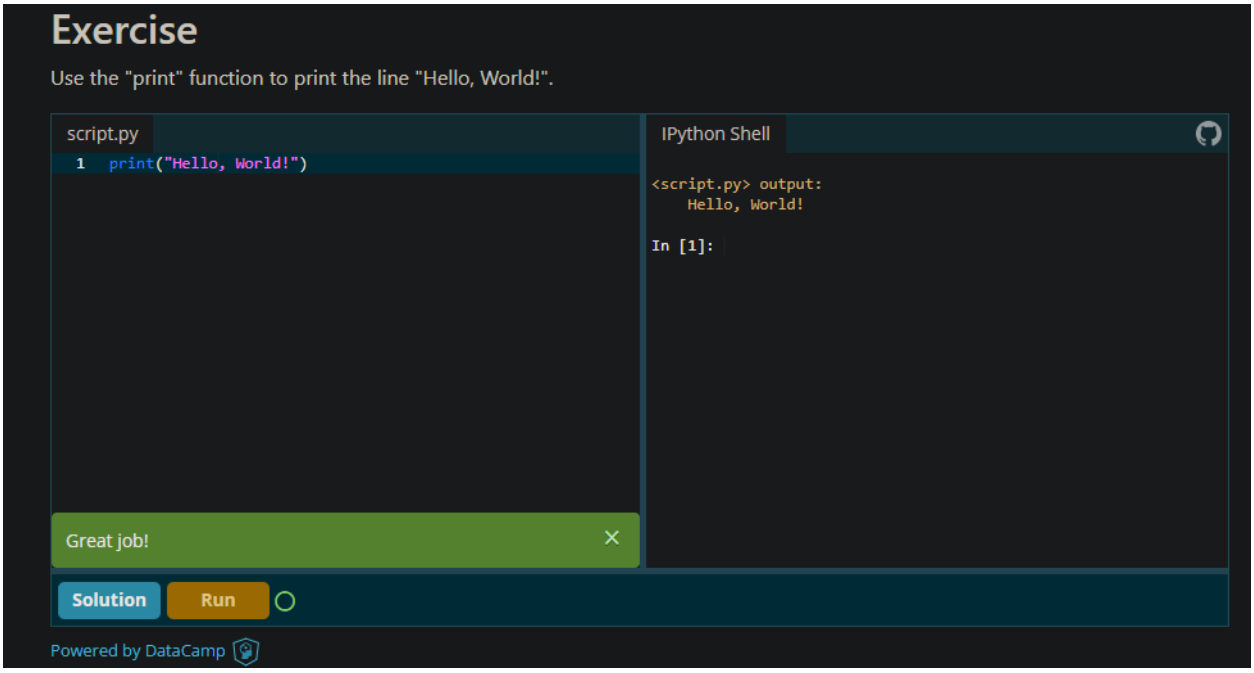

Exercise
Use the "print" function to print the line "Hello, World!".
script.py
1 print("Hello, World!")
IPython Shell
<script.py> output:
Hello, World!
In [1]:
Great job!
Solution
Run
Powered by DataCamp

Exercise

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

script.py

```
# change this code
mystring = "hello"
myfloat = 10.0
myint = 20

# testing code
if mystring == "hello":
    print("String: %s" % mystring)
if isinstance(myfloat, float) and myfloat == 10.0:
    print("Float: %f" % myfloat)
if isinstance(myint, int) and myint == 20:
    print("Integer: %d" % myint)
```

Great job!

IPython Shell

```
<script.py> output:
    Float: 10.000000
    Integer: 20

<script.py> output:
    String: hello
    Float: 10.000000
    Integer: 20

In [1]:
```

Solution | Run | Workspace Ready

Powered by DataCamp

Exercise

In this exercise, you will need to add numbers and strings to the correct lists using the "append" list
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 t
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
5   # write your code here
6   second_name = names[1]
7   strings= 'hello'+ 'world'
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!

IPython Shell

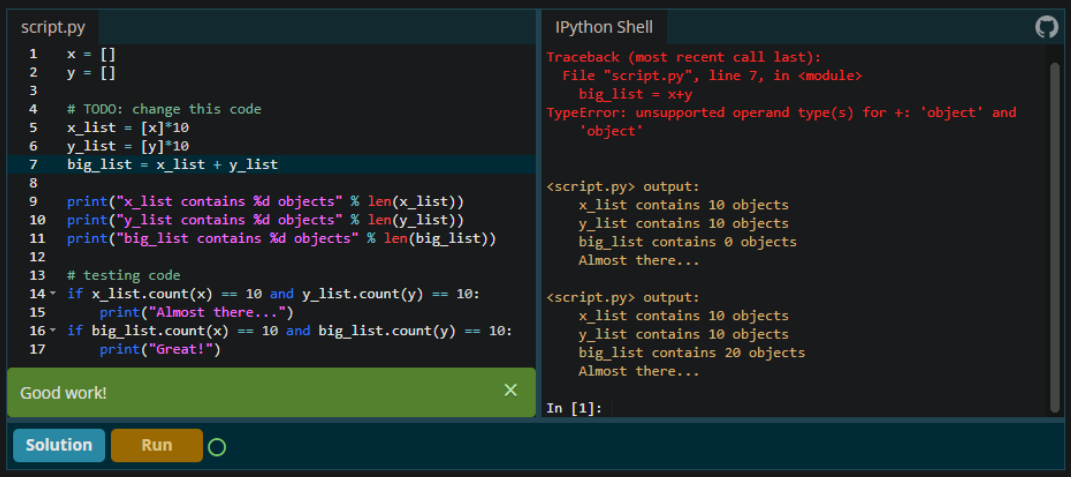
```
<script.py> output:
    [1, 2, 3]
    helloworld
    The second name on the names

In [1]:
```

Solution Run

Powered by DataCamp

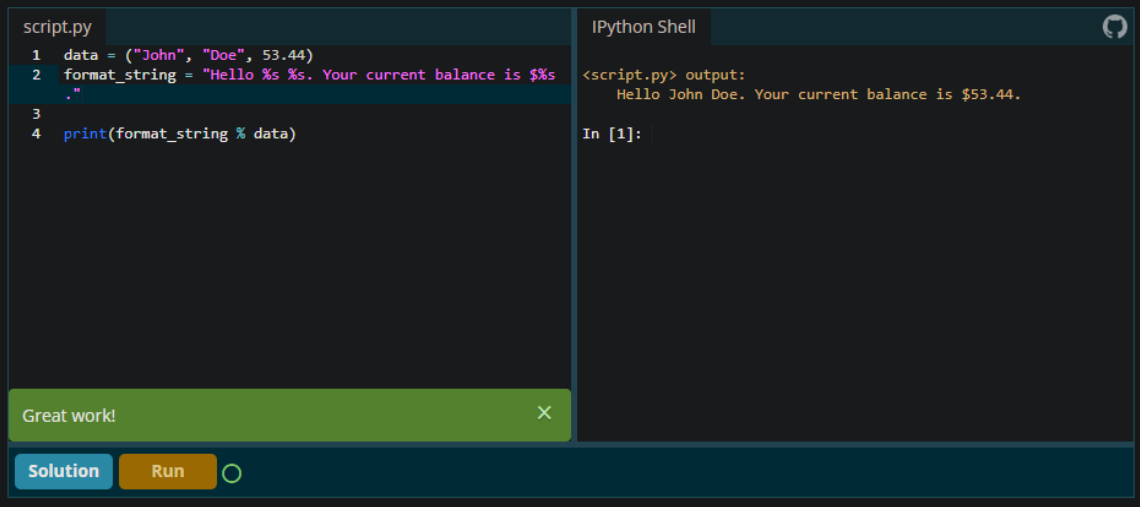
Exercise

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.



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.`



This site is generously supported by DataCamp. DataCamp offers online interactive Python Tutorials for Data Science. Join **over a**

Exercise

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

script.py | solution.py

```
3   print("Length of s = %d" % len(s))
4
5   # First occurrence of "a" should be at index 8
6   print("The first occurrence of the letter a = %d" % s
    .index("a"))
7
8   # Number of a's should be 2
9   print("a occurs %d times" % s.count("a"))
10
11  # Slicing the string into bits
12  print("The first five characters are '%s'" % s[:5]) #
    Start to 5
13  print("The next five characters are '%s'" % s[5:10]) #
    5 to 10
14  print("The thirteenth character is '%s'" % s[12]) #
    Just number 12
15  print("The characters with odd index are '%s'" %s[1
    ::2]) #(0-based indexing)
```

Great work!

IPython Shell

```
    Split the words of the string: ['Strings', 'are',
        'awesome.']

<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]:
```

Run

Powered by DataCamp

Exercise

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

script.py

```
1   # change this code
2   number = 16
3   second_number = 0
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")
```

Great Work!

IPython Shell

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

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

In [1]:
```

Solution | Run

Powered by DataCamp

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.

script.py

```
4       386, 462, 47, 418, 907, 344, 236, 375, 823, 566,
    597, 978, 328, 615, 953, 345,
5       399, 162, 758, 219, 918, 237, 412, 566, 826, 248,
    866, 950, 626, 949, 687, 217,
6       815, 67, 104, 58, 512, 24, 892, 894, 767, 553, 81,
    379, 843, 831, 445, 742, 717,
7       958, 609, 842, 451, 688, 753, 854, 685, 93, 857,
    440, 380, 126, 721, 328, 753, 470,
8       743, 527
9   ]
10
11  # your code goes here
12  for number in numbers:
13      if number == 237:
14          break
15      elif number %2 ==0:
16          print(number)
```

Great work!

IPython Shell

```
980
544
390
984
592
236
942
386
462
418
344
236
566
978
328
162
758
918

In [1]:
```

Solution | Run | Workspace Ready

Powered by DataCamp

3. Run and see all the functions work together!

script.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 "%s is a benefit of functions!" %benefit
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
```

Nice work!

IPython Shell

```
<script.py> output:
    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!

In [1]:
```

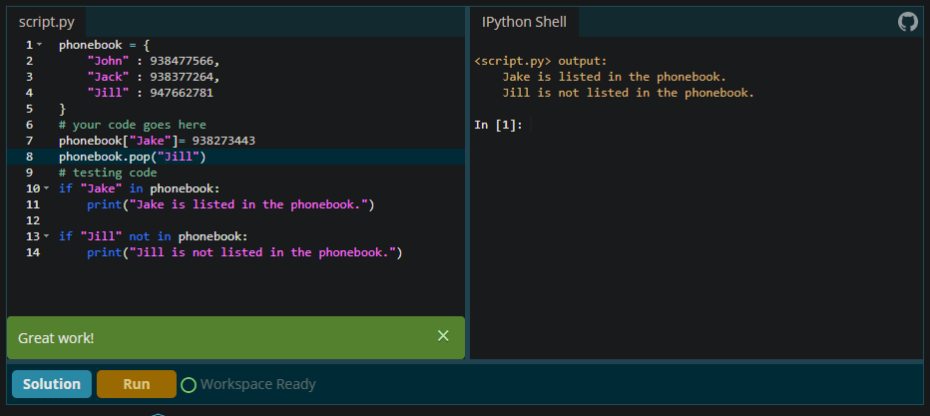
Solution | Run

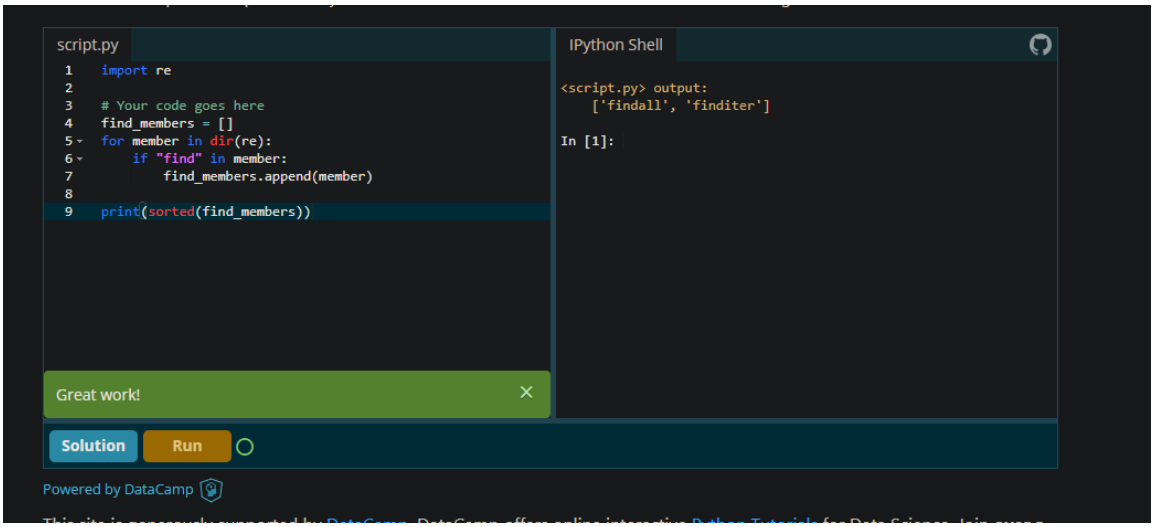
Powered by DataCamp

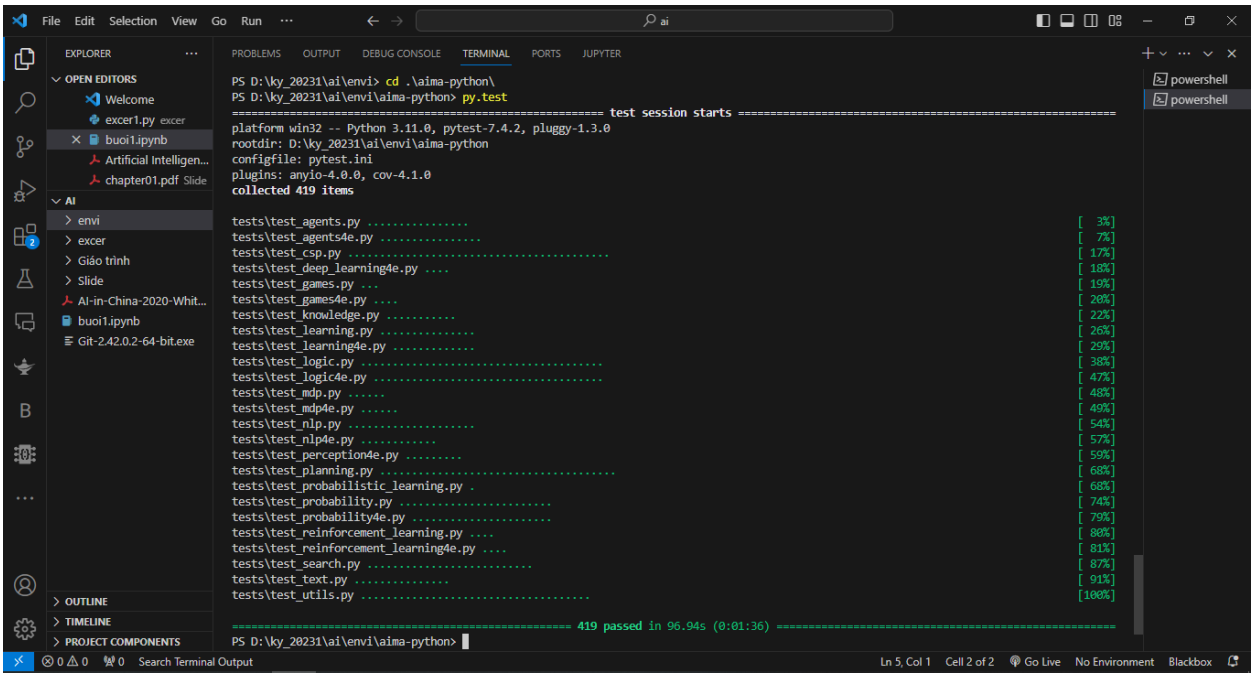
$60,000.00 with a name of Fer, and car2 to be a blue van named Jump worth $10,000.00.

Exercise

Add "Jake" to the phonebook with the phone number 938273443, and remove Jill from the phonebook.






File Edit Selection View Go Run ···
ai
EXPLORER
OPEN EDITORS
Welcome
excer1.py excer
buoi1.ipynb
Artificial Intelligen...
chapter01.pdf Slide
AI
envi
excer
Giáo trình
Slide
AI-in-China-2020-Whit...
buoi1.ipynb
Git-2.42.0.2-64-bit.exe
OUTLINE
TIMELINE
PROJECT COMPONENTS
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER
PS D:\ky_20231\ai\envi> cd .\aima-python\
PS D:\ky_20231\ai\envi\aima-python> py.test
test session starts
platform win32 -- Python 3.11.0, pytest-7.4.2, pluggy-1.3.0
rootdir: D:\ky_20231\ai\envi\aima-python
configfile: pytest.ini
plugins: anyio-4.0.0, cov-4.1.0
collected 419 items
tests\test_agents.py [3%]
tests\test_agents4e.py [7%]
tests\test_csp.py [17%]
tests\test_deep_learning4e.py [18%]
tests\test_games.py [19%]
tests\test_games4e.py [20%]
tests\test_knowledge.py [22%]
tests\test_learning.py [26%]
tests\test_learning4e.py [29%]
tests\test_logic.py [38%]
tests\test_logic4e.py [47%]
tests\test_mdp.py [48%]
tests\test_mdp4e.py [49%]
tests\test_nlp.py [54%]
tests\test_nlp4e.py [57%]
tests\test_perception4e.py [59%]
tests\test_planning.py [68%]
tests\test_probabilistic_learning.py [68%]
tests\test_probability.py [74%]
tests\test_probability4e.py [79%]
tests\test_reinforcement_learning.py [80%]
tests\test_reinforcement_learning4e.py [81%]
tests\test_search.py [87%]
tests\test_text.py [91%]
tests\test_utils.py [100%]
419 passed in 96.94s (0:01:36)
PS D:\ky_20231\ai\envi\aima-python>
powershell
powershell
Search Terminal Output
Ln 5, Col 1 Cell 2 of 2 Go Live No Environment Blackbox