Week 3

Intro Python

https://codewith.mu/

http://cadtx.pw/intropy3

Objectives

Chapter 4

- Review of lists
- For loops
- List slices
- Tuples

Review

- Lists are ordered collections of items
- Declare them with []
- my_list = [1,”hi”,3.0]
- Access values using brackets, indices start at 0
- print(my_list[0]) -> 1
- Reassign using the same syntax:
  - My_list[1] = “hello”
  - List is now [1,”hello”,3.0]

- Add element to the end using list.append()
- Remove using del list[index] or list.pop(index)
- Remove by value using list.remove(value)
- Can sort in place using list.sort or create a copy using sorted(list)
- len(list) returns its length
- list.reverse() will reverse the list

Looping Through an Entire List

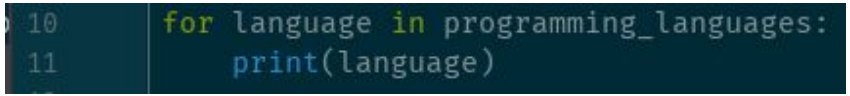
- Oftentimes you'll want to perform the same task with each item, we can do this with **for** loops
- We can do this manually, but it's tedious and we have to know the length of the list

```
programming_languages = ["Python", "Java", "C++", "JavaScript"]
print(programming_languages[0])
print(programming_languages[1])
print(programming_languages[2])
print(programming_languages[3])

# With a list
for language in programming_languages:
    print(language)
```

```
Python
Java
C++
JavaScript

Process finished with exit code 0
```

A Closer Look at Looping

- When the Python interpreter reads the first line, it retrieves the value of the first item in the list “programming_languages” and assigns that value to “language”
- It then executes the code in the body of the statement
- This will repeat until it has gone through every value in the list



https://goo.gl/ZFwqT5

Indentation

- Python uses indentation to determine whether lines of code are connected to the lines above it
- If you forget to indent after the “for” statement Python will throw an error
- If you wanted to print those two lines for each element of the list, but only indent the first you won’t get the output you expect - this is a *logical* error.

```
# Forgetting to indent
for language in programming_languages:
print(language)

# Not indenting additional lines
for language in programming_languages:
    print("I want to learn", language)
print(language, " seems like it will be useful.")
```

```
C:\Users\sean9\Anaconda3\python.exe "C:/Users/
  File "C:/Users/sean9/PycharmProjects/fa18-ir
    print(language)
          ^
IndentationError: expected an indented block
```

```
I want to learn Python
I want to learn Java
I want to learn C++
I want to learn JavaScript
JavaScript  seems like it will be useful.
```

Making numerical lists

- Python’s range() function allows you to generate numbers
- Uses the syntax range(start, stop, step). This will generate a list starting at the start value, and going up to but not including the stop value. Step is optional.
- If you want a list of numbers, convert it to one using list()

```
for value in range(1, 5):
    print(value)
```

```
1
2
3
4
```

```
numbers = list(range(1,6))
print(numbers)

even_numbers = list(range(2,11,2))
print(even_numbers)
```

```
[1, 2, 3, 4, 5]
[2, 4, 6, 8, 10]
```

```python
squares = []
for value in range(1,11):
    square = value ** 2
    squares.append(square)

print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Simple statistics

- Python offers a few built in functions specific to lists of numbers
- min()
- max()
- sum()

```
In[2]: numbers = list(range(1,11))
In[3]: min(numbers)
Out[3]:
1
In[4]: max(numbers)
Out[4]:
10
In[5]: sum(numbers)
Out[5]:
55
In[6]: print(numbers)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Exercises

TRY IT YOURSELF

4-3. Counting to Twenty: Use a `for` loop to print the numbers from 1 to 20, inclusive.

4-4. One Million: Make a list of the numbers from one to one million, and then use a `for` loop to print the numbers. (If the output is taking too long, stop it by pressing CTRL-C or by closing the output window.)

4-5. Summing a Million: Make a list of the numbers from one to one million, and then use `min()` and `max()` to make sure your list actually starts at one and ends at one million. Also, use the `sum()` function to see how quickly Python can add a million numbers.

4-6. Odd Numbers: Use the third argument of the `range()` function to make a list of the odd numbers from 1 to 20. Use a `for` loop to print each number.

4-7. Threes: Make a list of the multiples of 3 from 3 to 30. Use a `for` loop to print the numbers in your list.

4-8. Cubes: A number raised to the third power is called a *cube*. For example, the cube of 2 is written as `2**3` in Python. Make a list of the first 10 cubes (that is, the cube of each integer from 1 through 10), and use a `for` loop to print out the value of each cube.

Working with parts of a list

- Last week we learned how to access single elements in a list and so far we've learned how to work with all of the elements
- You can also work with parts of a list, called *slices*
- You can slice lists using the following syntax:
  - list[start:stop:step]

```
In[2]: people = ['charles', 'martina', 'michael', 'florence', 'eli']
In[3]: # first three elements of the list
In[4]: print(people[0:3])
['charles', 'martina', 'michael']
In[5]: # can slice any subset of a list
In[6]: print(people[1:4])
['martina', 'michael', 'florence']
In[7]: # if you omit the first index it defaults to 0
In[8]: print(people[:3])
['charles', 'martina', 'michael']
In[9]: # if you omit the last index it defaults to the end
In[10]: print(people[2:])
['michael', 'florence', 'eli']
In[11]: # negative indexing works too
In[12]: print(people[-3:]) # third to last to the end
['michael', 'florence', 'eli']
```

More on slicing

- You can specify a “step” much like range()
- You can also slice backwards by providing a negative value for step

```
In[2]: people = ['charles', 'martina', 'michael', 'florence', 'eli']
In[13]: # There is also a step param
In[14]: print(people[::2])
['charles', 'michael', 'eli']
In[15]: # We can also slice and step backwards
In[16]: print(people[::-1])
['eli', 'florence', 'michael', 'martina', 'charles']
In[17]: print(people[::-2])
['eli', 'michael', 'charles']
```

Copying a list

- You can specify a “step” much like range()
- You can also slice backwards by providing a negative value for step

More exercises

TRY IT YOURSELF

4-10. Slices: Using one of the programs you wrote in this chapter, add several lines to the end of the program that do the following:

- Print the message, *The first three items in the list are:*. Then use a slice to print the first three items from that program's list.
- Print the message, *Three items from the middle of the list are:*. Use a slice to print three items from the middle of the list.
- Print the message, *The last three items in the list are:*. Use a slice to print the last three items in the list.