Python 8

Split

split() makes a list into a string

s.split(separator, maxsplit)

For example:

```
text = “I like apples”

apples = text.split()

apples = [“I”, “like”, “apples”]
```

Calculating averages

How can we split scores so we can separate each student’s code and calculate the average?

scores = “Leia 90 81 96 92

Luke 75 80 94 87

Han 79 84 88 83

Obi-wan 93 97 91 95”

Calculating averages

How can we split scores so we can separate each student’s code and calculate the average?

We can use (\n) as the separator. \n stands for new line.

So,

```
students = scores.split(“\n”)

students = [“Leia 90 81 96 92”, “Luke 75 80 94 87”...]
```

```
How can we calculate one student’s average?
```

Calculate

Let’s look at one line:

```
text = “Leia 90 81 96 92”
```

You can perform operations on this line

```
grades = text.split()

print(grades)  #[“Leia”, “90”, “81”, “96”, “92”]
```

The next step is to write a loop that calculates the average. You can start from index one and work your way to the end of the loop. Remember these are strings!!!

1 + “123”

1 + int(“123”)

Don’t do this, use a loop!

```
grades =[“Leia”, “90”, “81”, “96”, “92”]

average = int(grades[1]) + int(grades[2])
```

Calculate the average and print with the name for each name.

```
print(grades[0] + “ “ + average +”\n”)
```

Your friend really likes talking about owls. Write a function owlCount(text) that takes a block of text and counts how many words they say have word “owl” in them. Any word with “owl” in it should count, so “owls,” “owlette,” and “howl” should all count.

Here’s what an example run of your program might look like:

```
text = "I really like owls. Did you know that an owl's eyes are more than twice as big as the eyes of other birds of comparable weight? And that when an owl partially closes its eyes during the day, it is just blocking out light? Sometimes I wish I could be an owl."

owlCount(text)
```

Sequences

In python a SEQUENCE can be a list, string, or a few things we haven’t learned yet: dictionary, set, tuple.

Negative indices count backwards

Negative indexes refer to the positions of elements within an array-like object such as a list, tuple, or string, counting from the end of the data structure rather than the beginning.

Examples:

```
lst = [‘a’, ‘b’, ‘c’, ‘d’]

>>> print( lst[-1] )

d

>>> print( lst[-3] )

b
```