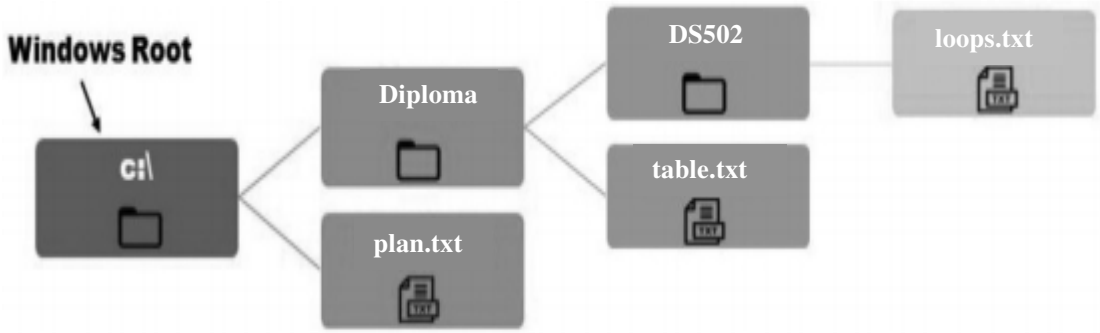
Files

Files

- Variables are a fine way to store data while your program is running,
- But if you want your data to persist even after your program has finished, you need to save it to a file.
- You can think of a file's contents as a single string value, potentially gigabytes in size.
- In this Lecture, you will learn how to use Python to create, read, and save files on the hard drive.
- Python provides built-in functions for opening a file, reading from a file, writing to a file, and closing a file.

File Paths



- File path is a path that specifies the file location on the computer.
- A file path can be a fully qualified path name or relative path.
- **Fully qualified path** (Absolute path): complete path of the file,
  - it always contains the root and the complete directory list.
  - it is **used** when the **file** is in **different location** from **python source file**
- Example on Absolute path:
  - **"C:\plan.txt"** refers to a file named "plan.txt" under root directory C:\.
  - **"C:\Diploma\table.txt"** refers to a file named "table.txt" in a subdirectory Diploma under root directory C:\.
- **Relative path**: path relative to the current working directory.
  - it always contains **"double-dots"** or **"single-dot"** as one of the directory components in a path or
  - **double dots**: are used to denote the **directory above the current directory**
  - **single dot**: is used to represent the **current directory**.
  - it is **used** when the **file** is in the **same directory as python source file**.
- Examples on relative path:
  - **"..\plan.txt"** specifies a file named "plan.txt" located in the parent of the current directory Diploma.
  - **".\table.txt"** specifies a file named "table.txt" located in a current directory named Diploma.
  - **"..\..\plan.txt"** specifies a file that is two directories above the current directory DS502.

Use of *with* Statements to Open and Close Files

- The syntax of the with statement for the file I/O is,

```
with open (file, mode, encoding) as file_handler:
        Statement_1
        Statement_2
        ...
        Statement_N
```

- **first argument (file)** is a string containing the file name

- **second argument(mode)** is another string containing a few characters describing the way in which the file will be used. **r** is the default mode

- The **as** keyword acts like an **alias** and is used to **assign** the **returning object from the open()** function **to** a **new variable file_handler**.

- The **with** statement **creates** a **context manager** and it will **automatically close the file handler object** for you **when you** are **done with it,** even if an **exception is raised** on the way, and **thus** properly **managing the resources**.

Encoding

- Unlike other languages, the character a does not imply the number 97 until it is encoded using ASCII (or other equivalent encodings).
- Moreover, the default encoding is platform dependent. In windows, it is cp1252 but utf-8 in Linux.
- So, we must not also rely on the default encoding or else our code will behave differently in different platforms.
- Hence, when working with files in text mode, it is highly recommended to specify the encoding type.

```
with open("test.txt", mode='r', encoding='utf-8') as f
```

Access Modes of the Files

Mode	Description
"r"	Opens the file in read only mode and this is the default mode.
"w"	Opens the file for writing. If a file already exists, then it'll get overwritten. If the file does not exist, then it creates a new file.
"a"	Opens the file for appending data at the end of the file automatically. If the file does not exist it creates a new file.
"r+"	Opens the file for both reading and writing.
"w+"	Opens the file for reading and writing. If the file does not exist it creates a new file. If a file already exists then it will get overwritten.
"a+"	Opens the file for reading and appending. If a file already exists, the data is appended. If the file does not exist it creates a new file.
"x"	Creates a new file. If the file already exists, the operation fails.
"rb"	Opens the binary file in read-only mode.
"wb"	Opens the file for writing the data in binary format.
"rb+"	Opens the file for both reading and writing in binary format.

Example: Program to Read and Print Each Line in "japan.txt" File Using *with Statement*.

Sample Content of "japan.txt" File is Given Below.

japan.txt

```
1  National Treasures of Japan are the most precious of Japan's Tangible Cultural Properties.
2  A Tangible Cultural Property is considered to be of historic or artistic value, classified either as
3  "buildings and structures", or as "fine arts and crafts".
```

Solution

```python
def read_file():
    print("Printing each line in text file")
    with open("japan.txt") as file_handler:
        for each_line in file_handler:
            print(each_line, end="")
def main():
    read_file()

main()
```

```
Printing each line in text file
National Treasures of Japan are the most precious of Japan's Tangible Cultural Properties.
A Tangible Cultural Property is considered to be of historic or artistic value, classified either as "buildings and
structures", or as "fine arts and crafts".
```

with statement to open more than one file

```
1. with open(in_filename) as in_file, open(out_filename, 'w') as out_file:
2.     for line in in_file:
                    .
                    .
                    .
3.         out_file.write(parsed_line)
```

File Methods to Read and Write Data

- When you use the open() function a file object is created.
- Here is the list of methods that can be called on this object

List of Methods Associated with the File Object

Method	Syntax	Description
read()	file_handler. read([size])	This method is used to read the contents of a file up to a size and return it as a string. The argument *size* is optional, and, if it is not specified, then the entire contents of the file will be read and returned.
readline()	file_handler.readline()	This method is used to read a single line in file.
readlines()	file_handler.readlines()	This method is used to read all the lines of a file as list items.
write()	file_handler. write(string)	This method will write the contents of the string to the file, returning the number of characters written. If you want to start a new line, you must include the new line character.
writelines()	file_handler. writelines(sequence)	This method will write a sequence of strings to the file.
tell()	file_handler.tell()	This method returns an integer giving the file handler's current position within the file, measured in bytes from the beginning of the file.
seek()	file_handler. seek(offset, from_what)	This method is used to change the file handler's position. The position is computed from adding *offset* to a reference point. The reference point is selected by the *from_what* argument. A *from_what* value of 0 measures from the beginning of the file, 1 uses the current file position, and 2 uses the end of the file as the reference point. If the *from_what* argument is omitted, then a default value of 0 is used, indicating that the beginning of the file itself is the reference point.

Example: Write Python Program to Read "rome.txt" File Using *read()* Method. Sample Content of "rome.txt" File is Given Below

rome.txt

```
1  Ancient Rome was a civilization which began on the Italian Peninsula in the 8th century BC.
2  The Roman Emperors were monarchial rulers of the Roman State.
3  The Emperor was supreme ruler of Rome.
4  Rome remained a republic.
```

Solution

```
1 def main():
2     with open("rome.txt") as file_handler:
3         print("Print entire file contents")
4         print(file_handler.read(), end=" ")
5 main()
```

```
Print entire file contents
Ancient Rome was a civilization which began on the Italian Peninsula in the 8th century BC.
The Roman Emperors were monarchial rulers of the Roman State.
The Emperor was supreme ruler of Rome.
Rome remained a republic.
```

- The file named rome.txt is opened using with statement.
- Even if there are any errors the file handler is closed and the resources are deallocated ②.
- Above program prints the contents of the entire file as a string ④.
- If you replace the line in ④ as ***print(file_handler.read(13), end=““)*** then it prints the first 13 characters.
  Output will be **“Ancient Rome”**

Example: Write Python Program to Read "rome.txt" file Using readline() Method

In [27]:

```
1 def main():
2     with open("rome.txt") as file_handler:
3         print("Print a single line from the file")
4         print(file_handler.readline(), end="")
5         print("Print another single line from the file")
6         print(file_handler.readline(), end="")
7 main()
```

```
Print a single line from the file
Ancient Rome was a civilization which began on the Italian Peninsula in the 8th century BC.
Print another single line from the file
The Roman Emperors were monarchial rulers of the Roman State.
```

The file_handler.readline() method reads a single line from the file ③–⑥.
If the ***file_handler .readline()*** returns an **empty string**, the **end of the file** has been **reached**.

In [36]:

```
def main():
    with open("rome.txt") as file_handler:
        print("Print a single line from the file")
        line=file_handler.readline()
        while (line!=''):
            print(line, end="")
            line=file_handler.readline()
main()
```

```
Print a single line from the file
Ancient Rome was a civilization which began on the Italian Peninsula in the 8th century BC.
The Roman Emperors were monarchial rulers of the Roman State.
The Emperor was supreme ruler of Rome.
Rome remained a republic.
```

Example: Write Python Program to Read "rome.txt" File Using *readlines()* Method

In [40]:

```
1 def main():
2     with open("rome.txt") as file_handler:
3         print("Print file contents as a list")
4         print(file_handler.readlines())
5 
6 main()
```

```
Print file contents as a list
['Ancient Rome was a civilization which began on the Italian Peninsula in the 8th century BC.\n', 'The Roman Emperor
s were monarchial rulers of the Roman State.\n', 'The Emperor was supreme ruler of Rome.\n', 'Rome remained a republ
ic.']
```

- The ***readlines()*** method **returns** a **list of strings** with each line being a list item ④.
- A newline character (\n) is left at the end of each string item of the list indicating that the end of the line has been reached.
- It is only omitted on the last line of the file if the file does not end in a newline character.

Example:

- Consider "Sample_Program.py" Python file. Write Python program to remove the comment character from all the lines in a given Python source file.
- Sample content of "Sample_Program.py" Python file is given below

Sample_Program.py

```
1  print("This is a sample program")
2  #print("Python is a very versatile language")
```

Solution

```
def main():
    with open("Sample_Program.py") as file_handler:
        for each_row in file_handler:
            each_row = each_row.replace("#", "")
            print(each_row, end="")
main()
```

```
print("This is a sample program")
print("Python is a very versatile language")
```

Example:

- Write Python Program to Reverse Each Word in "secret_societies.txt" file.
  Sample Content of "secret_societies.txt" is Given Below

secret_societies.txt

```
1  Secret Societies
2  Freemasons Illuminati
3  Rosicrucians Bilderberg Knights Templar
```

Solution

```
1  def main():
2      reversed_word=''
3      with open("secret_societies.txt") as file_handler:
4          for each_row in file_handler:
5              word_list = each_row.rstrip().split(" ")
6              for each_word in word_list:
7                  reversed_word+= each_word[::-1]
8                  reversed_word+=' '
9              print(reversed_word)
10             reversed_word=''
11 main()
```

```
etrccS seiteicoS
snosameerF itanimullI
snaicurcisoR grebredliB sthgniK ralpmeT
```

Example:

- Write Python Program to Count the Occurrences of Each Word and Also Count the Number of Words in a "quotes.txt" File.
- Sample Content of "quotes.txt" File is Given Below

quotes.txt

```
1  Happiness is the longing for repetition.
2  Artificial intelligence is no match for natural stupidity.
```

Solution

```python
def main():
    occurrence_of_words = dict()
    total_words = 0
    with open("quotes.txt") as file_handler:
        for each_row in file_handler:
            words = each_row.rstrip().split()
            total_words += len(words)
            for each_word in words:
                occurrence_of_words[each_word] = occurrence_of_words[each_word]+1 if each_word in occurrence_of_words else 1
        print("The number of times each word appears in a sentence is")
        print(occurrence_of_words)
        print(f"Total number of words in the file are {total_words}")
main()
```

```
The number of times each word appears in a sentence is
{'Happiness': 1, 'is': 2, 'the': 1, 'longing': 1, 'for': 2, 'repetition.': 1, 'Artificial': 1, 'intelligence': 1, 'n
o': 1, 'match': 1, 'natural': 1, 'stupidity.': 1}
Total number of words in the file are 14
```

Example:

- Write Python Program to Find the Longest Word in a File.
- Get the File Name from User.
- (Assume User Enters the File Name as "animals.txt" and its Sample Contents are as Below)

animals.txt

```
1  Rhinocerose Porcupine Squirrel
2  Flamingo Pangolin Woodpecker
3  Salamander Antelope Kangaroo
```

Solution

```
1  def read_file(file_name):
2      with open(file_name) as file_handler:
3          longest_word = ""
4          for each_row in file_handler:
5                  word_list = each_row.rstrip().split()
6                  for each_word in word_list:
7                      if len(each_word) > len(longest_word):
8                          longest_word = each_word
9      print(f"The longest word in the file is {longest_word}")
10 def main():
11     file_name = input("Enter file name: ")
12     read_file(file_name)
13 main()
```

```
Enter file name: animals.txt
The longest word in the file is Rhinocerose
```

Project

Project: Generating Random Quiz Files

- Say you're a geography teacher with 35 students in your class and you want to give a pop quiz on US state capitals.

- You'd like to randomize the order of questions so that each quiz is unique, making it impossible for anyone to crib answers from anyone else.

- Of course, doing this by hand would be a lengthy and boring affair. Fortunately, you know some Python.

Project: Generating Random Quiz Files ...

- Here is what the program does:
  - Creates 35 different quizzes
  - Creates 50 multiple-choice questions for each quiz, in random order
  - Provides the correct answer and three random wrong answers for each question, in random order
  - Writes the quizzes to 35 text files
  - Writes the answer keys to 35 text files

Project: Generating Random Quiz Files ...

- This means the code will need to do the following:
  - Store the states and their capitals (questions, answers) in a dictionary
  - Call open(), write(), and close() for the quiz and answer key text files
  - Use random.shuffle() to randomize the order of the questions and multiple-choice options

Output: A Quiz file

After you run the program, this is how your capitalsquiz1.txt file will look, though of course your questions and answer options may be different from those shown here, depending on the outcome of your random.shuffle() calls:

```
Name:

Date:

Period:

                    State Capitals Quiz (Form 1)

1. What is the capital of West Virginia?

    A. Hartford

    B. Santa Fe

    C. Harrisburg

    D. Charleston

2. What is the capital of Colorado?

    A. Raleigh

    B. Harrisburg

    C. Denver

    D. Lincoln
```

Output: An Answer file

The corresponding capitalsquiz_answers1.txt text file will look like this:

```
1. D

2. C

3. A

4. C
```

Solution

Step 1: Store the Quiz Data in a Dictionary

```
❶import random
```

Since this program will be randomly ordering the questions and answers, you'll need to import the random module ❶ to make use of its functions.

Step 1: Store the Quiz Data in a Dictionary

❷

```
# The quiz data. Keys are states and values are their capitals.
capitals = {'Alabama': 'Montgomery', 'Alaska': 'Juneau', 'Arizona': 'Phoenix', 'Arkansas': 'Little Rock',
    'California': 'Sacramento', 'Colorado': 'Denver', 'Connecticut': 'Hartford', 'Delaware': 'Dover',
    'Florida': 'Tallahassee', 'Georgia': 'Atlanta', 'Hawaii': 'Honolulu', 'Idaho': 'Boise', 'Illinois':
    'Springfield', 'Indiana': 'Indianapolis', 'Iowa': 'Des Moines', 'Kansas': 'Topeka', 'Kentucky': 'Frankfort',
    'Louisiana': 'Baton Rouge', 'Maine': 'Augusta', 'Maryland': 'Annapolis', 'Massachusetts': 'Boston',
    'Michigan': 'Lansing', 'Minnesota': 'Saint Paul', 'Mississippi': 'Jackson', 'Missouri':'Jefferson City',
    'Montana': 'Helena', 'Nebraska': 'Lincoln', 'Nevada': 'Carson City', 'New Hampshire': 'Concord',
    'New Jersey': 'Trenton', 'New    Mexico': 'Santa Fe', 'New York': 'Albany', 'North Carolina': 'Raleigh',
    'North Dakota': 'Bismarck', 'Ohio': 'Columbus', 'Oklahoma': 'Oklahoma City', 'Oregon': 'Salem',
    'Pennsylvania': 'Harrisburg', 'Rhode Island': 'Providence', 'South Carolina': 'Columbia',
    'South Dakota': 'Pierre', 'Tennessee': 'Nashville', 'Texas': 'Austin', 'Utah': 'Salt Lake City',
    'Vermont': 'Montpelier', 'Virginia': 'Richmond', 'Washington': 'Olympia', 'West Virginia': 'Charleston',
    'Wisconsin': 'Madison', 'Wyoming': 'Cheyenne'}
```

The capitals variable ❷ contains a dictionary with US states as keys and their capitals as values.

Step 1: Store the Quiz Data in a Dictionary ...

❸

```
   # Generate 35 quiz files.
for quizNum in range(35):

        # TODO: Create the quiz and answer key files.

        # TODO: Write out the header for the quiz.

        # TODO: Shuffle the order of the states.

        # TODO: Loop through all 50 states, making a question for each.

```

And since you want to create 35 quizzes, the code that actually generates the quiz and answer key files (marked with TODO comments for now) will go inside a for loop that loops 35 times ❸. (This number can be changed to generate any number of quiz files.)

Step 2: Create the Quiz File and Shuffle the Question Order

- Now it's time to start filling in those TODOs.
- The code in the loop will be repeated 35 times—once for each quiz—so you have to worry about only one quiz at a time within the loop.
- First you'll create the actual quiz file.
- It needs to have a unique filename and should also have some kind of standard header in it, with places for the student to fill in a name, date, and class period.
- Then you'll need to get a list of states in randomized order, which can be used later to create the questions and answers for the quiz.

Step 2: Create the Quiz File and Shuffle the Question Order ...

```
17    # Generate 35 quiz files.
18  for quizNum in range(35):
19
20      # Create the quiz and answer key files.
21  ❶   quizFile = open(f'capitalsquiz{quizNum + 1}.txt', 'w')
22
23
24  ❷   answerKeyFile = open(f'capitalsquiz_answers{quizNum + 1}.txt', 'w')
25
```

- The filenames for the quizzes will be capitalsquiz<N>.txt, where <N> is a unique number for the quiz that comes from quizNum, the for loop's counter.
- The answer key for capitalsquiz<N>.txt will be stored in a text file named capitalsquiz_answers<N>.txt.

- Each time through the loop, the {quizNum + 1} placeholder in f'capitalsquiz{quizNum + 1}.txt' and f'capitalsquiz_answers{quizNum + 1}.txt' will be replaced by the unique number, so the first quiz and answer key created will be capitalsquiz1.txt and capitalsquiz_answers1.txt.
- These files will be created with calls to the open() function at ❶ and ❷, with 'w' as the second argument to open them in write mode.

Step 2: Create the Quiz File and Shuffle the Question Order ...

```
17    # Generate 35 quiz files.
18  for quizNum in range(35):
19      # Create the quiz and answer key files.
20      quizFile = open(f'capitalsquiz{quizNum + 1}.txt', 'w')
21      answerKeyFile = open(f'capitalsquiz_answers{quizNum + 1}.txt', 'w')
22
23      # Write out the header for the quiz.
24      quizFile.write('Name:\n\nDate:\n\nPeriod:\n\n')
25      quizFile.write((' ' * 20) + f'State Capitals Quiz (Form{quizNum + 1})')
26      quizFile.write('\n\n')
```

❸

The write() statements at ❸ create a quiz header for the student to fill out.

Step 2: Create the Quiz File and Shuffle the Question Order ...

```
17      # Generate 35 quiz files.
18  for quizNum in range(35):
19      # Create the quiz and answer key files.
20      quizFile = open(f'capitalsquiz{quizNum + 1}.txt', 'w')
21      answerKeyFile = open(f'capitalsquiz_answers{quizNum + 1}.txt', 'w')
22
23       # Write out the header for the quiz.
24      quizFile.write('Name:\n\nDate:\n\nPeriod:\n\n')
25      quizFile.write((' ' * 20) + f'State Capitals Quiz (Form{quizNum + 1})')
26      quizFile.write('\n\n')
27
28      # Shuffle the order of the states.
29      states = list(capitals.keys())
30      random.shuffle(states)
31
32       # TODO: Loop through all 50 states, making a question for each.
```

❹

Finally, a randomized list of US states is created with the help of the random.shuffle() function ❹, which randomly reorders the values in any list that is passed to it.

Step 3: Create the Answer Options

- Now you need to generate the answer options for each question, which will be multiple choice from A to D.
- You'll need to create another for loop—this one to generate the content for each of the 50 questions on the quiz
- Then there will be a third for loop nested inside to generate the multiple-choice options for each question.
- Make your code look like the following:

```
18  for quizNum in range(35):
19      # Create the quiz and answer key files.
20      quizFile = open(f'capitalsquiz{quizNum + 1}.txt', 'w')
21      answerKeyFile = open(f'capitalsquiz_answers{quizNum + 1}.txt', 'w')
22       # Write out the header for the quiz.
23      quizFile.write('Name:\n\nDate:\n\nPeriod:\n\n')
24      quizFile.write((' ' * 20) + f'State Capitals Quiz (Form{quizNum + 1})')
25      quizFile.write('\n\n')
26       # Shuffle the order of the states.
27      states = list(capitals.keys())
28      random.shuffle(states)
29
30      # Loop through all 50 states, making a question for each.
31      for questionNum in range(50):
32
33           # Get right and wrong answers.
34          correctAnswer = capitals[states[questionNum]]
35
36          wrongAnswers = list(capitals.values())
37
38          del wrongAnswers[wrongAnswers.index(correctAnswer)]
39
40          wrongAnswers = random.sample(wrongAnswers, 3)
41
42          answerOptions = wrongAnswers + [correctAnswer]
43
44          random.shuffle(answerOptions)
45
46           # TODO: Write the question and answer options to the quiz file.
47
48           # TODO: Write the answer key to a file.
```

Step 3: Create the Answer Options ...

```
30    # Loop through all 50 states, making a question for each.
31    for questionNum in range(50):
32
33        # Get right and wrong answers.
34        correctAnswer = capitals[states[questionNum]]
35
36        wrongAnswers = list(capitals.values())
37
38        del wrongAnswers[wrongAnswers.index(correctAnswer)]
39
40        wrongAnswers = random.sample(wrongAnswers, 3)
41
42        answerOptions = wrongAnswers + [correctAnswer]
43
44        random.shuffle(answerOptions)
45
46        # TODO: Write the question and answer options to the quiz file.
47
48        # TODO: Write the answer key to a file.
```

❶

- The correct answer is easy to get—it's stored as a value in the capitals dictionary ❶.
- This loop will loop through the states in the shuffled states list, from states[0] to states[49], find each state in capitals, and store that state's corresponding capital in correctAnswer.

Step 3: Create the Answer Options ...

```
30      # Loop through all 50 states, making a question for each.
31      for questionNum in range(50):
32
33          # Get right and wrong answers.
34          correctAnswer = capitals[states[questionNum]]
35
36      ❷   wrongAnswers = list(capitals.values())
37
38      ❸   del wrongAnswers[wrongAnswers.index(correctAnswer)]
39
40      ❹   wrongAnswers = random.sample(wrongAnswers, 3)
41
42          answerOptions = wrongAnswers + [correctAnswer]
43
44          random.shuffle(answerOptions)
45
46          # TODO: Write the question and answer options to the quiz file.
47
48          # TODO: Write the answer key to a file.
```

- The list of possible wrong answers is trickier.
- You can get it by retrieving all the values in the capitals dictionary ❷, deleting the correct answer ❸, and selecting three random values from this list ❹.
- The random.sample() function makes it easy to do this selection.
- Its first argument is the list you want to select from; the second argument is the number of values you want to select.

Step 3: Create the Answer Options ...

```
30      # Loop through all 50 states, making a question for each.
31      for questionNum in range(50):
32
33          # Get right and wrong answers.
34          correctAnswer = capitals[states[questionNum]]
35
36          wrongAnswers = list(capitals.values())
37
38          del wrongAnswers[wrongAnswers.index(correctAnswer)]
39
40          wrongAnswers = random.sample(wrongAnswers, 3)
41
42          answerOptions = wrongAnswers + [correctAnswer]
43
44          random.shuffle(answerOptions)
45
46          # TODO: Write the question and answer options to the quiz file.
47
48          # TODO: Write the answer key to a file.
```

❺

The full list of answer options is the combination of these three wrong answers with the correct answers ❺.

Step 3: Create the Answer Options ...

```
30      # Loop through all 50 states, making a question for each.
31      for questionNum in range(50):
32
33          # Get right and wrong answers.
34          correctAnswer = capitals[states[questionNum]]
35
36          wrongAnswers = list(capitals.values())
37
38          del wrongAnswers[wrongAnswers.index(correctAnswer)]
39
40          wrongAnswers = random.sample(wrongAnswers, 3)
41
42          answerOptions = wrongAnswers + [correctAnswer]
43
44      ❻   random.shuffle(answerOptions)
45
46          # TODO: Write the question and answer options to the quiz file.
47
48          # TODO: Write the answer key to a file.
```

Finally, the answers need to be randomized ❻ so that the correct response isn't always choice D.

Step 4: Write Content to the Quiz and Answer Key Files

- All that is left is to write the question to the quiz file and the answer to the answer key file.

Make your code look like the following:

```
18  for quizNum in range(35):
19      # Create the quiz and answer key files.
20      quizFile = open(f'capitalsquiz{quizNum + 1}.txt', 'w')
21      answerKeyFile = open(f'capitalsquiz_answers{quizNum + 1}.txt', 'w')
22       # Write out the header for the quiz.
23      quizFile.write('Name:\n\nDate:\n\nPeriod:\n\n')
24      quizFile.write((' ' * 20) + f'State Capitals Quiz (Form{quizNum + 1})')
25      quizFile.write('\n\n')
26       # Shuffle the order of the states.
27      states = list(capitals.keys())
28      random.shuffle(states)
29      # Loop through all 50 states, making a question for each.
30      for questionNum in range(50):
31           # Get right and wrong answers.
32          correctAnswer = capitals[states[questionNum]]
33          wrongAnswers = list(capitals.values())
34          del wrongAnswers[wrongAnswers.index(correctAnswer)]
35          wrongAnswers = random.sample(wrongAnswers, 3)
36          answerOptions = wrongAnswers + [correctAnswer]
37          random.shuffle(answerOptions)
38
39           # Write the question and the answer options to the quiz file.
40          quizFile.write(f'{questionNum + 1}. What is the capital of {states[questionNum]}?\n')
41          for i in range(4):
42              quizFile.write(f"    {'ABCD'[i]}. { answerOptions[i]}\n")
43          quizFile.write('\n')
44
45          # Write the answer key to a file.
46          answerKeyFile.write(f"{questionNum + 1}.{'ABCD'[answerOptions.index(correctAnswer)]}")
47
48      quizFile.close()
49      answerKeyFile.close()
```

Step 4: Write Content to the Quiz and Answer Key Files ...

```
39          # Write the question and the answer options to the quiz file.
40          quizFile.write(f'{questionNum + 1}. What is the capital of {states[questionNum]}?\n')
41      ❶   for i in range(4):
42      ❷       quizFile.write(f"    {'ABCD'[i]}. { answerOptions[i]}\n")
43          quizFile.write('\n')
44
45          # Write the answer key to a file.
46          answerKeyFile.write(f"{questionNum + 1}.{'ABCD'[answerOptions.index(correctAnswer)]}")
47
48      quizFile.close()
49      answerKeyFile.close()
```

- A for loop that goes through integers 0 to 3 will write the answer options in the answerOptions list ❶.
- The expression 'ABCD'[i] at ❷ treats the string 'ABCD' as an array and will evaluate to 'A','B', 'C', and then 'D' on each respective iteration through the loop.

Step 4: Write Content to the Quiz and Answer Key Files ...

```
39          # Write the question and the answer options to the quiz file.
40          quizFile.write(f'{questionNum + 1}. What is the capital of {states[questionNum]}?\n')
41          for i in range(4):
42              quizFile.write(f"    {'ABCD'[i]}. { answerOptions[i]}\n")
43          quizFile.write('\n')
44
45          # Write the answer key to a file.
46          answerKeyFile.write(f"{questionNum + 1}.{'ABCD'[answerOptions.index(correctAnswer)]}")
47
48      quizFile.close()
49      answerKeyFile.close()
```

❸

- In the final line ❸, the expression answerOptions.index(correctAnswer) will find the integer index of the correct answer in the randomly ordered answer options, and **'ABCD'[answerOptions.index(correctAnswer)]** will evaluate to the correct answer's letter to be written to the answer key file.

Output: A Quiz file

After you run the program, this is how your capitalsquiz1.txt file will look, though of course your questions and answer options may be different from those shown here, depending on the outcome of your random.shuffle() calls:

```
Name:

Date:

Period:

                    State Capitals Quiz (Form 1)

1. What is the capital of West Virginia?

    A. Hartford

    B. Santa Fe

    C. Harrisburg

    D. Charleston

2. What is the capital of Colorado?

    A. Raleigh

    B. Harrisburg

    C. Denver

    D. Lincoln
```

Output: An Answer file

The corresponding capitalsquiz_answers1.txt text file will look like this:

```
1. D

2. C

3. A

4. C
```