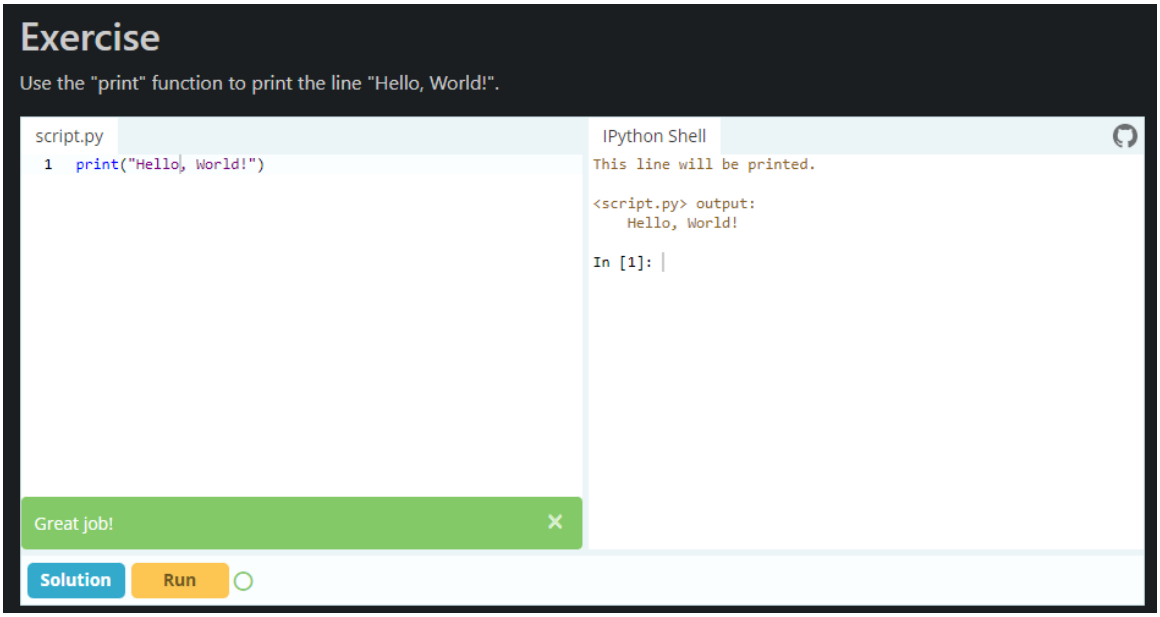
Assignment 1

Gavin Kerr
B00801584

January 13, 2023



Exercise

The target of this exercise is to create a string, an integer, and a floating point number. The string should be named `mystring` and should contain the word "hello". The floating point number should be named `myfloat` and should contain the number 10.0, and the integer should be named `myint` and should contain the number 20.

Exercise

In this exercise, you will need to add numbers and strings to the correct lists using the "append" list method. You must add the numbers 1,2, and 3 to the "numbers" list, and the words 'hello' and 'world' to the strings variable.

You will also have to fill in the variable second_name with the second name in the names list, using the brackets operator []. Note that the index is zero-based, so if you want to access the second item in the list, its index will be 1.

script.py

```
1   numbers = [1,2,3]
2   strings = []
3   names = ["John", "Eric", "Jessica"]
4   strings = [name for name in names]
5   # write your code here
6   second_name = names[1]
7
8
9   # this code should write out the filled arrays and the
    second name in the names list (Eric).
10  print(numbers)
11  print(strings)
12  print("The second name on the names list is %s" %
    second_name)
```

Great Job!

Solution Run

IPython Shell

```
<script.py> output:
    [1, 2, 3]
    ['John', 'Eric', 'Jessica']
    The second name on the names list is Jessica

<script.py> output:
    [1, 2, 3]
    ['John', 'Eric', 'Jessica']
    The second name on the names list is Eric

In [1]:
```

Exercise

The target of this exercise is to create two lists called `x_list` and `y_list`, which contain 10 instances of the variables `x` and `y`, respectively. You are also required to create a list called `big_list`, which contains the variables `x` and `y`, 10 times each, by concatenating the two lists you have created.

script.py

```
x = object()
y = object()

# TODO: change this code
x_list = [x]*10
y_list = [y]*10
big_list = x_list+y_list

print("x_list contains %d objects" % len(x_list))
print("y_list contains %d objects" % len(y_list))
print("big_list contains %d objects" % len(big_list))

# testing code
if x_list.count(x) == 10 and y_list.count(y) == 10:
    print("Almost there...")
if big_list.count(x) == 10 and big_list.count(y) == 10:
    print("Great!")
```

Good work!

Solution Run

IPython Shell

```
<script.py> output:
    x_list contains 10 objects
    y_list contains 10 objects
    big_list contains 20 objects
    Almost there...
    Great!

In [1]:
```

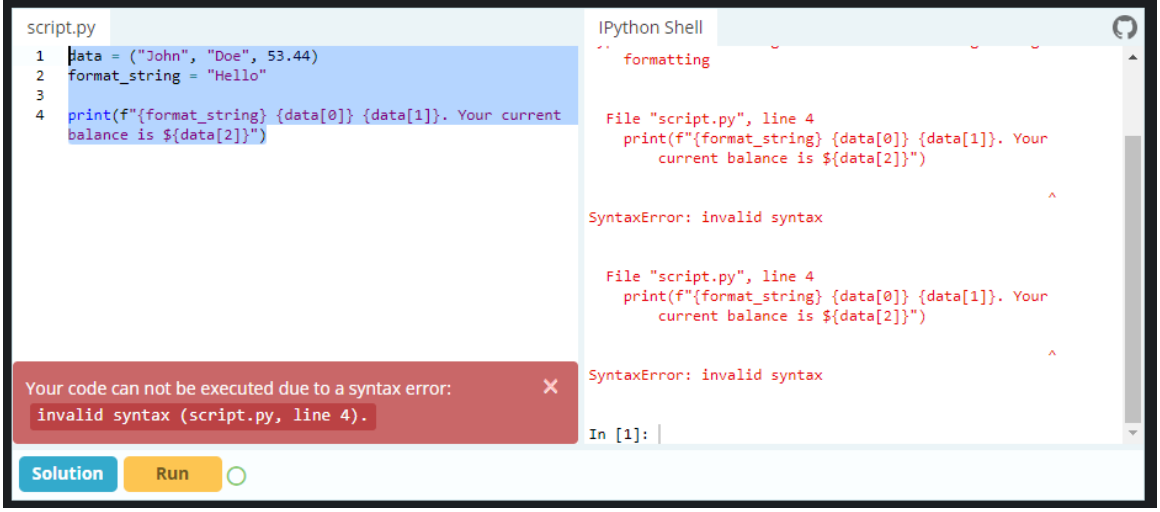
```
data = ("John", "Doe", 53.44)
format_string = "Hello"

print(f"{format_string} {data[0]} {data[1]}. Your current balance is ${data[2]}")
```

✓ 0.3s

```
Hello John Doe. Your current balance is $53.44
```

Exercise

You will need to write a format string which prints out the data using the following syntax: `Hello John Doe. Your current balance is $53.44.`



Exercise

Try to fix the code to print out the correct information by changing the string.

script.py

```
    5th-from-last to end
17
18  # Convert everything to uppercase
19  print("String in uppercase: %s" % s.upper())
20
21  # Convert everything to lowercase
22  print("String in lowercase: %s" % s.lower())
23
24  # Check how a string starts
25  if s.startswith("Str"):
26      print("String starts with 'Str'. Good!")
27
28  # Check how a string ends
29  if s.endswith("ome!"):
30      print("String ends with 'ome!'. Good!")
31
32  # Split the string into three separate strings,
```

Great work!

Solution Run

IPython Shell

```
<script.py> output:
    Length of s = 20
    The first occurrence of the letter a = 8
    a occurs 2 times
    The first five characters are 'Strin'
    The next five characters are 'gs ar'
    The thirteenth character is 'a'
    The characters with odd index are 'tig r wsm!'
    The last five characters are 'some!'
    String in uppercase: STRINGS ARE AWESOME!
    String in lowercase: strings are awesome!
    String starts with 'Str'. Good!
    String ends with 'ome!'. Good!
    Split the words of the string: ['Strings', 'are',
        'awesome!']

In [1]:
```

Exercise

Change the variables in the first section, so that each if statement resolves as True.

script.py

```
4   first_array = [1,2,3]
5   second_array = [1,2]
6
7   if number > 15:
8       print("1")
9
10  if first_array:
11      print("2")
12
13  if len(second_array) == 2:
14      print("3")
15
16  if len(first_array) + len(second_array) == 5:
17      print("4")
18
19  if first_array and first_array[0] == 1:
20      print("5")
```

Great Work!

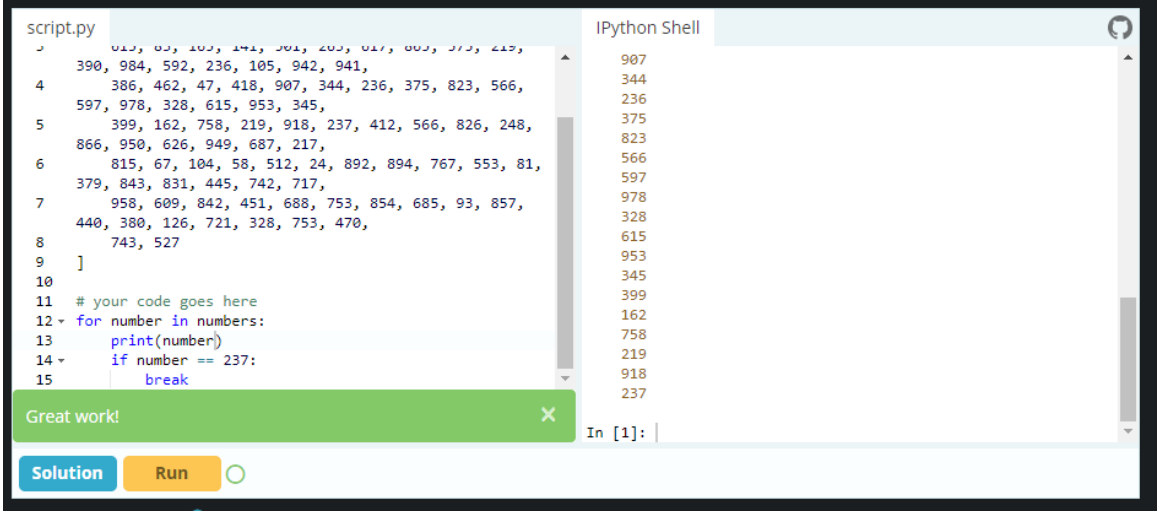
Solution Run

IPython Shell

```
<script.py> output:
    1
    2
    3
    4
    5
    6

In [1]:
```

Exercise

Loop through and print out all even numbers from the numbers list in the same order they are received. Don't print any numbers that come after 237 in the sequence.



```
# Modify this function to return a list of strings as defined above
def list_benefits():
    return ["More organized code", "More readable code", "Easier code reuse", \
        "Allowing programmers to share and connect code together"]

# Modify this function to concatenate to each benefit - " is a benefit of functions!"
def build_sentence(benefit):
    return f"{benefit} is a benefit of functions!"

def name_the_benefits_of_functions():
    list_of_benefits = list_benefits()
    for benefit in list_of_benefits:
        print(build_sentence(benefit))

name_the_benefits_of_functions()
```

[7] ✓ 0.2s

```
More organized code is a benefit of functions!
More readable code is a benefit of functions!
Easier code reuse is a benefit of functions!
Allowing programmers to share and connect code together is a benefit of functions!
```

Exercise

In this exercise you'll use an existing function, and while adding your own to create a fully functional program.

1. Add a function named `list_benefits()` that returns the following list of strings: "More organized code", "More readable code", "Easier code reuse", "Allowing programmers to share and connect code together"
2. Add a function named `build_sentence(info)` which receives a single argument containing a string and returns a sentence starting with the given string and ending with the string " is a benefit of functions!"
3. Run and see all the functions work together!

script.py | solution.py

```
1  # Modify this function to return a list of strings as
   defined above
2  def list_benefits():
3      return ["More organized code", "More readable code",
   "Easier code reuse", "Allowing programmers to share and
   connect code together"]
4
5  # Modify this function to concatenate to each benefit - "
   is a benefit of functions!"
6  def build_sentence(benefit):
7      return f"{benefit} is a benefit of functions!"
8
9  def name_the_benefits_of_functions():
10     list_of_benefits = list_benefits()
11     for benefit in list_of_benefits:
12         print(build_sentence(benefit))
13
14 name_the_benefits_of_functions()
```

IPython Shell

```
  File "script.py", line 7
    return f"{benefit} is a benefit of functions!"
                                                  ^
SyntaxError: invalid syntax


In [1]:
```

Run

Exercise

We have a class defined for vehicles. Create two new vehicles called car1 and car2. Set car1 to be a red convertible worth $60,000.00 with a name of Fer, and car2 to be a blue van named Jump worth $10,000.00.