

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 should contain the word "hello". The floating point number should be named `myfloat` and should contain the value 10.0. The integer should be named `myint` and should contain the number 20.

script.py

```
1 # change this code
2 mystring = "hello"
3 myfloat = 10.0
4 myint = 20
5
6 # testing code
7 if mystring == "hello":
8     print("String: %s" % mystring)
9 if isinstance(myfloat, float) and myfloat == 10.0:
10     print("Float: %f" % myfloat)
11 if isinstance(myint, int) and myint == 20:
12     print("Integer: %d" % myint)
```

IPython Shell

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

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

```
In [1]:
```

Great job!



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 method. 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 `names` list. Remember 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)
```

IPython Shell

<script.py> output:

[1, 2, 3]

helloworld

The second name on the names list is Eric

In [1]:

Great Job!



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.

script.py

```
1 x = []
2 y = []
3
4 # TODO: change this code
5 x_list = [x]*10
6 y_list = [y]*10
7 big_list = x_list + y_list
8
9 print("x_list contains %d objects" % len(x_list))
10 print("y_list contains %d objects" % len(y_list))
11 print("big_list contains %d objects" % len(big_list))
12
13 # testing code
14 if x_list.count(x) == 10 and y_list.count(y) == 10:
15     print("Almost there...")
16 if big_list.count(x) == 10 and big_list.count(y) == 10:
17     print("Great!")
```

Good work!

[Solution](#)

[Run](#)



IPython Shell

```
Traceback (most recent call last):
  File "script.py", line 7, in <module>
    big_list = x+y
TypeError: unsupported operand type(s) for +: 'object' and
'object'
```

```
<script.py> output:
x_list contains 10 objects
y_list contains 10 objects
big_list contains 0 objects
Almost there...
```

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

In [1]:

Powered by DataCamp

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.`

script.py

```
1 data = ("John", "Doe", 53.44)
2 format_string = "Hello %s %s. Your current balance is $%.
3
4 print(format_string % data)
```

Great work!

[Solution](#)

[Run](#)



IPython Shell

```
<script.py> output:
Hello John Doe. Your current balance is $53.44.
```

In [1]:

Powered by DataCamp

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!



Run



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]:

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")
18
```

Great Work!



Solution

Run



IPython Shell



<script.py> output:

```
1
2
3
4
5
6
```

<script.py> output:

```
1
2
3
4
5
6
```

In [1]:

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
1 330, 304, 330, 230, 103, 342, 341,
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
14 name_the_benefits_of_functions()
```

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.

script.py

```
4 kind = "car"
5 color = ""
6 value = 100.00
7 def description(self):
8     desc_str = "%s is a %s %s worth $%.2f." %
9     (self.name, self.color, self.kind, self.value)
10    return desc_str
11 # your code goes here
12 car1 = Vehicle()
13 car1.name="Fer"
14 car1.color="red"
15 car1.value=60000.00
16 car2 = Vehicle()
17 car2.name="Jump"
18 car2.color="blue"
19 car2.value=10000.00
20 # test code
21 print(car1.description())
```

Great job!

IPython Shell

```
Traceback (most recent call last):
  File "script.py", line 11, in <module>
    car1 = vehicle()
NameError: name 'vehicle' is not defined

<script.py> output:
  Fer is a red car worth $60000.00.
  Jump is a blue car worth $10000.00.

In [1]:
```

Solution Run

Powered by DataCamp

Exercise

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

script.py

```
1 phonebook = {
2     "John" : 938477566,
3     "Jack" : 938377264,
4     "Jill" : 947662781
5 }
6 # your code goes here
7 phonebook["Jake"] = 938273443
8 phonebook.pop("Jill")
9 # testing code
10 if "Jake" in phonebook:
11     print("Jake is listed in the phonebook.")
12
13 if "Jill" not in phonebook:
14     print("Jill is not listed in the phonebook.")
```

Great work!

IPython Shell

```
<script.py> output:
  Jake is listed in the phonebook.
  Jill is not listed in the phonebook.

In [1]:
```

Solution Run Workspace Ready

Powered by DataCamp

script.py

```
1 import re
2
3 # Your code goes here
4 find_members = []
5 for member in dir(re):
6     if "find" in member:
7         find_members.append(member)
8
9 print(sorted(find_members))
```

Great work!

IPython Shell

```
<script.py> output:
  ['findall', 'finditer']

In [1]:
```

Solution Run

Powered by DataCamp

This interactive environment is supported by DataCamp. DataCamp offers online data science training for Data Analysts, Data Engineers, Data Scientists, and Machine Learning Engineers.

```
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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER
PS D:\ky_20231\ai\envi> cd .\ai-python\
PS D:\ky_20231\ai\envi\ai-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\ai-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 ..... [ 60%]
tests\test_probabilistic_learning.py . [ 60%]
tests\test_probability4e.py ..... [ 74%]
tests\test_reinforcement_learning.py .... [ 79%]
tests\test_reinforcement_learning4e.py .... [ 80%]
tests\test_search.py ..... [ 81%]
tests\test_text.py ..... [ 87%]
tests\test_utils.py ..... [ 91%]
tests\test_utils4e.py ..... [ 100%]

===== 419 passed in 96.94s (0:01:36) =====
PS D:\ky_20231\ai\envi\ai-python>
```