← Back **Module 4 Graded Assessment**
Graded Quiz • 50 min

English

**Due** Dec 24, 11:59 PM EST

## Congratulations! You passed!

**Grade received** 100%

**Latest Submission Grade** 100%

**To pass** 80% or higher

Go to next item

1. Fill in the blank to complete the “first_character” function. This function should return the first character of any string passed in. Complete the string operation needed in this function so that input like "Hello, World" will produce the output "H".

1 / 1 point

```python
def first_character(string):
    # Complete the return statement using a string operation.
    return string[0]


print(first_character("Hello, World")) # Should print H
print(first_character("Python is awesome")) # Should print P
print(first_character("Keep going")) # Should print K
```

Run

Reset

```
H
P
K
```

Correct

Correct

2. Fill in the blank to complete the “string_words” function. This function should split up the words in the given “string” and return the number of words in the “string”. Complete the string operation and method needed in this function so that a function call like "string_words("Hello, World")" will return the output "2".

1 / 1 point

```python
def string_words(string):
    result = string.split()
    number_of_string = len(result)
    # Complete the return statement using both a string operation and
    # a string method in a single line.
    return number_of_string


print(string_words("Hello, World")) # Should print 2
print(string_words("Python is awesome")) # Should print 3
print(string_words("Keep going")) # Should print 2
print(string_words("Have a nice day")) # Should print 4
```

Run

Reset

```
2
3
2
4
```

Correct

Correct

3. Consider the following scenario about using Python lists:

1 / 1 point

Employees at a company shared  the distance they drive to work (in miles) through an online survey. These distances were automatically added by Python to a list called “distances” in the order that each employee submitted their distance. Management wants the list to be sorted in the order of the longest distance to the shortest distance.

Complete the function to sort the “distances” list. This function should:

1. sort the given “distances” list, passed through the function’s parameters; ;
2. reverse the sort order so that it goes from the longest to the shortest distance;
3. return the modified “distances” list.

```python
def sort_distance(distances):
    distances.sort() # Sort the list
    distances.reverse() # Reverse the order of the list
    return distances


print(sort_distance([2,4,0,15,8,9]))
# Should print [15, 9, 8, 4, 2, 0]
```

Run

Reset

```
[15, 9, 8, 4, 2, 0]
```

Correct

Correct

**4.** Fill in the blank to complete the "even_numbers" function. This function should use a list comprehension to create a list of even numbers using a conditional if statement with the modulo operator to test for numbers evenly divisible by 2. The function receives two variables and should return the list of even numbers that occur between the "first" and "last" variables **exclusively** *(meaning don't modify the default behavior of the range to exclude the "end" value in the range)*. For example, even_numbers(2, 7) should return [2, 4, 6].

1 / 1 point

```
def even_numbers(first, last):
  return [ x for x in range(first,last) if x%2==0]


print(even_numbers(4, 14)) # Should print [4, 6, 8, 10, 12]
print(even_numbers(0, 9))  # Should print [0, 2, 4, 6, 8]
print(even_numbers(2, 7))  # Should print [2, 4, 6]
```

Run

Reset

```
[4, 6, 8, 10, 12]
[0, 2, 4, 6, 8]
[2, 4, 6]
```

✓ **Correct**

Correct

**5.** Fill in the blanks to complete the "car_listing" function. This function accepts a "car_prices" dictionary. It should iterate through the keys (car models) and values (car prices) in that dictionary. For each item pair, the function should format a string so that a dictionary entry like ""Kia Soul":19000" will print "A Kia Soul costs 19000 dollars". Each new string should appear on its own line.

1 / 1 point

```
def car_listing(car_prices):
  result = ""
  # Complete the for loop to iterate through the key and value items
  # in the dictionary.
  for car, price in car_prices.items():
    result += (f"A {car} costs {price} dollars\n") # Use a string method to format the
  return result

print(car_listing({"Kia Soul":19000, "Lamborghini Diablo":55000, "Ford Fiesta":13000, "

# Should print:
# A Kia Soul costs 19000 dollars
# A Lamborghini Diablo costs 55000 dollars
# A Ford Fiesta costs 13000 dollars
# A Toyota Prius costs 24000 dollars
```

Run

Reset

```
A Kia Soul costs 19000 dollars
A Lamborghini Diablo costs 55000 dollars
A Ford Fiesta costs 13000 dollars
A Toyota Prius costs 24000 dollars
```

✓ **Correct**

Correct

**6.** Consider the following scenario about using Python dictionaries:

1 / 1 point

Tessa and Rick are hosting a party. Both sent out invitations to their friends, and each one collected responses into dictionaries, with names of their friends and how many guests each friend was bringing. Each dictionary is a partial guest list, but Rick's guest list has more current information about the number of guests.

Complete the function to combine both dictionaries into one, with each friend listed only once, and the number of guests from Rick's dictionary taking precedence, if a name is included in both dictionaries. Then print the resulting dictionary. This function should:

1. accept two dictionaries through the function's parameters;
2. combine both dictionaries into one, with each key listed only once;
3. the values from the "guests1" dictionary taking precedence, if a key is included in both dictionaries;
4. then print the new dictionary of combined items.

```
;ts1, guests2):
_guests) # Use a dictionary method to combine the dictionaries.


':2, "Camila":3, "David":1, "Jamal":3, "Charley":2, "Titus":1, "Raj":4}
.d":4, "Noemi":1, "Raj":2, "Adam":1, "Sakira":3, "Chidi":5}

.cks_guests, Tessas_guests))

 1, 'Raj': 4, 'Adam': 2, 'Sakira': 3, 'Chidi': 5, 'Camila': 3, 'Jamal': 3, 'Charley': 2, 'Titus' 1}
```

Run

Reset

```
David': 1, 'Noemi': 1, 'Raj': 4, 'Adam': 2, 'Sakira': 3, 'Chidi': 5, 'Camila': 3, 'Jamal': 3, 'Charley': 2, 'Titus': 1}
```

✓ **Correct**

Correct

**7.** Consider the following scenario about using Python dictionaries:

1 / 1 point

A teacher is using a dictionary to store student grades. The grades are stored as a point value out of 100. Currently, the teacher has a dictionary setup for Term 1 grades and wants to duplicate it for Term 2. The student name keys in the dictionary should stay the same, but the grade values should be reset to 0.

Complete the "setup_gradebook" function so that input like "{"James": 93, "Felicity": 98, "Barakaa": 80}" will produce a resulting dictionary that contains "{"James": 0, "Felicity": 0, "Barakaa": 0}". This function should:

1. accept a dictionary "old_gradebook" variable through the function's parameters;
2. make a copy of the "old_gradebook" dictionary;
3. iterate over each key and value pair in the new dictionary;
4. replace the value for each key with the number 0;
5. return the new dictionary.

```
def setup_gradebook(old_gradebook):
  # Use a dictionary method to create a new copy of the "old_gradebook".
  new_gradebook = old_gradebook.copy()
    # Complete the for loop to iterate over the new gradebook.
  for student in new_gradebook.keys():
      # Use a dictionary operation to reset the grade values to 0.
      new_gradebook[student] = 0
  return new_gradebook

fall_gradebook = {"James": 93, "Felicity": 98, "Barakaa": 80}
print(setup_gradebook(fall_gradebook))
# Should output {'James': 0, 'Felicity': 0, 'Barakaa': 0}

```

Run

Reset

```
{'James': 0, 'Felicity': 0, 'Barakaa': 0}
```

✓ **Correct**

Correct

**8.** What do the following commands return when genre = "transcendental"?

1 / 1 point

```
print(genre[:-8])
print(genre[-7:9])
```

○ dental, trans

○ endental, tr

○ ran, ental

● transc, nd

✓ **Correct**

**9.** What does the list "music_genres" contain after these commands are executed?

1 / 1 point

```
music_genres = ["rock", "pop", "country"]
music_genres.append("blues")
```

○ ['rock', 'pop', 'blues']

○ ['rock', 'blues', 'country']

○ ['rock', 'blues', 'pop', 'country']

● ['rock', 'pop', 'country', 'blues']

✓ **Correct**

**10.** What do the following commands return?

1 / 1 point

```
speed_limits = {"street": 35, "highway": 65, "school": 15}
speed_limits["highway"]
```

○ [65]

○ {"highway": 65}

○ ["highway", 65]

● 65

✓ **Correct**