Practice Quiz: Managing Files & Directories

TOTAL POINTS 5

1. The create_python_script function creates a new python script in the current working directory, adds the line of comments to it declared by the 'comments' variable, and returns the size of the new file. Fill in the gaps to create a script called "program.py".

1 / 1 point

```
def create_python_script(filename):
  comments = "# Start of a new Python program"
  with open(filename,'w') as file:
    filesize = file.write(comments)
  return(filesize)

print(create_python_script("program.py"))
```

Run

Reset

```
31
```

✓ **Correct**

Great work! Your new python script is now ready for some real code!

2. The new_directory function creates a new directory inside the current working directory, then creates a new empty file inside the new directory, and returns the list of files in that directory. Fill in the gaps to create a file "script.py" in the directory "PythonPrograms".

1 / 1 point

```python
import os

def new_directory(directory, filename):
  # Before creating a new directory, check to see if it already exists
  if os.path.isdir(directory) == False:
    os.mkdir(directory)

  # Create the new file inside of the new directory
  os.chdir(directory)
  with open (filename,'w') as file:
    pass

  # Return the list of files in the new directory
  return os.listdir()

print(new_directory("PythonPrograms", "script.py"))
```

Run

Reset

```
['script.py']
```

✓ **Correct**

Well done, you! Working with files and directories can be a

```
Well done, you! Working with files and directories can be a
little tricky, and you're getting the hang of it!
```

3. Which of the following methods from the os module will create a new directory?

1 / 1 point

- ○ path.isdir()
- ○ listdir()
- ● mkdir()
- ○ chdir()

✓ **Correct**

Right on! os.mkdir() will create a new directory with the name provided as a string parameter.

4. The file_date function creates a new file in the current working directory, checks the date that the file was modified, and returns just the date portion of the timestamp in the format of yyyy-mm-dd. Fill in the gaps to

1 / 1 point

4. The file_date function creates a new file in the current working directory, checks the date that the file was modified, and returns just the date portion of the timestamp in the format of yyyy-mm-dd. Fill in the gaps to create a file called "newfile.txt" and check the date that it was modified.

1 / 1 point

```python
import os
import datetime

def file_date(filename):
  # Create the file in the current directory
  with open(filename, 'w') as file:
    pass
  timestamp = os.path.getmtime(filename)
  # Convert the timestamp into a readable format, then into a string
  current_time = datetime.datetime.fromtimestamp(timestamp)
  # Return just the date portion
  # Hint: how many characters are in “yyyy-mm-dd”?
  return ("{}".format(current_time.strftime("%Y-%m-%d")))

print(file_date("newfile.txt"))
# Shojuld be today's date in the format of yyyy-mm-ddcddd
```

Run

Reset

```
2020-06-29
```

✓ Correct

Way to go! You remembered the commands to convert timestamps

5. The parent_directory function returns the name of the directory that's located just above the current working directory. Remember that '..' is a relative path alias that means "go up to the parent directory". Fill in the gaps to complete this function.

1 / 1 point

```python
import os
def parent_directory():
  # Create a relative path to the parent
  # of the current working directory
  relative_parent = os.path.join('..',"w" )

  # Return the absolute path of the parent directory
  return os.path.abspath('..')

print(parent_directory())
```

Run

Reset

```
/
```

✓ **Correct**

Excellent! You made all the right moves to print the path of
the parent directory!