**Module 4 Graded Assessment**

**LATEST SUBMISSION GRADE**

80%

1.Question 1

The format\_address function separates out parts of the address string into new strings: house\_number and street\_name, and returns: "house number X on street named Y". The format of the input string is: numeric house number, followed by the street name which may contain numbers, but never by themselves, and could be several words long. For example, "123 Main Street", "1001 1st Ave", or "55 North Center Drive". Fill in the gaps to complete this function.

**1 / 1 point**

def format\_address(address\_string):

  # Declare variables

  x=''

  # Separate the address string into parts

  address\_string=address\_string.split()

  # Traverse through the address parts

    # Determine if the address part is the

    # house number or part of the street name

  i=1

  while i < len(address\_string):

    x+=address\_string[i]+' '

    i=i+1

  # Does anything else need to be done

  # before returning the result?

  # Return the formatted string

  return "house number {} on street named {}".format(address\_string[0],x)

print(format\_address("123 Main Street"))

# Should print: "house number 123 on street named Main Street"

print(format\_address("1001 1st Ave"))

# Should print: "house number 1001 on street named 1st Ave"

print(format\_address("55 North Center Drive"))

# Should print "house number 55 on street named North Center Drive"

house number 123 on street named Main Street

house number 1001 on street named 1st Ave

house number 55 on street named North Center Drive

**Correct**

Great work! You've remembered how to work with string

methods and use variables for formatting output

2.Question 2

The highlight\_word function changes the given word in a sentence to its upper-case version. For example, highlight\_word("Have a nice day", "nice") returns "Have a NICE day". Can you write this function in just one line?

**1 / 1 point**

def highlight\_word(sentence, word):

    return(sentence.replace(word,word.upper()))

print(highlight\_word("Have a nice day", "nice"))

print(highlight\_word("Shhh, don't be so loud!", "loud"))

print(highlight\_word("Automating with Python is fun", "fun"))

RunReset

Have a NICE day

Shhh, don't be so LOUD!

Automating with Python is FUN

**Correct**

Nice job! You're mastering your string skills!

3.Question 3

A professor with two assistants, Jamie and Drew, wants an attendance list of the students, in the order that they arrived in the classroom. Drew was the first one to note which students arrived, and then Jamie took over. After the class, they each entered their lists into the computer and emailed them to the professor, who needs to combine them into one, in the order of each student's arrival. Jamie emailed a follow-up, saying that her list is in reverse order. Complete the steps to combine them into one list as follows: **the contents of Drew's list, followed by Jamie's list in reverse order**, to get an accurate list of the students as they arrived.

**1 / 1 point**

def combine\_lists(list1, list2):

  # Generate a new list containing the elements of list2

  # Followed by the elements of list1 in reverse order

  i=-1

  for value in list1:

    list2.append(list1[i])

    i=i-1

  return list2

Jamies\_list = ["Alice", "Cindy", "Bobby", "Jan", "Peter"]

Drews\_list = ["Mike", "Carol", "Greg", "Marcia"]

print(combine\_lists(Jamies\_list, Drews\_list))

RunReset

['Mike', 'Carol', 'Greg', 'Marcia', 'Peter', 'Jan', 'Bobby', 'Cindy', 'Alice']

**Correct**

Excellent! You're using the list functions correctly, and it

shows!

4.Question 4

Use a list comprehension to create a list of squared numbers (n\*n). The function receives the variables *start*and *end*, and returns a list of squares of consecutive numbers between *start*and *end***inclusively**. For example, squares(2, 3) should return [4, 9].

**1 / 1 point**

def squares(start, end):

    return [ n\*n for n in range(start,end+1)]

print(squares(2, 3)) # Should be [4, 9]

print(squares(1, 5)) # Should be [1, 4, 9, 16, 25]

print(squares(0, 10)) # Should be [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

RunReset

[4, 9]

[1, 4, 9, 16, 25]

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

**Correct**

Right on! You're making the correct calculation, and using

the correct range.

5.Question 5

Complete the code to iterate through the keys and values of the car\_prices dictionary, printing out some information about each one.

**1 / 1 point**

def car\_listing(car\_prices):

  result = ""

  for key,value in car\_prices.items():

    result += "{} costs {} dollars".format(key,value) + "\n"

  return result

print(car\_listing({"Kia Soul":19000, "Lamborghini Diablo":55000, "Ford Fiesta":13000, "Toyota Prius":24000}))

RunReset

Kia Soul costs 19000 dollars

Lamborghini Diablo costs 55000 dollars

Ford Fiesta costs 13000 dollars

Toyota Prius costs 24000 dollars

**Correct**

You got it! You've correctly gone through the items of the

dictionary!

6.Question 6

Taylor and Rory are hosting a party. They sent out invitations, and each one collected responses into dictionaries, with names of their friends and how many guests each friend is bringing. Each dictionary is a partial list, but Rory's list has more current information about the number of guests. Fill in the blanks to combine both dictionaries into one, with each friend listed only once, and the number of guests from Rory's dictionary taking precedence, if a name is included in both dictionaries. Then print the resulting dictionary.

**0 / 1 point**

  # Combine both dictionaries into one, with each key listed

  # only once, and the value from guests1 taking precedence

  for key,value in  guests2.items():

Rorys\_guests = { "Adam":2, "Brenda":3, "David":1, "Jose":3, "Charlotte":2, "Terry":1, "Robert":4}

Taylors\_guests = { "David":4, "Nancy":1, "Robert":2, "Adam":1, "Samantha":3, "Chris":5}

    if key in guests1.keys():

      guests1[key]=value+guests1[key]

def combine\_guests(guests1, guests2):

    else:

      guests1.update(key=value)

  return guests1

print(combine\_guests(Rorys\_guests, Taylors\_guests))

RunReset

{'Adam': 3, 'Brenda': 3, 'David': 5, 'Jose': 3, 'Charlotte': 2, 'Terry': 1, 'Robert': 6, 'key': 5}

**Incorrect**

Not quite. What dictionary method updates the dictionary

with the items coming from the other dictionary, replacing

existing entries and adding new entries?

7.Question 7

Use a dictionary to count the frequency of letters in the input string. Only letters should be counted, not blank spaces, numbers, or punctuation. Upper case should be considered the same as lower case. For example, count\_letters("This is a sentence.") should return {'t': 2, 'h': 1, 'i': 2, 's': 3, 'a': 1, 'e': 3, 'n': 2, 'c': 1}.

**0 / 1 point**

def count\_letters(text):

  result = {}

  # Go through each letter in the text

  for letter in text:

    # Check if the letter needs to be counted or not

   if letter not in result:

     print(letter.lower())

    # Add or increment the value in the dictionary

print(count\_letters("AaBbCc"))

# Should be {'a': 2, 'b': 2, 'c': 2}

print(count\_letters("Math is fun! 2+2=4"))

# Should be {'m': 1, 'a': 1, 't': 1, 'h': 1, 'i': 1, 's': 1, 'f': 1, 'u': 1, 'n': 1}

print(count\_letters("This is a sentence."))

# Should be {'t': 2, 'h': 1, 'i': 2, 's': 3, 'a': 1, 'e': 3, 'n': 2, 'c': 1}

RunReset

a

a

b

b

c

c

None

m

a

t

h

i

s

f

u

n

!

2

+

2

=

4

None

t

h

i

s

i

s

a

s

e

n

t

e

n

c

e

.

None

**Incorrect**

Not quite. Are you using the correct string commands to

convert the text to lower case, and check that each letter

is an alphabetic character, before checking if it's part of

the existing dictionary?

8.Question 8

What do the following commands return when animal = "Hippopotamus"?

>>> print(animal[3:6])

>>> print(animal[-5])

>>> print(animal[10:])

**1 / 1 point**



ppo, t, mus



ppop, o, s



pop, t, us



popo, t, mus

**Correct**

You got it! When both parts of a string index range are included, the substring starts at first index and ends at second index minus 1. When the index is negative, the character is counted from the end of the string. When the second index is omitted, it goes until the end of the string.

9.Question 9

What does the list "colors" contain after these commands are executed?

colors = ["red", "white", "blue"]

colors.insert(2, "yellow")

**1 / 1 point**



['red', 'white', 'yellow', 'blue']



['red', 'yellow', 'white', 'blue']



['red', 'yellow', 'blue']



['red', 'white', 'yellow']

**Correct**

Right on! The insert command inserts the new element into the list at the specified index, shifting the other elements over afterwards.

10.Question 10

What do the following commands return?

host\_addresses = {"router": "192.168.1.1", "localhost": "127.0.0.1", "google": "8.8.8.8"}

host\_addresses.keys()

**1 / 1 point**



{"router": "192.168.1.1", "localhost": "127.0.0.1", "google": "8.8.8.8"}



["router", "192.168.1.1", "localhost", "127.0.0.1", "google", "8.8.8.8"]



['192.168.1.1', '127.0.0.1', '8.8.8.8']



['router', 'localhost', 'google']

**Correct**

You got it! In dictionaries, the keys() command returns a list of just the keys, which is what this is.