**Practice Quiz: Managing Files & Directories**

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".

def create\_python\_script(filename):

  comments = "# Start of a new Python program"

  with \_\_\_:

    filesize = \_\_\_

  return(filesize)

print(create\_python\_script("program.py"))

Solution:

1. 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".

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:

    \_\_\_

  # Create the new file inside of the new directory

  os.chdir(\_\_\_)

  with open (\_\_\_) as file:

    pass

  # Return the list of files in the new directory

  return \_\_\_

print(new\_directory("PythonPrograms", "script.py"))

solution:

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

1 point



path.isdir()



listdir()



mkdir()



chdir()

4.Question 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.

import os

import datetime

def file\_date(filename):

  # Create the file in the current directory

  \_\_\_

  timestamp = \_\_\_

  # Convert the timestamp into a readable format, then into a string

  \_\_\_

  # Return just the date portion

  # Hint: how many characters are in “yyyy-mm-dd”?

  return ("{\_\_\_}".format(\_\_\_))

print(file\_date("newfile.txt"))

# Should be today's date in the format of yyyy-mm-dd

Solution:

1. 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(\_\_\_, \_\_\_)

  # Return the absolute path of the parent directory

  return \_\_\_

print(parent\_directory())

solution: