**LOGIC BUILDING**

**ASSIGNMENT\_01**

Search online for the Python documentation for the len() function. It will be on a web page titled “Built-in Functions.” Skim the list of other functions Python has, look up what the round() function does, and experiment with it in the interactive shell.

**len() is a built-in function in python.You can use the len() to get the length of the given string, array, list, tuple, dictionary, etc.**

**The len built-in returns the number of elements in a collection.**

**# testing len()**

**str1 = "Welcome to Python"**

**print("The length of the string is :", len(str1))**

**Output :**

**The length of the string is : 18**

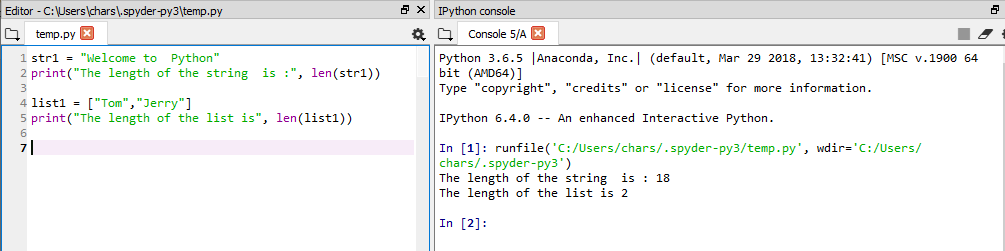
**# to find the length of the list**

**list1 = ["Tom","Jerry"]**

**print("The length of the list is", len(list1))**

**Output :**

**The length of the list is 2**



Look up the round() and abs() functions on the Internet, and find out what they do. Experiment with them in the interactive shell.

**Round()**

**Python provides an inbuilt function round() which rounds off to the given number of digits and returns the floating point number,**

**if no number of digits is provided for round off , it rounds off the number to the nearest integer.**

**>>> round(1.5789,3)**

**1.579**

**>>> round(1.2222,1)**

**1.2**

**abs()**

**The abs() function of Python's standard library returns the absolute value of the given number. Absolute value of a number is**

**the value without considering its sign. Hence absolute of 10 is 10, -10 is also 10.**

**If the number is a complex number, abs() returns its magnitude.**

**>>> abs(11.11)**

**11.11**

**>>> abs(-11.11)**

**11.11**

**ASSIGNMENT\_02**

Write a function named collatz() that has one parameter named number.

If number is even, then collatz() should print number // 2 and return this value.

If number is odd, then collatz() should print and return 3 \* number + 1.

**def collatz(number):**

**if number % 2 == 0:**

**print(number//2)**

**else:**

**result = 3\*number + 1**

**print(result)**

**try:**

**n = input("Give me a number: ")**

**while n != 1:**

**n = collatz(int(n))**

**except ValueError:**

**print('Type a number please!')**

Then write a program that lets the user type in an integer and that keeps calling collatz() on that number until the function returns the value 1. (Amazingly enough, this sequence actually works for any integer—sooner or later, using this sequence, you’ll arrive at 1! Even mathematicians aren’t sure why. Your program is exploring what’s called the Collatz sequence, sometimes called “the simplest impossible math problem.”)

**try:**

**n = input("Give me a number: ")**

**while n != 1:**

**n = collatz(int(n))**

**except ValueError:**

**print('Type a number please!')**

Remember to convert the return value from input() to an integer with the int() function; otherwise, it will be a string value.

Hint: An integer number is even if number % 2 == 0, and it’s odd if

number % 2 == 1.

The output of this program could look something like this:-

Enter number:

3

10

5

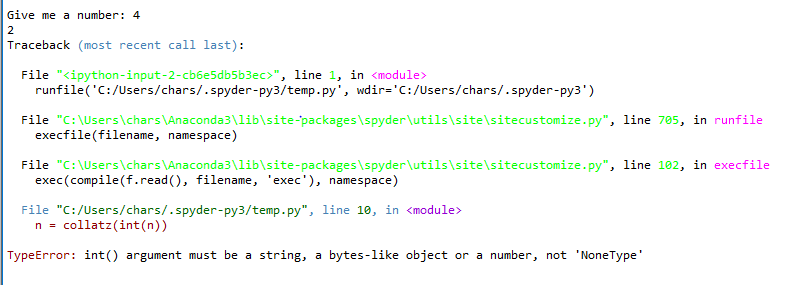
16

8

4

2

1





Input Validation:-

Add try and except statements to the previous project to detect whether the user types in a noninteger string. Normally, the int() function will raise a ValueError error if it is passed a noninteger string, as in int('puppy'). In the except clause, print a message to the user saying they must enter an integer.

**ASSIGNMENT\_03**

Say you have a list value like this:

spam = ['apples', 'bananas', 'tofu', 'cats']

Write a function that takes a list value as an argument and returns a string with all the items separated by a comma and a space, with and inserted before the last item.

For example, passing the previous spam list to the function would return 'apples, bananas, tofu, and cats'. But your function should be able to work with any list value passed to it.

**def listToString(list):**

**if list[-1]:**

**list.append('and '+str(list[-1]))**

**list.remove(list[-2])**

**for i in range(len(list)):**

**print(''+list[i]+', ')**

**spam = ['apples', 'bananas', 'tofu', 'cats']**

**listToString(spam)**

**spam = ['apples', 'bananas', 'tofu', 'cats']**

**def lst2str(spam):**

**str = ''**

**for word in spam:**

**if word != spam[-1]:**

**spacing = word + ", "**

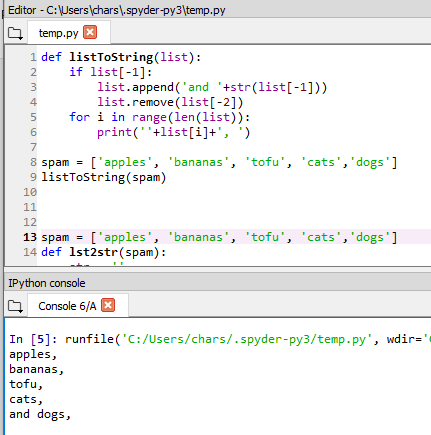
**str += spacing**

**else:**

**spacing = "and " + word**

**str += spacing**

**return str**



**ASSIGNMENT\_04**

Say you have a list of lists where each value in the inner lists is a one-character string, like this:

grid = [['.', '.', '.', '.', '.', '.'],

['.', 'O', 'O', '.', '.', '.'],

['O', 'O', 'O', 'O', '.', '.'],

['O', 'O', 'O', 'O', 'O', '.'],

['.', 'O', 'O', 'O', 'O', 'O'],

['O', 'O', 'O', 'O', 'O', '.'],

['O', 'O', 'O', 'O', '.', '.'],

['.', 'O', 'O', '.', '.', '.'],

['.', '.', '.', '.', '.', '.']]

You can think of grid[x][y] as being the character at the x- and y-coordinates of a “picture” drawn with text characters. The (0, 0) origin will be in the upper-left corner, the x-coordinates increase going right, and the y-coordinates increase going down.

Copy the previous grid value, and write code that uses it to print the image.

..OO.OO..

.OOOOOOO.

.OOOOOOO.

..OOOOO..

...OOO...

....O....

Hint: You will need to use a loop in a loop in order to print grid[0][0], then grid[1][0], then grid[2][0], and so on, up to grid[8][0]. This will finish the first row, so then print a newline. Then your program should print grid[0][1], then grid[1][1], then grid[2][1], and so on. The last thing your program will print is grid[8][5]. Also, remember to pass the end keyword argument to print() if you

don’t want a newline printed automatically after each print() call.

**grid = [['.','.','.','.','.','.'],**

**['.','0','0','.','.','.'],**

**['0','0','0','0','.','.'],**

**['0','0','0','0','0','.'],**

**['.','0','0','0','0','0'],**

**['0','0','0','0','0','.'],**

**['0','0','0','0','.','.'],**

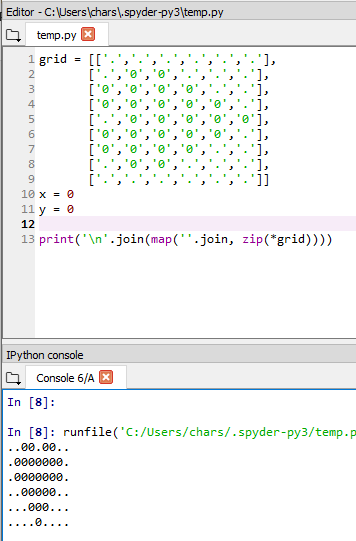
**['.','0','0','.','.','.'],**

**['.','.','.','.','.','.']]**

**x = 0**

**y = 0**

**print('\n'.join(map(''.join, zip(\*grid))))**



**ASSIGNMENT\_05**

You are creating a fantasy video game.

The data structure to model the player’s inventory will be a dictionary where the keys are string values describing the item in the inventory and the value is an integer value detailing how many of that item the player has.

For example, the dictionary value {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12} means the player has 1 rope, 6 torches, 42 gold coins, and so on.

Write a function named displayInventory() that would take any possible “inventory” and display it like the following:

Inventory:

12 arrow

42 gold coin

1 rope

6 torch

1 dagger

Total number of items: 62

Hint: You can use a for loop to loop through all the keys in a dictionary.

# inventory.py

stuff = {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12}

def displayInventory(inventory):

print("Inventory:")

item\_total = 0

for k, v in inventory.items():

# FILL THIS PART IN

print("Total number of items: " + str(item\_total))

displayInventory(stuff)

**ASSIGNMENT\_06**

Imagine that a vanquished dragon’s loot is represented as a list of strings like this:

dragonLoot = ['gold coin', 'dagger', 'gold coin', 'gold coin', 'ruby']

Write a function named addToInventory(inventory, addedItems), where the inventory parameter is a dictionary representing the player’s inventory (like in the previous project) and the addedItems parameter is a list like dragonLoot.

The addToInventory() function should return a dictionary that represents the updated inventory. Note that the addedItems list can contain multiples of the same item. Your code could look something like this:

def addToInventory(inventory, addedItems):

# your code goes here

inv = {'gold coin': 42, 'rope': 1}

dragonLoot = ['gold coin', 'dagger', 'gold coin', 'gold coin', 'ruby']

inv = addToInventory(inv, dragonLoot)

displayInventory(inv)

The previous program (with your displayInventory() function from the previous project) would output the following:

Inventory:

45 gold coin

1 rope

1 ruby

1 dagger

Total number of items: 48

**1) def addtoinventory(inventory,lootlist):**

**for i in range(len(lootlist)):**

**if lootlist[i] in inventory:**

**inventory[lootlist[i]] = inventory[lootlist[i]] + 1**

**else:**

**inventory.setdefault(lootlist[i],1)**

**return inventory**

**def displayinventory(inventory):**

**total\_items = 0**

**for k,v in inventory.items():**

**print(k + ' : ' + str(v))**

**print("Total number of items: " + str(total\_items))**

**inv = {'gold coin': 42, 'rope': 1}**

**dragonLoot = ['gold coin', 'dagger', 'gold coin', 'gold coin', 'ruby']**

**inv = addtoinventory(inv, dragonLoot)**

**displayinventory(inv)**

**output:**

**gold coin : 45**

**ruby : 1**

**rope : 1**

**Total number of items: 47**

**inv = {'gold coin': 42, 'rope': 1}**

**dragonLoot = ['gold coin', 'dagger', 'gold coin', 'gold coin', 'ruby']**

**def addToInventory(backpack, added\_items):**

**for i in added\_items:**

**if i in backpack:**

**backpack[i] += 1**

**else:**

**count = 0**

**count += 1**

**backpack[i] = count**

**return backpack**

**ASSIGNMENT\_07**

Write a function named printTable() that takes a list of lists of strings and displays it in a well-organized table with each column right-justified.

Assume that all the inner lists will contain the same number of strings.

For example, the value could look like this:

tableData = [['apples', 'oranges', 'cherries', 'banana'],

['Alice', 'Bob', 'Carol', 'David'],

['dogs', 'cats', 'moose', 'goose']]

Your printTable() function would print the following:

apples Alice dogs

oranges Bob cats

cherries Carol moose

banana David goose

Hint: Your code will first have to find the longest string in each of the inner lists so that the whole column can be wide enough to fit all the strings. You can store the maximum width of each column as a list of integers. The printTable() function can begin with colWidths = [0] \* len(tableData), which will create a list containing the same number of 0 values as the number of inner lists in tableData. That way, colWidths[0] can store the width of the longest string in tableData[0], colWidths[1] can store the width of the longest string in tableData[1], and so on. You can then find the largest value in the colWidths list to find out what integer width to pass to the rjust() string

method.

**# orginal data**

**tableData=[['apples', 'oranges', 'cherries', 'banana'],**

**['Alice', 'Bob', 'Carol', 'David'],**

**['dogs', 'cats', 'moose', 'goose']]**

**# empty dictonary for sorting the data**

**newTable = {0:[], 1:[], 2:[], 3:[]}**

**# iterate through each list in tableData**

**for li in tableData:**

**for i in range(len(li)):**

**# put each item of tableData into newTable by index**

**newTable[i].append(li[i])**

**# determine the longest list by number of total characters**

**# for instance ['apples', 'Alice', 'dogs'] would be 15 characters**

**# we will start with longest being zero at the start**

**longest = 0**

**# iterate through newTable**

**# for example the first key:value will be 0:['apples', 'Alice', 'dogs']**

**# we only really care about the value (the list) in this case**

**for key, value in newTable.items():**

**# determine the total characters in each list**

**# so effectively len('applesAlicedogs') for the first list**

**length = len(''.join(value))**

**# if the length is the longest length so far,**

**# make that equal longest**

**if length > longest:**

**longest = length**

**for key, value in newTable.items():**

**print(' ' \* (longest - len(''.join(value))) + ' '.join(value))**

**######################################**

**#!/usr/bin/env python3**

**tableData = [['apples', 'oranges', 'cherries', 'banana'],**

**['Alice', 'Bob', 'Carol', 'David'],**

**['dogs', 'cats', 'moose','goose']]**

**def printTable():**

**colWidths = [0] \* len(tableData)**

**# find longest word in each list, convert to int**

**# and add to colWidths var**

**for i in range(len(tableData)):**

**for l in tableData[i]:**

**if len(l) >= colWidths[i]:**

**colWidths[i] = len(l)**

**# print and justify using the values from colWidths + 1**

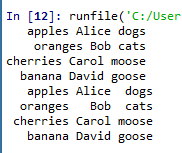
**for t in range(4):**

**print(tableData[0][t].rjust(colWidths[0]+1) + \**

**tableData[1][t].rjust(colWidths[1]+1) + \**

**tableData[2][t].rjust(colWidths[2]+1))**

**printTable()**



**AUTOMATION**

**Assignment-01**

Strong Password Detection

Write a function that uses regular expressions to make sure the password string it is passed is strong. A strong password is defined as one that is at least eight characters long, contains both uppercase and lowercase characters, and has at least one digit. You may need to test the string against multiple regex patterns to validate its strength.

**import re**

**print('Please set a new password: ')**

**def strongpassword():**

**while True:**

**password = input()**

**if lowcase.search(password) == None:**

**print('The entered password doesn\'t have a lower case character')**

**continue**

**if upcase.search(password) == None:**

**print('The entered password doesn\'t have an upper case character')**

**continue**

**if digit.search(password) == None:**

**print('The entered password doesn\'t have a digit')**

**continue**

**if space\_8.search(password) == None:**

**print('The entered password should have atleast 8 characters and no space in between')**

**continue**

**else:**

**print('New Password is Valid and Saved')**

**break**

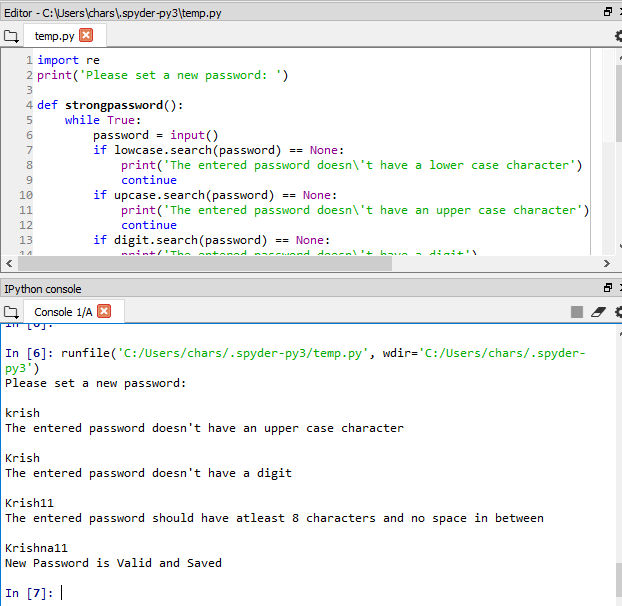
**lowcase = re.compile(r'[a-z]') # this regex searches for atleast one lower case alphabet**

**upcase = re.compile(r'[A-Z]') # this regex searches for atleast one upper case alphabet**

**digit = re.compile(r'(\d)') # this regex searches for atleast one digit**

**space\_8 = re.compile(r'^[a-zA-Z0-9]{8,}$') # this regex searches for expressions without any space and atleast 8 characters**

**strongpassword()**



**Assignment-02**

Regex Version of strip()

Write a function that takes a string and does the same thing as the strip() string method. If no other arguments are passed other than the string to strip, then whitespace characters will be removed from the beginning and end of the string. Otherwise, the characters specified in the second argument to the function will be removed from the string.

**import re**

**def regex\_strip(s, char=None):**

**if not char:**

**strip\_left = re.compile(r'^\s\*') #string starting with whitespace**

**strip\_right = re.compile(r'\s\*$') #string ending with whitespace**

**s = re.sub(strip\_left, "", s) #replacing strip\_left with "" in string s**

**s = re.sub(strip\_right, "", s) #replacing strip\_right with "" in string s**

**else:**

**strip\_char = re.compile(char)**

**s = re.sub(strip\_char, "", s)**

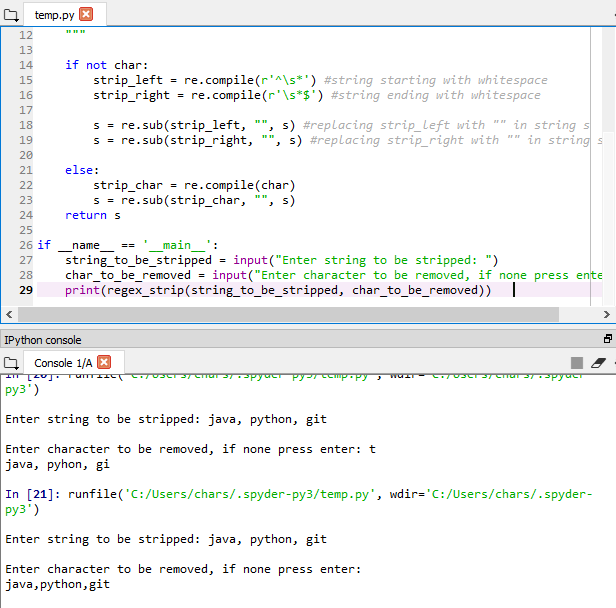
**return s**

**if \_\_name\_\_ == '\_\_main\_\_':**

**string\_to\_be\_stripped = input("Enter string to be stripped: ")**

**char\_to\_be\_removed = input("Enter character to be removed, if none press enter: ")**

**print(regex\_strip(string\_to\_be\_stripped, char\_to\_be\_removed))**



**Assignment-03**

Create a Mad Libs program that reads in text files and lets the user add their own text anywhere the word ADJECTIVE, NOUN, ADVERB, or VERB appears in the text file.

For example, a text file may look like this:

The ADJECTIVE panda walked to the NOUN and then VERB. A nearby NOUN was unaffected by these events.

The program would find these occurrences and prompt the user to replace them.

Enter an adjective:

silly

Enter a noun:

chandelier

Enter a verb:

screamed

Enter a noun:

pickup truck

The following text file would then be created:

The silly panda walked to the chandelier and then screamed. A nearby pickup truck was unaffected by these events.

The results should be printed to the screen and saved to a new text file.

**import os, re**

**def start():**

**fileName = input('Enter the file name: ')**

**exist\_fileName = os.path.abspath(fileName)**

**if os.path.exists(exist\_fileName) == True:**

**text\_file = open(exist\_fileName)**

**text\_content = text\_file.read()**

**text\_file.close()**

**print(text\_content)**

**text\_regex = re.compile(r'ADJECTIVE|NOUN|VERB|ADVERB')**

**match\_text = text\_regex.findall(text\_content)**

**for match in match\_text:**

**user\_input = input('Enter ' + match + ': ')**

**text\_content = text\_content.replace(match, user\_input, 1)**

**print(text\_content)**

**new\_fileName = 'new\_' + fileName**

**new\_file = open(new\_fileName, 'w')**

**new\_file.write(text\_content)**

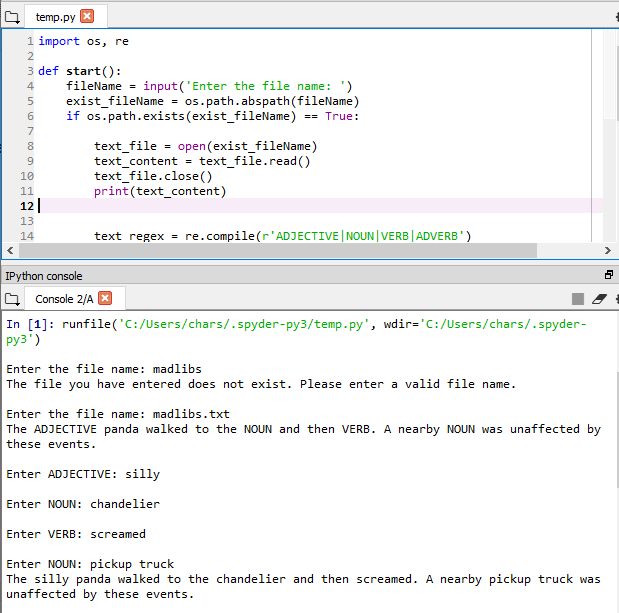
**new\_file.close()**

**else:**

**print('The file you have entered does not exist. Please enter a valid file name.')**

**start()**

**start()**



**Assignment-04**

Write a program that opens all .txt files in a folder and searches for any line that matches a user-supplied regular expression. The results should be printed to the screen.

**def main():**

**import os, re**

**# Get list of all .txt files**

**all\_files = os.listdir("./") # use current working directory**

**text\_files = []**

**for file in all\_files:**

**if file.endswith(".txt"):**

**text\_files.append(file)**

**# Get regular expression**

**regex = input("Enter regular expression to search for: ")**

**search\_regex = re.compile(regex)**

**# Open .txt file**

**for file in text\_files:**

**input\_file = open(file)**

**input\_content = input\_file.readlines()**

**input\_file.close()**

**# Search for regex in file**

**for line in input\_content:**

**match\_objects = search\_regex.findall(line)**

**if match\_objects is not None:**

**# Print result**

**for match in match\_objects:**

**print(match)**

**if \_\_name\_\_ == '\_\_main\_\_':**

**main()**

**Assignment-05**

Selective Copy:

Write a program that walks through a folder tree and searches for files with a certain file extension (such as .pdf or .jpg). Copy these files from whatever location they are in to a new folder.

**import os, shutil**

**sourcePath = input("Enter the absolute path of the source folder: ")**

**fileExtType = input("Enter the type of file to copy: ").lower()**

**destPath = input("Enter the absolute path of the destination folder: ")**

**for foldername, subfolders, filenames in os.walk(sourcePath):**

**for filename in filenames:**

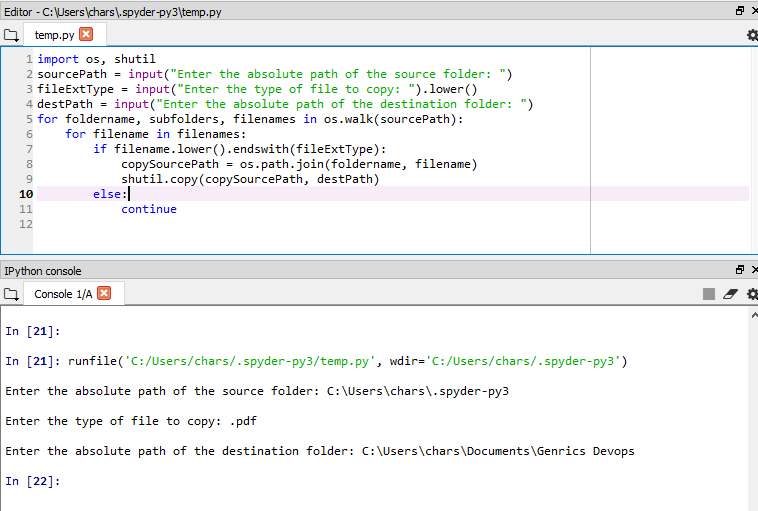
**if filename.lower().endswith(fileExtType):**

**copySourcePath = os.path.join(foldername, filename)**

**shutil.copy(copySourcePath, destPath)**

**else:**

**continue**



Deleting Unneeded Files:

It’s not uncommon for a few unneeded but humongous files or folders to take up the bulk of the space on your hard drive. If you’re trying to free up room on your computer, you’ll get the most bang for your buck by deleting the most massive of the unwanted files. But first you have to find them.

Write a program that walks through a folder tree and searches for exceptionally large files or folders—say, ones that have a file size of more than 100MB. (Remember, to get a file’s size, you can use os.path.getsize() from the os module.) Print these files with their absolute path to the screen

Filling in the Gaps:

Write a program that finds all files with a given prefix, such as spam001.txt, spam002.txt, and so on, in a single folder and locates any gaps in the numbering (such as if there is a spam001.txt and spam003.txt but no spam002.txt).

Have the program rename all the later files to close this gap.

As an added challenge, write another program that can insert gaps into numbered files so that a new file can be added

**Assignment-06**

Debugging a Program:-

The following program is meant to be a simple coin toss guessing game. The player gets two guesses (it’s an easy game).

However, the program has several bugs in it. Run through the program a few times to find the bugs that keep the program from working correctly.

import random

guess = ''

while guess not in ('heads', 'tails'):

print('Guess the coin toss! Enter heads or tails:')

guess = input()

toss = random.randint(0, 1) # 0 is tails, 1 is heads

if toss == guess:

print('You got it!')

else:

print('Nope! Guess again!')

guesss = input()

if toss == guess:

print('You got it!')

else:

print('Nope. You are really bad at this game.')

