

Conditions

Exercise

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

script.py	IPython Shell
<pre>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")</pre>	<pre><script.py> output: 1 2 3 4 5 6 In [1]: </pre>
Great Work!	
Solution	Submit ●

Loops

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	IPython Shell
<pre>1 numbers = [2 951, 402, 984, 651, 360, 69, 408, 319, 601, 3 485, 980, 507, 725, 547, 544, 4 615, 83, 165, 141, 501, 263, 617, 865, 575, 5 219, 390, 984, 592, 236, 105, 942, 941, 6 386, 462, 47, 418, 907, 344, 236, 375, 823, 7 566, 597, 978, 328, 615, 953, 345, 8 399, 162, 758, 219, 918, 237, 412, 566, 826 9 , 248, 866, 950, 626, 949, 687, 217, 10 815, 67, 104, 58, 512, 24, 892, 894, 767, 11 553, 81, 379, 843, 831, 445, 742, 717, 12 958, 609, 842, 451, 688, 753, 854, 685, 93, 13 857, 440, 380, 126, 721, 328, 753, 470, 14 743, 527 15]</pre>	<pre>In [1]: </pre>
Great work!	
Solution	Submit ●

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

```
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
10- def name_the_benefits_of_functions():
11     list_of_benefits = list_benefits()
12     for benefit in list_of_benefits:
13         build_sentence(benefit)
```

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!

<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!
```

Nice work!

Solution

Submit

Classes and Objects

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.

script.py

```
1 # define the Vehicle class
2- class Vehicle:
3     name = ""
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
12 # your code goes here
13 car1 = Vehicle()
14 car1.name = "Fer"
15 car1.color = "red"
16 car1.kind = "convertible"
17 car1.value = 60000.00
18
19 car2 = Vehicle()
20 car2.name = "Jump"
21 car2.color = "blue"
22 car2.kind = "van"
23 car2.value = 10000.00
```

IPython Shell

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

In [1]:
```

Great job!

Solution

Submit

Dictionaries

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
7 # your code goes here
8 phonebook["Jake"] = 938273443
9 del phonebook["Jill"]
10
11 # testing code
12- if "Jake" in phonebook:
13     print("Jake is listed in the phonebook.")
14
15- if "Jill" not in phonebook:
16     print("Jill is not listed in the phonebook.")
```

IPython Shell

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

In [1]: |
```

Nice work!

Solution

Submit

Modules and Packages

Exercise

In this exercise, print an alphabetically sorted list of all the functions in the `re` module containing the word `find`.

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))
```

IPython Shell

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

In [1]: |
```

Great work!

Solution

Submit