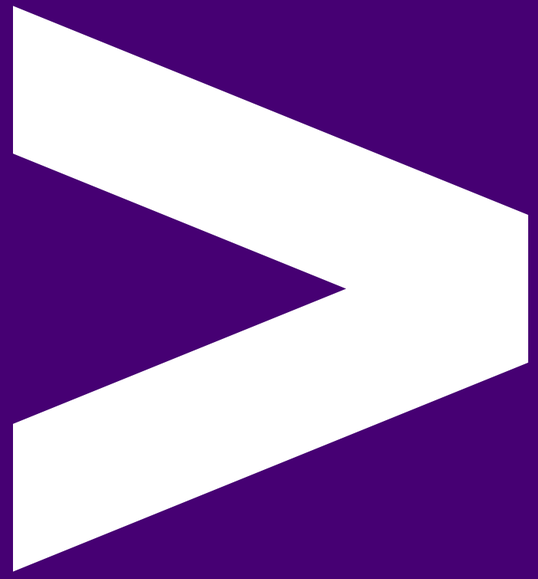
School of Tech
part of accenture

Data Persistence



School of Tech
part of accenture

Overview

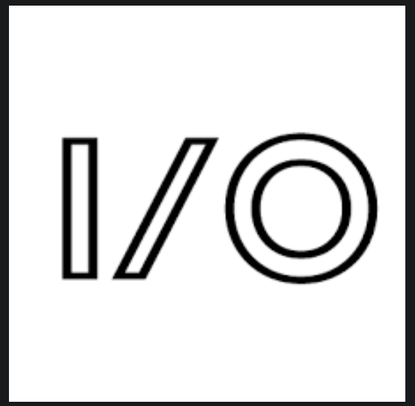
- Opening and closing files
- Reading data from files
- Writing data to files

Learning Objectives

- Identify the importance of various file handling procedures
- Implement file handling using Python

School of Tech
part of accenture

Input and Output



School of Tech
part of accenture

I / O

Input is the data you feed into your application.

Output is the new data that your application produces.

Types of I / O

- Hardware such as a keyboard and monitor
- User commands and program text on the console
- Files
- Network

Saving data permanently

- Any data that is not hardcoded into the application will be lost when the application stops
- Data needs to be saved into a file for it to outlive the application

File Handling

Some of the main things you want to do with a file are:

- Create
- Open
- Read
- Write
- Close

Opening a File

```
file = open("my_file.txt", "r")
```

`open`: built-in function to create or open a file

`my_file.txt`: path to the file, relative to the program

`"r"`: file access mode (read)

File Access Mode

A file access mode is a character that specifies the mode in which the file is opened. The default is `r`.

1. `r`: read-only
2. `r+`: both read and write
3. `w`: write-only - overwrites the file
4. `w+`: both read and write - overwrites the file
5. `a`: write-only - appends to the file
6. `a+`: both read and write - appends to the file

More exist but these are the ones we need to know for now.

Comparing File Access Modes

Mode	Read	Write	Create if missing	Overwrites file?	Appends?	Cursor Position
`r`	✅	❌	❌	❌	❌	Start of file
`w`	❌	✅	✅	✅	❌	Start (empties file)
`a`	❌	✅	✅	❌	✅	End of file
`r+`	✅	✅	❌	❌	❌	Start of file
`w+`	✅	✅	✅	✅	❌	Start (empties file)
`a+`	✅	✅	✅	❌	✅	End of file

School of Tech
part of accenture

Reading from a File

```
# hello.txt:
Hello!
```

```
# hello.py:
file = open("hello.txt", "r")
contents = file.read()
print(contents)
```

```
# Mac/Unix
$ python3 hello.py
# or on Windows
$ py hello.py
Hello!
```

School of Tech
part of accenture

Quiz Time! 🤓

I have a file called `file.txt` which has a single line of text - "hello world!". I open the file using `file.open("file.txt", "r+")`. I then immediately close the file. What is now the content of my file?

1. "hello world!"
2. "hello world!" \n
3. The file is blank
4. The file has been deleted

Answer: `1`

I have a file called `file.txt` which has a single line of text - "hello world!". I open the file using `file.open("file.txt", "w+")`. I then immediately close the file. What is now the content of my file?

1. "hello world!"
2. "hello world!" \n
3. The file is blank
4. The file has been deleted

Answer: 3

I have a file called `file.txt` which has a single line of text - "hello world!". I open the file using `file.open("file.txt", "r+")`. I then run `contents = file.read()` followed by `print(contents)`. What is my output?

1. "hello world!"
2. A blank line
3. I will get an error

Answer: `1`

I have a file called `file.txt` which has a single line of text - "hello world!". I open the file using `file.open("file.txt", "a+")`. I then run `contents = file.read()` followed by `print(contents)`. What is my output?

1. "hello world!"
2. A blank line
3. I will get an error

Answer: `2`

School of Tech
part of accenture

Emoji Check:

Do you feel you understand opening files? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😄 Yes, enough to start working on it collaboratively

School of Tech
part of accenture

Reading Line-by-Line

You can read the contents of the file line by line:

```
# people.txt
Lisa
John
```

```
# people.py
file = open("people.txt", "r")
lines = file.readlines()
for line in lines:
    print(line)
```

```
$ python3 people.py
Lisa
John
```

Exercise - 10 mins

1. Create a text file in your text editor and add some names of people on each line
2. Read the names from the file in your application and populate an empty list with the names
3. Print out the list!
4. Did you notice something weird?

Solution

```
items = []
file = open("people.txt", 'r')
for line in file.readlines():
    items.append(line)

print(items)
```

```
$ python app.py
['Lisa\n', 'John\n', 'Bob\n']
```

Every line in the file is terminated by a newline character '\n'. We need to trim that off.

School of Tech
part of accenture

Exercise - 5 mins

Find a way to trim off the newline character from every line you read in your app.

Google is your friend here!

Solution

Python has some built-in functions to help us with this:.

`rstrip()` will strip all trailing whitespace characters:

```
>>> 'test string \n \r\n\n\r \n\n'.rstrip()
'test string'
```

`strip()` will strip off all whitespace characters around any characters:

```
>>> s = "   \n\r\n  \n  abc   def \n\r\n  \n  "
>>> s.strip()
'abc   def'
```

Opening Files Safely

Remember our good friend, the `try-except` block?

```
try:
  file = open('file.txt', 'r')

except FileNotFoundError as fnfe:
  print(f'Unable to open file: {fnfe}')

except Exception as e:
  print(f'An error occurred: {e}')
```

Note: you can chain `except` blocks!

Writing to files

If for example we wrote only these two lines of code:

```
file = open('people.txt', 'w')
file.write('Susan')
```

...then what could be an issue here?

Always Close Your Files

```
file = open('people.txt', 'w')
file.write('Susan\n')
file.close()
```

When we close a file, we release our handle on it, and push our changes to disk. This allows other programs to read from or write to the latest version of the document.

School of Tech
part of accenture

try-except-finally

The `finally` keyword can be used with `try` or `try-except` to execute a block of code, no matter what the outcome is.

```
file = None

try:
    file = open(filepath, 'w')
    for item in items:
        file.write(item + '\n')

except FileNotFoundError as fnfe:
    print(f'Unable to open file: {fnfe}')

finally:
    if file:
        file.close()
```

Emoji Check:

Do you feel you understand writing to and closing files? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😄 Yes, enough to start working on it collaboratively

File context managers 1/2

These allow for the previous slide to be made easier by automatically setting up and tearing down (closing) resources.

For this we use the `with` keyword in combination with the `open()` function:

```
with open('filename.txt', 'r') as some_file:
  contents = some_file.read()
  print('file contents=', contents)
```

This creates a variable called `some_file`.

When the `with` block ends, the file object (`some_file` here) is automatically closed by python - Context managers are great to prevent accidentally holding onto resources longer than necessary.

File context managers 2/2

We can use these in `try-except` blocks like any other code:

```
items = ['apple', 'berry', 'banana']

try:
  with open(filepath, 'w') as items_file:
    for item in items:
      items_file.write(item + '\n')
except:
  print('Failed to open file')
```

School of Tech
part of accenture

Open - recap

- `open()` is a built-in function that allows us to open a file and read it's contents, returning us a file object.
- `open()` takes two main arguments:
  - The name of the file
  - The 'file mode' which we discussed earlier (e.g. `'r'`, `'w'`, `'a'`, and more)
- `as` takes whatever object is returned from the `open` function and aliases it to a temporary variable called `file`.

E.g.

```
with open(filepath, 'w') as file:
    ...
```

Exercise prep

> Instructor to give out the zip file of exercises for `data-persistence`.
>
> Everyone please unzip the file.

Exercise time

> From the zip, you should have a file `exercises/data-persistence-exercises.md`.
>
> Let's all do the exercises included in this file.

Emoji Check:

How did the exercises go? Is file handling making more sense now?

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😄 Yes, enough to start working on it collaboratively

Terms and Definitions - recap

Persistence: The characteristic of state of a system that outlives the process that created it.

Exception: Code that breaks the normal flow of execution and executes a pre-registered exception handler.

I / O: The communication between an information processing system, such as a computer, and the outside world, possibly a human or another information processing system.

Overview - recap

- Opening and closing files
- Reading data from files
- Writing data to files

Learning Objectives - recap

- Identify the importance of various file handling procedures
- Implement file handling using Python

Further Reading

- Exceptions
- File Handling

School of Tech
part of accenture

Emoji Check:

On a high level, do you think you understand the main concepts of this session? Say so if not!

1. 😢 Haven't a clue, please help!
2. 😕 I'm starting to get it but need to go over some of it please
3. 😐 Ok. With a bit of help and practice, yes
4. 🙂 Yes, with team collaboration could try it
5. 😀 Yes, enough to start working on it collaboratively