lab3_data_loops

January 24, 2024

1 Lab 3: Python Basic Data, Lists Loops

1.1 Greeting

In Python, there are several different "data types," which are different categories of variables you can do different things with. For example, there are:

int (Simple integers like 1, 2, 5216…) str (Strings of text, like "Hi", "My name is Spencer", "?!?!?!?!?!?!?"…) bool (Booleans, which are variables that can only take the value True or False) float (Numbers with decimal points, like 1.1, 2.56, 0.000002…)

…and several others that we'll talk about later!

First, Make a new variable called `greeting_part_1` and assign it the string: `"Welcome, "`

Note: There is an extra space after the word

[1]:
```
greeting_part_1 = "Welcome, "
```

Make a new variable called `greeting_part_2` and assign it the string: `"! It is nice to meet you!"`

Note: this string starts with an exlamation mark.

[2]:
```
greeting_part_2 = "! It is nice to meet you!"
```

Make a variable called name_1 and assign it with a string of someone's name.

[3]:
```
name_1 = "Edward"
```

Now, combine them all together into a new variable called `full_greeting`. Combine the parts together by taking `greeting_part_1` then adding `name` then adding `greeting_part_2`.

Then display the variable `full_greeting`.

[4]:
```
full_greeting = greeting_part_1 + name_1 + greeting_part_2
display(full_greeting)
```

```
'Welcome, Edward! It is nice to meet you!'
```

Now make a variable called `name_2` with another name in it and make a variable called `called_full_greeting_2` the same way you did `full_greeting`, but with `name_2` instead.

Then display the variable `full_greeting_2`

```
[5]: name_2 ="Aurora"
     full_greeting_2 = greeting_part_1 + name_2 + greeting_part_2
     display(full_greeting_2)
```

```
'Welcome, Aurora! It is nice to meet you!'
```

1.2 Engagement report

Make up numbers for tweet metrics and save them into variables: `number_likes`, `number_retweets`, `number_quote_tweets`

```
[6]: number_likes = 17
     number_retweets = 5
     number_quote_tweets = 9
```

Write three different `display()` function calls, one to report on each of those variables.

Inside the parentheses first put a string like, `"The number of likes is: "` and then add the relevant variable to it, but remember, since the variable has a number in it, you have to put a `str()` function call around that variable name

```
[7]: display("The number of likes is: " + str(number_likes))
     display("The number of retweets is: " + str(number_retweets))
     display("The number of tweers quote is: " + str(number_quote_tweets))
```

```
'The number of likes is: 17'

'The number of retweets is: 5'

'The number of tweers quote is: 9'
```

Make a new variable called `total_engagement` and save the total number of all the likes, retweets and quote tweets.

Then display that information the way you did the other variables.

```
[8]: total_engagement = number_likes + number_retweets + number_quote_tweets
     display("The number of total engagement is: " + str(total_engagement))
```

```
'The number of total engagement is: 31'
```

1.3 Is your tweet too long

Make a variable called `tweet_1` with a string of your choosing that is fairly short

```
[9]: tweet_1 = "Cat"
```

Make a variable called `tweet_1_length`, and save the length of the string `tweet_1` into it (using the `len()` function)

```
[10]: tweet_1_length = len(tweet_1)
```

Check if the `tweet_1_length` is less than or equal to 280 characters (the max length of a tweet) using the less than or equal operator: `<=`, and save the result into a variable called `is_tweet_1_short_enough` (note that this is a boolean)

```
[11]: is_tweet_1_short_enough = tweet_1_length <= 280
```

Use the `display()` function to display a string (`"Is tweet 1 short enough? "`), adding the variable `is_tweet_1_short_enough` to that string.

Remember, since the variable has a boolean and not a string in it, you have to put a str() function call around that variable name

```
[12]: display("Is tweet 1 short enough? " + str(is_tweet_1_short_enough))
```

```
'Is tweet 1 short enough? True'
```

```
[15]: tweet_2 = "hahahhahahhaha hahahahhahahha hahahahhahahahahahah ahhaahahahahahha␣
       ↪ha ahha ah hahahaha ha haha h ahah ha ha ha hhahahahahah aha ah ah aha ha␣
       ↪ahahhahahahaahhahahah aha ha haahhahahaahahahahaahahahaha.␣
       ↪ahahahahahahahahahahahahahha ah ahhaahahaha hahahahhahahahahahaha ah ahahah␣
       ↪ah ahh"
```

Repeat the rest of the steps from before, but with tweet_2 - find tweet_2_length - make a variable is_tweet_2_short_enough - display the result

```
[16]: tweet_2_length = len(tweet_2)
      is_tweet_2_short_enough = tweet_2_length <= 280
      display("Is tweet 2 short enough? " + str(is_tweet_2_short_enough))
```

```
'Is tweet 2 short enough? False'
```

1.4 Lists

In python, sometimes we want to store several variables inside another variable. This is called a list. For example, we can have a list of pets:

mylist = [“dog”, “cat”, “fish”]

If we want to take something out of our list, we can use its index. In Python, items in a list start at index 0, then 1, and so on. So if we want to get the first item in a list, we could call:

mylist[0]

Which give us:

“dog”

If we wanted to change the second item in a list, we could do mylist[1] = “zebra”

Which would turn our list into:

[“dog”, “zebra”, “fish”]

We can also add things to lists using the “append” function. For example, if we run mylist.append(“hamster”), we would now have:

[“dog”, “zebra”, “fish”, “hamster”]

Now, make a list of names (at least three), and save it in a variable called `names`

[17]:
```
names = ["Edward","Aurora","Jack"]
```

Next, use the print() function to print the third name in your list (remember, list index starts at 0…)

[18]:
```
print(names[2])
```

```
Jack
```

1.5 Loops

In python, we can use loops as a way of running the same chunk of code over and over again, usually on a different variable or in some different context.

There are two types of loops in python: “For loops”, and “while loops.” In this lab, we’ll focus on “for loops”, which allow us to run a block of code on every variable in a list. For example, if we have a list of posts called “post_list”, we could run the following code:

for post in post_list: display(post)

…the program will go through every element in our list of posts, and display. To break this down, the keyword “for” means you’re starting a for loop, “post” is the name we’re giving to the elements in the list we’re iterating through, “in” is a required keyword before we specify our list, and “post_list” is the list of posts. We add a “:” to the end to show we’re starting a loop.

A more generic version might look like:

for x in some_list: do_something(x) do_another_thing(x) …

Now, you should loop through every name in your “names” list from before, and for each name display “[name] is awesome!”

[19]:
```
for user in names: display(user + " is awesome!")
```

```
'Edward is awesome!'
```

```
'Aurora is awesome!'
```

```
'Jack is awesome!'
```

Now, do the same thing as before, but for each name, first make a string that has “[name] is awesome!” and save it in a variable, then use the .upper() function on the string to make it all uppercase and save it into a variable, then display the final string.

[20]:
```
for user in names:
    original = user + " is awesome!"
    after = original.upper()
    display(after)
```

```
'EDWARD IS AWESOME!'

'AURORA IS AWESOME!'

'JACK IS AWESOME!'
```

```
[ ]:
```