|  |
| --- |
| Evidence 1 |
| # task 1.1  ### MAIN PROGRAM ###  def main():  freq\_store = [] # store as array of tuples (x\_value, freq)  counter = 0 # counts number of X values  while True: # repeats until ZZZ is typed in  x\_value = input("Next X value ... <ZZZ to END> ")  if x\_value == "ZZZ":  break # exit out of loop and process  if counter == 6:  break # exit out of loop and process  # ask for frequency  while True:  freq = input("Frequency ... ")  if freq.isdigit(): # is a number  freq = int(freq)  if freq >= 0 and freq <= 60: # within range  break # break out of loop  else:  # print error message  print("Error: Frequency entered out of range [0, 60]")  print("Enter an integer between 0 and 60.")  print()  else:  # print error message  print("Error: Frequency entered not an integer.")  print("Enter an integer between 0 and 60.")  print()  freq\_store.append((x\_value, freq)) # add to freq\_store array  counter += 1 # increment counter  ### DATA PROCESSING ###  print() # heading  print("+" \* 30)  print("Frequency distribution")  print("+" \* 30)  for data\_tuple in freq\_store:  x\_value = data\_tuple[0]  freq = data\_tuple[1]  print(" {0:<10}{1}".format(x\_value, "@" \* freq)) |
| Evidence 2 |
|  |
| Evidence 3 |
| # task 1.2  ### MAIN PROGRAM ###  def main():  freq\_store = [] # store as array of tuples (x\_value, freq)  max\_freq = 0 # maximum frequency (width: 80)  counter = 0 # counts number of X values  while True: # repeats until ZZZ is typed in  x\_value = input("Next X value ... <ZZZ to END> ")  if x\_value == "ZZZ":  break # exit out of loop and process  if counter == 6:  break # exit out of loop and process  # ask for frequency  while True:  freq = input("Frequency ... ")  if freq.isdigit(): # is a number  freq = int(freq)  if freq >= 0 and freq <= 60: # within range  break # break out of loop  else:  # print error message  print("Error: Frequency entered out of range [0, 60]")  print("Enter an integer between 0 and 60.")  print()  else:  # print error message  print("Error: Frequency entered not an integer.")  print("Enter an integer between 0 and 60.")  print()    if freq > max\_freq:  max\_freq = freq # set maximum frequency    freq\_store.append((x\_value, freq)) # add to freq\_store array  counter += 1 # increment counter  ### DATA PROCESSING ###  print() # heading  print("+" \* 30)  print(" Frequency distribution")  print("+" \* 30)  print()  for data\_tuple in freq\_store:  x\_value = data\_tuple[0]  freq = data\_tuple[1]    # print five lines each (6 \* (5 lines + 1 space) + 4 for heading = 40)  print(" {0:<16}{1}".format("", chr(9608) \* freq))  print(" {0:<16}{1}".format("", chr(9608) \* freq))  print(" {0:<16}{1}".format(x\_value, chr(9608) \* freq))  print(" {0:<16}{1}".format("", chr(9608) \* freq))  print(" {0:<16}{1}".format("", chr(9608) \* freq))  # chr(9608) is a shaded box character.  # print newline to separate X values  print() |
| Evidence 4 |
|  |
| Evidence 5 |
| # task 1.3  ### MAIN PROGRAM ###  def main():  freq\_store = [] # store as array of tuples (x\_value, freq)  max\_freq = 0 # maximum frequency (width: 80)  counter = 0 # counts number of X values  while True: # repeats until ZZZ is typed in  x\_value = input("Next X value ... <ZZZ to END> ")  if x\_value == "ZZZ":  break # exit out of loop and process  if counter == 6:  break # exit out of loop and process  # ask for frequency  while True:  freq = input("Frequency ... ")  if freq.isdigit(): # is a number  freq = int(freq)  if freq >= 0: # within range  break # break out of loop  else:  # print error message  print("Error: Frequency entered must be 0 or a positive integer.")  print()  else:  # print error message  print("Error: Frequency entered must be 0 or a positive integer.")  print()    if freq > max\_freq:  max\_freq = freq # set maximum frequency    freq\_store.append((x\_value, freq)) # add to freq\_store array  counter += 1 # increment counter  ### DATA PROCESSING ###  print() # heading  print("+" \* 30)  print(" Frequency distribution")  print("+" \* 30)  print()    # 60 = 80 total - 20 used for labelling at the start of the line  # scaling will only be used if max\_freq > 60.  if max\_freq > 60:  freq\_scale = 60 / max\_freq  else:  freq\_scale = 1 # normal scale  for data\_tuple in freq\_store:  x\_value = data\_tuple[0]  freq = data\_tuple[1]    # print five lines each (6 \* (5 lines + 1 space) + 4 for heading = 40)  print(" {0:<16}{1}".format("", chr(9608) \* round(freq \* freq\_scale)))  print(" {0:<16}{1}".format("", chr(9608) \* round(freq \* freq\_scale)))  print(" {0:<16}{1}".format(x\_value, chr(9608) \* round(freq \* freq\_scale)))  print(" {0:<16}{1}".format("", chr(9608) \* round(freq \* freq\_scale)))  print(" {0:<16}{1}".format("", chr(9608) \* round(freq \* freq\_scale)))  # chr(9608) is a shaded box character.  # print newline to separate X values  print()  # print horizontal axis  print("{0:<19}|{1}".format("", "---------|" \* 6))  h\_labels = "{0:16}".format("") # a string containing labels  # if scaling is required  if max\_freq > 60:  for i in range(7): # from 0 to 6 inclusive  h\_labels += "{0:>6.2f}{1:<4}".format(max\_freq \* i / 6, "")  else:  for i in range(7): # from 0 to 6 inclusive  h\_labels += "{0:>6.2f}{1:<4}".format(60 \* i / 6, "")  print(h\_labels) |
| Evidence 6 |
| Dataset 1 |
| Dataset 2 |
| Dataset 3 |