Files and exceptions

Python basics

Kunal Khurana

2023-10-06

Table of contents

Learning outcomes-

1. working with files (reading entire files or its contents)
2. write a file and append text into the file
3. exceptions and handling those
4. Python data structures
5. 'json' module- saves the data when program stops running

Remarks*-

1. open() function is used to read a file.
2. Windows systems use a backslash (\) instead of a forward slash (/) when displaying file paths, but you can still use forward slashes in your code.
3. The 'pass' statement acts as a placeholder- used to pass on to the next step indicating - 'silent failing'.

Reading a file

```
file_path = "E:\\machine learning projects\\python.txt"
with open (file_path) as file_object:
    for line in file_object:
        print(line)
```

```
I love programming.

I love creating new games.

I also love finding meaning in the large datasets.

I want to create an application with python.
```

```
# stripping extra lines
file_path = "E:\\machine learning projects\\python.txt"
with open (file_path) as file_object:
    for line in file_object:
        print(line.rstrip())
```

```
I love programming.
```

making a list of lines from a file

```
with open (file_path) as file_object:
    lines = file_object.readlines()    #we used readlines method

for line in lines:
    print(line.rstrip())
```

```
This file contains text that will be viewed in python's crash_course#7!

there is some white space in between!
```

working with file's contents

```
pi_string = ''
for line in lines:
    pi_string += line.rstrip()

print(pi_string)
print(len(pi_string))
```

```
This file contains text that will be viewed in python's crash_course#7!there is some white sp
108
```

```
pi_string = ''
for line in lines:
    pi_string += line.rstrip() + ' '  #adding a space in between lines

print(pi_string)
print(len(pi_string))
```

```
This file contains text that will be viewed in python's crash_course#7!  there is some white
111
```

```
any_string = ''
for line in lines:
    any_string += line.rstrip() + '\n'  #similarly, adding a line in between lines

print(any_string)
print(len(any_string))
```

```
This file contains text that will be viewed in python's crash_course#7!

there is some white space in between!

111
```

```
any_string = ''
for line in lines:
    any_string += line.rstrip() + '\n'  #similarly, adding a line in between lines
```

```
print(f"{any_string[:52]}...")         #prints first 52 characters
print(len(any_string))
```

```
This file contains text that will be viewed in pytho...
111
```

Writing into a File

```
with open ('python.txt', 'w') as file_object:
    file_object.write("I love programming.\n")
    file_object.write("I love creating new games.\n")
```

Appending to a File

```
with open ("python.txt", 'a') as file_object:  # adding content
    file_object.write("I also love finding meaning in the large datasets.\n")
    file_object.write("I want to create an application with python.\n")
```

Some examples

```
# Prompt the user for their name
name = input("Please enter your name: ")

# Open the file in write mode and write the name to it
with open("guest.txt", "w") as file:
    file.write(name)

print("Thank you! Your name has been written to guest.txt.")
```

```
Please enter your name: Rahul
Thank you! Your name has been written to guest.txt.
```

```
# reading the above file
file_path = "E:\\machine learning projects\\guest.txt"
with open (file_path) as file_object:
    for line in file_object:
        print(line)
```

```
RahulKunal

Rahul

Kriti

Megha

Soumiksha

Roshni

Sridevi

jatin

sakshi
```

```
# using while and break to store more names
while True:
    # Prompt the user for their name
    name = input("Please enter your name (or 'q' to quit): ")

    if name.lower() == 'q':
        break

    # Open the file in append mode and write the name to it
    with open("guest.txt", "a") as file:
        file.write(name + '\n')

print("Thank you! Your names have been added to guest.txt.")
```

```
Please enter your name (or 'q' to quit): jatin
Please enter your name (or 'q' to quit): sakshi
```

```
Please enter your name (or 'q' to quit): q
Thank you! Your names have been added to guest.txt.
```

Using while and break loop to ask people why they like programming

```python
# using while and break to store more names
while True:
    # Prompt the user for their name
    name = input("Please enter your name  ")
    detail = input("why do you like programming? (or 'q' to quit): ")

    if detail.lower() == 'q':
        break

    # Open the file in append mode and write the name to it
    with open("programming.txt", "a") as file:
        file.write(name + detail + '\n')

print("Thank you! Your names have been added to programming.txt.")
```

```
Please enter your name  hogaya abhi
why do you like programming? (or 'q' to quit): q
Thank you! Your names have been added to programming.txt.
```

```python
# reading the above file
file_path = "E:\\machine learning projects\\programming.txt"
with open (file_path) as file_object:
    for line in file_object:
        print(line)
```

```
I want to analyse big data and make an applicaiton

I want to solve complex mathematical problems

i am passionate about learning new things

data analysis and deep learning

help vishal with complex analysis
```

online collaborations

Exceptions

```
5/0
```

```
ZeroDivisionError: division by zero
```

```
# using try-except block to handle this kind of error
```

```
try:
    print(5/0)           #contains the code that we wish to try
except ZeroDivisionError:
    print("Error: You can't divide by zero!")
```

```
Error: You can't divide by zero!
```

```
try:
    print(5/0)           #contains the code that we wish to try
except ZeroDivisionError:
    print("Error: You can't divide by zero!")
else:
    print(answer)
```

```
Error: You can't divide by zero!
```

handling the FileNotFoundError Exception

```
filename = 'kunal.txt'

with open(filename, encoding= 'utf-8') as f:
    contents = f.read()
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'kunal.txt'
```

Tackling the FileNotFoundError

```
filename = 'kunal.txt'

try:
    with open(filename, encoding= 'utf-8') as f:
        contents = f.read()

except FileNotFoundError:
    print(f"Sorry, the file {filename} doesnot exist.")
```

```
Sorry, the file kunal.txt doesnot exist.
```

Analyzing text

```
title = 'learning python the easy way'
title.split()
```

```
['learning', 'python', 'the', 'easy', 'way']
```

```
filename = 'kunal.txt'

try:
    with open(filename, encoding='utf-8') as f:
        contents = f.read()
except FileNotFoundError:
    print(f"Sorry, the file{filename} doesnot exist.")
else:
    #count the approxiate words
    words = contents.split()
    num_words = len(words)
    print(f"The file {filename} has about {num_words} words.\n")
    print(words)
```

```
The file kunal.txt has about 138 words.

['Ce', 'chapitre', 'propose', 'd’utiliser', 'l’extension', 'Git', 'de', 'JupyterLab.', 'Un',
```

Working with multiple files

```
def count_words(filename):


    try:
        with open(filename, encoding='utf-8') as f:
            contents = f.read()
    except FileNotFoundError:
        print(f"Sorry, the file{filename} doesnot exist.")
    else:
    #count the approxiate words
        words = contents.split()
        num_words = len(words)
        print(f"The file {filename} has about {num_words} words.\n")

filename = 'kunal.txt'
count_words(filename)
```

```
The file kunal.txt has about 138 words.
```

```
#missing file car.txt has no effect on program's execution

def count_words(filename):

    try:
        with open(filename, encoding='utf-8') as f:
            contents = f.read()
    except FileNotFoundError:
        print(f"Sorry, the file {filename} doesnot exist.")
    else:
    #count the approxiate words
        words = contents.split()
        num_words = len(words)
        print(f"The file {filename} has about {num_words} words.\n")

filenames = ['kunal.txt', 'programming.txt', 'python.txt', 'car.txt']   # car.txt is missi
for filename in filenames:
    count_words(filename)
```

```
The file kunal.txt has about 138 words.
```

```
The file programming.txt has about 36 words.

The file python.txt has about 25 words.

Sorry, the file car.txt doesnot exist.
```

Failing silently

```python
def count_words(filename):

    try:
        with open(filename, encoding='utf-8') as f:
            contents = f.read()
    except FileNotFoundError:
        pass          #awesome, no need to write a default message

    else:
    #count the approxiate words
        words = contents.split()
        num_words = len(words)
        print(f"The file {filename} has about {num_words} words.\n")

filenames = ['kunal.txt', 'programming.txt', 'python.txt', 'car.txt']   # car.txt is missi
for filename in filenames:
    count_words(filename)
```

```
The file kunal.txt has about 138 words.

The file programming.txt has about 36 words.

The file python.txt has about 25 words.
```

Deciding which errors to report

1. If users know which texts are supposed to be analyzed, they might appreciate a message informing them why some texts were not analyzed.
2. Contrary to it, If users expect to see some results but don't know which texts are supposed to be analyzed, they might not need to know that some texts were unavailable.

Storing data

1. The first program will use json.dump() to store the set of numbers, and the second program will use json.load().

```
#1
import json

numbers = [12,3,31,123,44,23]

filename = 'numbers.json'
with open(filename, 'w') as f:   #'w' for writing
    json.dump(numbers,f)
```

```
#2
import json

filename = 'numbers.json'
with open(filename) as f:
    numbers = json.load(f)    #' ' for reading

print(numbers)
```

```
[12, 3, 31, 123, 44, 23]
```

Saving and using User-Generated Data

1. Saving data with json is useful when you're working with user-generated data, because if you don't store your user's information somehow, you'll lose it when the program stops running.

```
import json

username = input("what is your name? ")

filename = "username.json"
with open(filename, 'w') as f:
    json.dump(username, f)
    print(f"We'll remember your name when you come back, {username_2}!")
```

```
what is your name? kk
We'll remember your name when you come back, kk!
```

```
import json

filename = "username.json"
with open(filename) as f:
    username = json.load(f)
    print(f"Welcome back, {username}!")
```

```
Welcome back, kk!
```

Refactoring

1. means to improve the code by breaking it up into a series of functions for specific jobs.

```
#1
import json

def greet_user():
    filename = "username.json"
    try:
        with open(filename) as f:
            username = json.load(f)
    except FileNotFoundError:
        username = input("What is your name? ")
        with open(filename, 'w') as f:
            json.dump(username, f)
            print(f"We'll remember you when you come back, {username}!")
    else:
        print(f"Welcome back, {username}!")

greet_user()
```

```
Welcome back, kk!
```

```
#2
# store information and greet the user
```

```
def get_stored_username():
    filename = 'username.json'
    try:
        with open(filename) as f:
            username = json.load(f)
    except FileNotFoundError:
        return None
    else:
        return username
def greet_user():
    username = input("What is your name? ")
    if username:
        print(f"Welcome back, {username}!")
    else:
        username = input("What is your name? ")
        filename = 'username.json'
        with open(filename, 'w') as f:
            json.dump(username, f)
            print(f"We'll remember you when you come back, {username}!")


greet_user()
```

```
What is your name? Rahul
Welcome back, Rahul!
```

```
#3
# writing a code that stores the prompt for a new username

import json

def get_stored_username():
    filename = 'username.json'
    try:
        with open(filename) as f:
            username = json.load(f)
    except FileNotFoundError:
        return None
    else:
        return username
```

```
def get_new_username():
    username = input("What is your name? ")
    filename = 'username.json'
    with open(filename, 'w') as f:
        json.dump(username, f)
    return username

def greet_user():
    username = get_stored_username()
    if username:
        print(f"Welcome back, {username}!")
    else:
        username = get_new_username()
        print(f"We'll remember you when your come back, {username}!")

greet_user()
```

```
Welcome back, kk!
```