
  student1.birth_year))
print('Student 2 - Name: %s, Number: %d, Birth Year:
  %d' % (student2.name, student2.number,
  student2.birth_year))

# continues in the next slide
```

Example: two objects in sequence (2)

```
# continuing the example

with open('file03.dat', 'wb') as file:
  dump(student1, file)
  dump(student2, file)

with open('file03.dat', 'rb') as file2:
  student3 = load(file2)
  student4 = load(file2)

print('Retrieve student 1 - Name: %s, Number: %d,
  Birth Year: %d' % (student3.name, student3.number,
  student3.birth_year))
print('Retrieved student 2 - Name: %s, Number: %d,
  Birth Year: %d' % (student4.name, student4.number,
  student4.birth_year))
```

Example: a 5x3 matrix

```
from pickle import dump, load
grades = []
for i in range(5):
    row = []
    for j in range(3):
        print('For student %d, type the grade %d: '
            % (i+1,j+1))
        grade = float(input())
        row.append(grade)
    grades.append(row)
with open('file04.dat', 'wb') as file:
  dump(grades, file)
print('File saved!')
with open('file04.dat', 'rb') as file2:
  grades2 = load(file2)
print('File loaded!')
print('\nValues read:\n')
for i in range(5):
    for j in range(3):
        print('%5.1f' % grades2[i][j], end = '')
    print()
```

Example: a list, one element at a time

```
from pickle import dump, load
list1 = []
n = int(input("Choose the number of elements: "))
for i in range(n):
    number=int(input("Type the element %d: " % (i + 1)))
    list1.append(number)
# one can save a list one element at a time
with open('file05.dat', 'wb') as file:
  for i in range(n):
      dump(list1[i], file)
print('File saved!')
list2 = []
# later, one can read one element at a time
with open('file05.dat', 'rb') as file2:
  for i in range(n):
    element = load(file2)
    list2.append(element)
print('File loaded!')
print('Retrieved list: ')
for i in range(n):
    print(list2[i])
```

Example: retrieving a list whose size is unknown

```
from pickle import load

# when one does not know how many elements the list has,
# EOFError can be used - exception raised when
# the end of file is reached
lista = []
with open('file05.dat', 'rb') as file:
  while True:
    try:
        element = load(file)
        lista.append(element)
    except EOFError:
        break

print('File loaded!')

print('Retrieved list: ')
for element in lista:
    print(element)
```

Example: a list of objects

```
from pickle import dump, load
class Student:
    def __init__(self, name, number, birth_year):
        self.name = name
        self.number = number
        self.birth_year = birth_year
students = []
n = int(input("Type the number of students: "))
for i in range(n):
    name = input('Name: ')
    number = int(input('Number: '))
    birth_year = int(input('Birth year: '))
    student = Student(name, number, birth_year)
    students.append(student)
with open('file07.dat', 'wb') as file:
  dump(students, file)
print('File saved!')
with open('file07.dat', 'rb') as file2:
  students2 = load(file2)
print('File loaded!')
print('Retrieved list of students: ')
for student in students2:
    print('Name: %s, Number: %d, Birth Year: %d' %
     (student.name, student.number, student.birth_year))
```

Example: saving a list of objects, writing one object at a time

```
from pickle import dump
class Student:
    def __init__(self, name, number, birth_year):
        self.name = name
        self.number = number
        self.birth_year = birth_year
# If the data input process is slow,
# one can save the elements one at a time.
students = []
n = int(input("Type the number of students to save: "))
for i in range(n):
    name = input('Name: ')
    number = int(input('Number: '))
    birth_year = int(input('Birth Year: '))
    student = Student(name, number, birth_year)
    students.append(student)
    # saving one student at a time
    with open('file08.dat', 'ab') as file:
      dump(student, file)
print('Students saved in the file!')
```

Example: loading a list of objects, reading one object at a time

```
from pickle import load
class Student:
    def __init__(self, name, number, birth_year):
        self.name = name
        self.number = number
        self.birth_year = birth_year
# Loading the students that were saved from another list
students2 = []
with open('file08.dat', 'rb') as file2:
  while True:
    try:
        student = load(file2)
        students2.append(student)
    except EOFError:
        break
print('File loaded!')

print('Retrieved list of students: ')
for student in students2:
    print('Name: %s, Number: %d, Birth Year: %d' %
      (student.name, student.number, student.birth_year))
```