**Iterators vs Iterables**

Let's do a quick recall of what you've learned about **iterables** and **iterators**. Recall from the video that an *iterable* is an object that can return an *iterator*, while an *iterator* is an object that keeps state and produces the next value when you call next() on it. In this exercise, you will identify which object is an *iterable* and which is an *iterator*.

The environment has been pre-loaded with the variables flash1 and flash2. Try printing out their values with print() and next() to figure out which is an *iterable* and which is an *iterator*.

**Instructions**

**50 XP**

**Possible Answers**

* 

Both flash1 and flash2 are iterators.

* 

Both flash1 and flash2 are iterables.

* 

flash1 is an iterable and flash2 is an iterator.

Submit Answer

[**Take Hint (-15 XP)**](javascript:void(0))

In [1]: print(flash1)

['jay garrick', 'barry allen', 'wally west', 'bart allen']

In [2]: print(flash2)

<list\_iterator object at 0x7f872ec46ef0>

In [3]: next()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

next()

TypeError: next expected at least 1 arguments, got 0

In [4]: next(flash1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

next(flash1)

TypeError: 'list' object is not an iterator

In [5]: next(flash2)

Out[5]: 'jay garrick'

In [6]: next(flash2)

Out[6]: 'barry allen'

In [7]:

**Iterating over iterables (1)**

Great, you're familiar with what iterables and iterators are! In this exercise, you will reinforce your knowledge about these by iterating over and printing from iterables and iterators.

You are provided with a list of strings flash. You will practice iterating over the list by using a for loop. You will also create an iterator for the list and access the values from the iterator.

**Instructions**

**100 XP**

* Create a for loop to loop over flash and print the values in the list. Use person as the loop variable.
* Create an *iterator* for the list flash and assign the result to superhero.
* Print each of the items from superhero using next() 4 times.

[**Take Hint (-30 XP)**](javascript:void(0))

# Create a list of strings: flash

flash = ['jay garrick', 'barry allen', 'wally west', 'bart allen']

# Print each list item in flash using a for loop

for person in flash: print(person)

# Create an iterator for flash: superhero

superhero = iter(flash)

# Print each item from the iterator

print(next(superhero))

print(next(superhero))

print(next(superhero))

print(next(superhero))

<script.py> output:

jay garrick

barry allen

wally west

bart allen

jay garrick

barry allen

wally west

bart allen

In [1]:

**Iterating over iterables (2)**

One of the things you learned about in this chapter is that not all iterables are *actual* lists. A couple of examples that we looked at are *strings* and the use of the range() function. In this exercise, we will focus on the range() function.

You can use range() in a for loop *as if* it's a list to be iterated over:

for i in range(5):

print(i)

Recall that range() doesn't actually create the list; instead, it creates a range object with an iterator that produces the values until it reaches the limit (in the example, until the value 4). If range() created the actual list, calling it with a value of 1010010100 may not work, especially since a number as big as that may go over a regular computer's memory. The value 1010010100 is actually what's called a **Googol** which is a 1 followed by a hundred 0s. That's a huge number!

Your task for this exercise is to show that calling range() with 1010010100 won't actually pre-create the list.

**Instructions**

**100 XP**

* Create an **iterator** object small\_value over range(3) using the function iter().
* Using a for loop, iterate over range(3), printing the value for every iteration. Use num as the loop variable.
* Create an **iterator** object googol over range(10 \*\* 100).

[**Take Hint (-30 XP)**](javascript:void(0))

# Create an iterator for range(3): small\_value

small\_value = iter(range(3))

# Print the values in small\_value

print(next(small\_value))

print(next(small\_value))

print(next(small\_value))

# Loop over range(3) and print the values

for i in range(3):

print(i)

# Create an iterator for range(10 \*\* 100): googol

googol = iter(range(10 \*\* 100))

# Print the first 5 values from googol

print(next(googol))

print(next(googol))

print(next(googol))

print(next(googol))

print(next(googol))

<script.py> output:

0

1

2

0

1

2

0

1

2

3

4

**Iterators as function arguments**

You've been using the iter() function to get an iterator object, as well as the next() function to retrieve the values one by one from the iterator object.

There are also functions that take iterators and iterables as arguments. For example, the list() and sum() functions return a list and the sum of elements, respectively.

In this exercise, you will use these functions by passing an iterable from range() and then printing the results of the function calls.

**Instructions**

**100 XP**

**Instructions**

**100 XP**

* Create a range object that would produce the values from 10 to 20 using range(). Assign the result to values.
* Use the list() function to create a list of values from the range object values. Assign the result to values\_list.
* Use the sum() function to get the sum of the values from 10 to 20 from the range object values. Assign the result to values\_sum.

[**Take Hint (-30 XP)**](javascript:void(0))

**Incorrect Submission**

Check the argument in your call of sum(). Did you use the correct *iterable*?

# Create a range object: values

values = range(10,21)

# Print the range object

print(values)

# Create a list of integers: values\_list

values\_list = list(values)

# Print values\_list

print(values\_list)

# Get the sum of values: values\_sum

values\_sum = sum(values)

# Print values\_sum

print(values\_sum)

<script.py> output:

range(10, 21)

[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

165