

Maps

We will be using python package GEOPY → geopy makes it easy for us to locate the coordinates of landmarks, cities, addresses etc.

```
geolocator.geocode("THE NAME OF THE PLACE YOU WANT TO STUDY")
```

- Note: this will return an object!
- An object, is something that has attributes

Maps

We will be using python package GEOPY → geopy makes it easy for us to locate the coordinates of landmarks, cities, addresses etc.

```
geolocator.geocode("THE NAME OF THE PLACE YOU WANT TO STUDY")
```

- Note: this will return an object!
- An object, is something that has attributes
- Ex. a cat (attributes include : hair quality, color of eyes....

Maps

We will be using python package GEOPY → geopy makes it easy for us to locate the coordinates of landmarks, cities, addresses etc.

```
geolocator.geocode("THE NAME OF THE PLACE YOU WANT TO STUDY")
```

- Note: this will return an object!
- An object, is something that has attributes
- Ex. a cat (attributes include : hair quality, color of eyes....

This location-object we are talking about has the following attributes:

location.latitude, location.longitude, location.altitude, location.address

Use "dot" notation ;)

For loops

- A way to **iterate** through your object to look at (or act upon) each item in your object

```
>>> list = [1, 2, 3, 4, 5]
```

```
>>> new_list = []
```

```
>>> for i in list:
```

```
    new_list.append(i + 5)
```

For loops

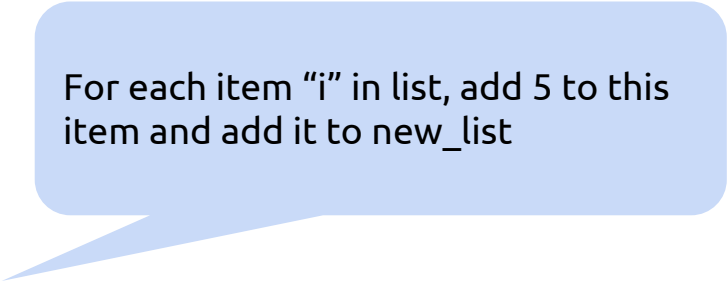
- A way to **iterate** through your object to look at (or act upon) each item in your object

```
>>> list = [1, 2, 3, 4, 5]
```

```
>>> new_list = []
```

```
>>> for i in list:
```

```
    new_list.append(i + 5)
```



For each item "i" in list, add 5 to this item and add it to new_list

For loops

- A way to **iterate** through your object to look at (or act upon) each item in your object

```
>>> list = [1, 2, 3, 4, 5]
```

```
>>> new_list = []
```

```
>>> for i in list:
```

```
    new_list.append(i + 5)
```

```
>>> new_list
```

```
[6, 7, 8, 9, 10]
```

Append

- `list.append(...)`
- `table.append(...)`

```
>>> dog_names = Table().with_columns('name', [Happy, Mochi, Doc], 'age', [1, 2, 3])
```

```
>>> dog_names
```

name	age
Happy	1
Mochi	2
Doc	3

Append

```
>>> dog_names = Table().with_columns('name', [Happy, Mochi, Doc], 'age', [1, 2, 3])
```

```
>>> dog_names
```

name	age
Happy	1
Mochi	2
Doc	3

```
>>> dog_names.append(['Lucky', 4])
```


Append

```
>>> dog_names.append(['Lucky', 4])
```

```
>>> dog_names
```

name	age
Happy	1
Mochi	2
Doc	3
Lucky	4

Concatenating strings

- We know strings can be added together!

Ex. `y = "Science"`

`X = "Rules"`

```
Awesome_phrase = X + Y
```

```
print(Awesome_phrase)
```

```
>>>>>
```

Concatenating strings

- We know strings can be added together!

Ex. `y = "Science"`

`X = "Rules"`

```
Awesome_phrase = X + Y
```

```
print(Awesome_phrase)
```

```
>>>>>
```

```
ScienceRules
```

```
....is this what we wanted?
```

2 options *either put a SPACE before Rules → " Rules"

* or add in the space yourself → " " (this is unnecessary work tho fam)

```
My_string = "Science" + " Rules! " + "Bill" + "Bill" + "Bill" + "Bill"
```

```
print(My_string)
```

Concatenating strings

- We know strings can be added together!

Ex. `y = "Science"`

`x = "Rules"`

```
Awesome_phrase = x + y
```

```
print(Awesome_phrase)
```

```
>>>>
```

```
ScienceRules
```

```
....is this what we wanted?
```

2 options *either put a SPACE before Rules → " Rules"

* or add in the space yourself → " " (this is unnecessary work tho fam)

```
My_string = "Science" + " Rules! " + "Bill" + "Bill" + "Bill" + " " + "Bill"
```

```
print(My_string)
```

```
Science Rules! BillBillBill Bill
```