Module 3: Conditions and Loops

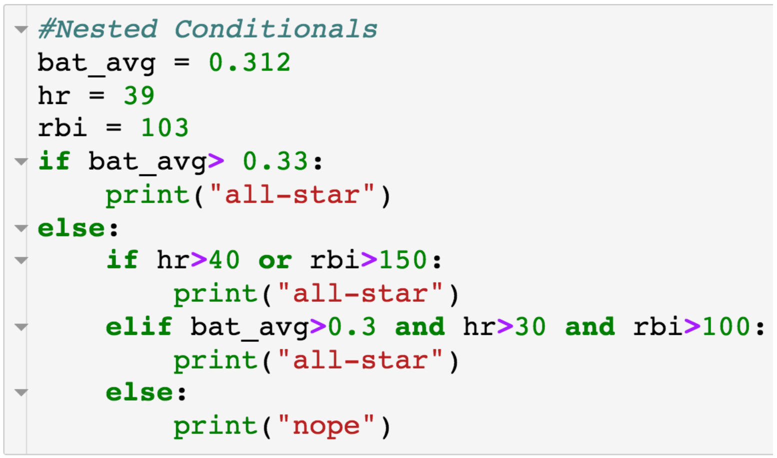
1

Knowledge Points

- If statement:
  - Conditional test
  - If, else, elif
  - Nested conditions
  - Multiple conditions
- For loop:
  - For loop with lists and strings
  - For loop with ranges
- Dictionary and Tuple
- Object and Methods:
  - String methods
  - List methods
  - Dictionary methods
  - Google

2

If statment

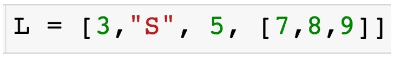


```
#Nested Conditionals
bat_avg = 0.312
hr = 39
rbi = 103
if bat_avg> 0.33:
    print("all-star")
else:
    if hr>40 or rbi>150:
        print("all-star")
    elif bat_avg>0.3 and hr>30 and rbi>100:
        print("all-star")
    else:
        print("nope")
```

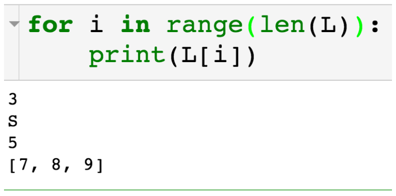
3

For Loops

How do I iterate over the indices of a list using a for loop?

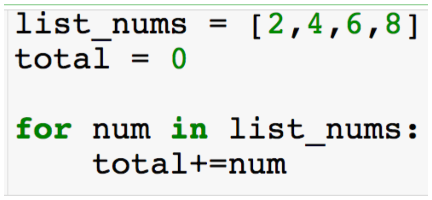


```
L = [3,"S", 5, [7,8,9]]
```

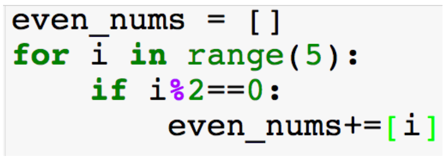


```
for i in range(len(L)):
    print(L[i])
```

```
3
S
5
[7, 8, 9]
```



```
list_nums = [2,4,6,8]
total = 0

for num in list_nums:
    total+=num
```

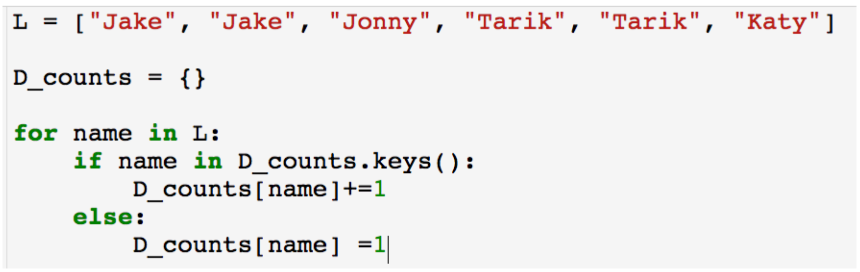


```
even_nums = []
for i in range(5):
    if i%2==0:
        even_nums+=[i]
```

4

Dictionaries

Let’s say I want to create a dictionary which stores the number of times each word appears in the list:

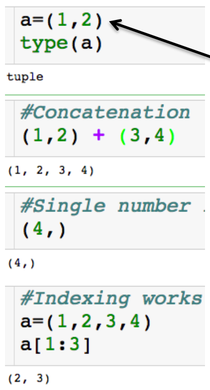


```
L = ["Jake", "Jake", "Jonny", "Tarik", "Tarik", "Katy"]

D_counts = {}

for name in L:
    if name in D_counts.keys():
        D_counts[name]+=1
    else:
        D_counts[name] =1
```

5

Tuples

- Tuples are essentially immutable lists.
  - They can be slice and index and used in for loops, but you can’t sort them.
- Since they are immutable, they can be keys in a dictionary, as we already saw.



```
a=(1,2)
type(a)
```

```
tuple
```

```
#Concatenation
(1,2) + (3,4)
```

```
(1, 2, 3, 4)
```

```
#Single number
(4,)
```

```
(4,)
```

```
#Indexing works
a=(1,2,3,4)
a[1:3]
```

```
(2, 3)
```

Notice parentheses instead of bracket!

6

Python Objects

- Python objects are **dynamically typed** – when we create a variable we don't have to say what type of object it will store.
- Python objects are wither **mutable** (can be changed) or **immutable** (cannot be changed)
- Python objects are **strongly typed** – there are built in type specific methods that help us manipulate objects.

7

String Methods

Replace method : global search and replace.

```
name = "Jaqe"
correct_name = name.replace("q", "k")

name
correct_name
```

```
'Jaqe'
'Jake'
```

name of string

method

Find method : finds the first location of the given substring (or a -1 if it is not found).

```
sentence = "Hello World."

sentence.find('e')
sentence.find(' ')
```

```
1
5
```

Split method : splits string into list, delimited by input.

```
line = 'I went to the store'
words = line.split(' ')

words
```

```
['I', 'went', 'to', 'the', 'store']
```

```
line = '      I went to the store.'
new_line_strip  = line.strip(" .")
final_line = new_line_strip.split(" ")
new_line_strip
final_line
```

```
'I went to the store'
['I', 'went', 'to', 'the', 'store']
```

8

List Methods

Append method : Add element to end of the list.

```
L = [1,2,3]
L.append(4)
L
```

```
[1, 2, 3, 4]
```

Sort method : Sorts the elements in the list

```
L = [4,5,1]
L.sort()
L
```

```
[1, 4, 5]
```

```
L = [4,5,1]
sorted_L = sorted(L)
sorted_L
L
```

```
[1, 4, 5]
[4, 5, 1]
```

Index method : returns the index of first occurrence of the inputted element.

```
L = [4,5,1]
index_five = L.index(5)
index_five
```

```
1
```

```
L = [4,5,1,5,12,3,4,1,5,7]

L.sort().index(5)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-28-11ce6a038eb9> in <module>()
      1 L = [4,5,1,5,12,3,4,1,5,7]
      2
----> 3 L.sort().index(5)

AttributeError: 'NoneType' object has no attribute 'index'
```

9

Dictionary Methods

keys method : returns the keys as an interable.

```
D = {"Jake":1, "Joe":2}
D.keys()
list(D.keys())
```

```
dict_keys(['Jake', 'Joe'])
['Jake', 'Joe']
```

values method : returns the values as an interable.

```
D = {"Jake":1, "Joe":2}
D.values()
list(D.values())
```

```
dict_values([1, 2])
[1, 2]
```

get method : another way to access a value through a key

```
D = {"Jake":1, "Joe":2}
#Get value associated with key "Jake
D.get("Jake")
```

```
1
```

```
D["Steve"]
```

```
------------------------------
KeyError
<ipython-input-9-f4219aed6c
----> 1 D["Steve"]

KeyError: 'Steve'
```

```
print(D.get("Steve"))
```

```
None
```

10