Practice Quiz: Managing Files & Directories

TOTAL POINTS 5

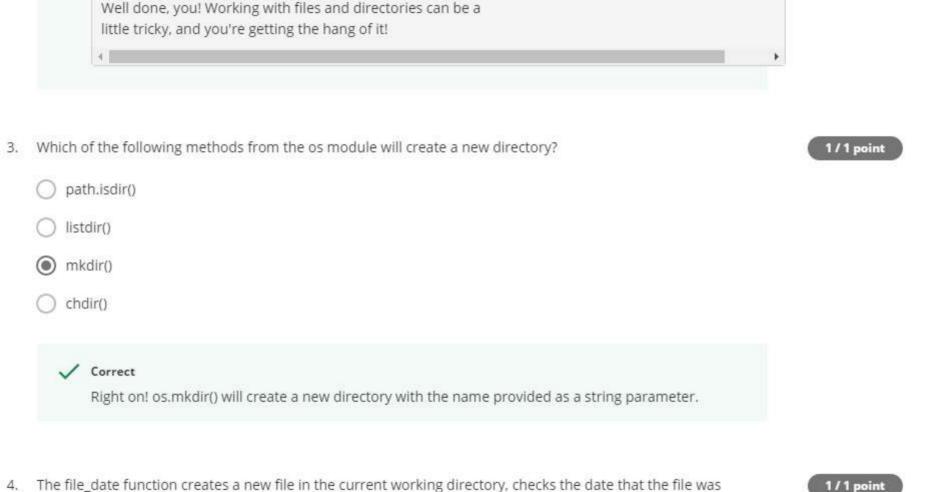
 The create_python_script function creates a new python script in the current working directory, adds the line of comments to it declared by the 'comments' variable, and returns the size of the new file. Fill in the gaps to create a script called "program.py".

1/1 point

✓ Correct

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

```
The new_directory function creates a new directory inside the current working directory, then creates a new
                                                                                                               1/1 point
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".
            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:
      10
      11
                pass
      12
              # Return the list of files in the new directory
      13
      14
              return os.listdir()
                                                                                             Run
      15
                                                                                             Reset
            print(new directory("PythonPrograms", "script.py"))
   ['script.py']
   ✓ Correct
         Well done, you! Working with files and directories can be a
```



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

```
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.
           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
      9
     10
             current time = datetime.datetime.fromtimestamp(timestamp)
             # Return just the date portion
     11
             # Hint: how many characters are in "yyyy-mm-dd"?
     12
             return ("{}".format(current_time.strftime("%Y-%m-%d")))
     13
     14
                                                                                          Run
           print(file_date("newfile.txt"))
                                                                                         Reset
           # Shojuld be today's date in the format of yyyy-mm-ddcddd
  2020-06-29
  ✓ Correct
```

Way to go! You remembered the commands to convert timestamps

4. The file_date function creates a new file in the current working directory, checks the date that the file was

1/1 point

 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.

```
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())
Reset
```

Correct

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