Non-Primitives Data types

Variables

Primitives (singular)	Non-Primitives(Collections)
Integer	List[]
Float	Dictionary{}
String	Tuple()
Boolean	Set{}
By Python	By Programmer

List

[ ]
Orderd
Mutable
Allows duplicates

- A list is …   a list- of items.
- Declared by [ ].
- Items in the list are separated with a comma.
- Can hold a mix of all kinds of data:

```
l_1 = [1, 2, 3]
l_2 = [1, "hi", 3.4, 'bye']
l_empty = []
```

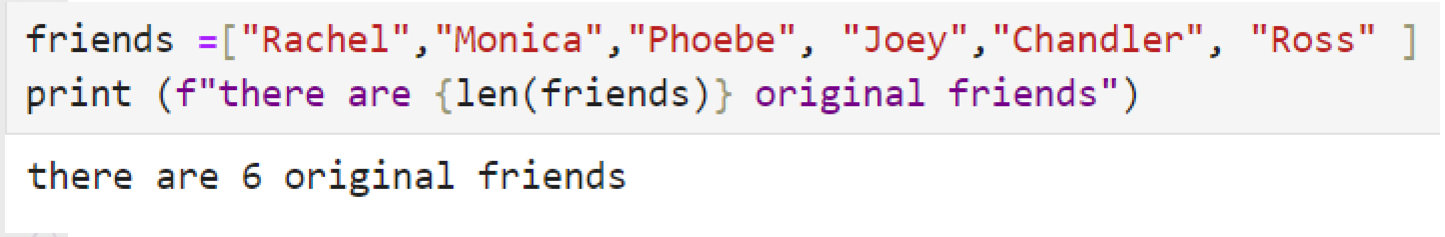
Index and Slicing

- Similar indexing like in string:
  - zero based
  - Can use negative numbers

- Slicing my_list[start: stop: step]
- Can use len()

Len() & List operators

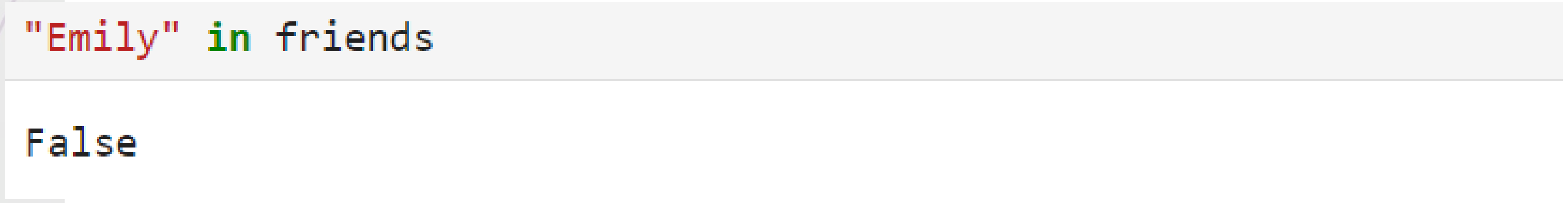
- LEN



```
friends =["Rachel","Monica","Phoebe", "Joey","Chandler", "Ross" ]
print (f"there are {len(friends)} original friends")
```

```
there are 6 original friends
```

- IN



```
"Emily" in friends
```

```
False
```

- Mathematical Operators (+, *)

List methods

•**Adding:**	extend(), insert(), append()
•**Removing:**	remove(), pop(), clear()
•**Accessing:**	index(), count()
•**Modifying:**	sort(), reverse()
•**Copying:**	copy()

Exercise

1) Create a list called shopping_cart with the following items: "bread",”milk”, "eggs", "milk", "butter".

2) Realize you already have "butter" at home, so remove it from the list.

3) You need to buy 3 new items: "flour“, "sugar“ and “milk”, add them to the end of the list using extend().

4)How many “milk”s do you have? (count)

4)Find the index position of "eggs" using index().

5)Sort the list alphabetically using sort().Print the final shopping list.

Tuple

( )
Orderd
Immutable
Allows duplicates

- Tuples are like lists - but immutable (meaning they cannot be changed).
- Once created they will remain constant.
- Use () to create a tuple:

```
a_tuple = (1, 2, 3, ‘a’)
```

- For data we mustn't change

Tuple

- Read value:

```
a_tuple[3] # get ‘a’
```

- Write value → error!

Tuple

- Use cases:
  - Immutable lists.
  - Return values from functions.
  - Assortments that have a logical connection between them, like coordinates.

Set

{ }

NOT Orderd

Immutable*

NO duplicates

methods

Add() – adds a value to the set

Remove() – removes a value from the set

Union() – unites sets , removing the duplicates

Intersection() –returns the duplicates of 2 sets

List	Tuple	Set
[ ]	( )	{ }
ordered	Ordered	Not ordered
mutable	Immutable	immutable
Allows duplicates	Allows duplicates	No duplicates

Dictionary

- Data structure of the form:

```
"key": "value"
```

- Declared by { }.
- Key-value pairs are separated with a comma.
- Key can be number of text, value can be of any type (even another dictionary).
- Key must be unique.

Dictionary - read

- Use `dict.get('key')` to read the value:

- If key not in the dictionary, a default value returns (usually None).
- You can change the default:

```
dict.get('key', {})
```

Will return an empty dictionary

Dictionary - add or edit

- Add:

```
dict[‘new key’] = ‘new value’
```

```python
my_dict = { 'key1': 'value1'}
my_dict['new key'] = 'new value'
print(my_dict)
```

```
{'key1': 'value1', 'new key': 'new value'}
```

- Edit:

```
dict[‘existing key’] = ‘new value’
```

```python
my_dict = {'key': 'old value'}
my_dict['key'] = 'new value'
print(my_dict)
```

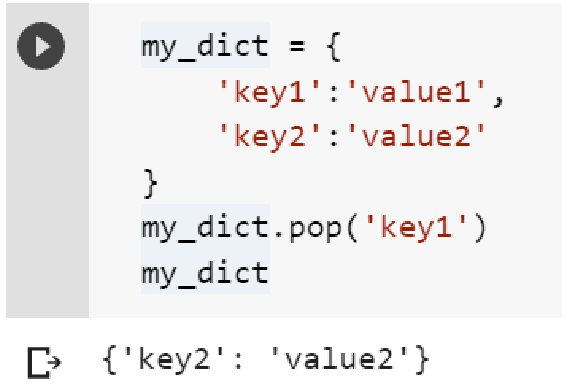
```
{'key': 'new value'}
```

Dictionary - remove

- Remove:

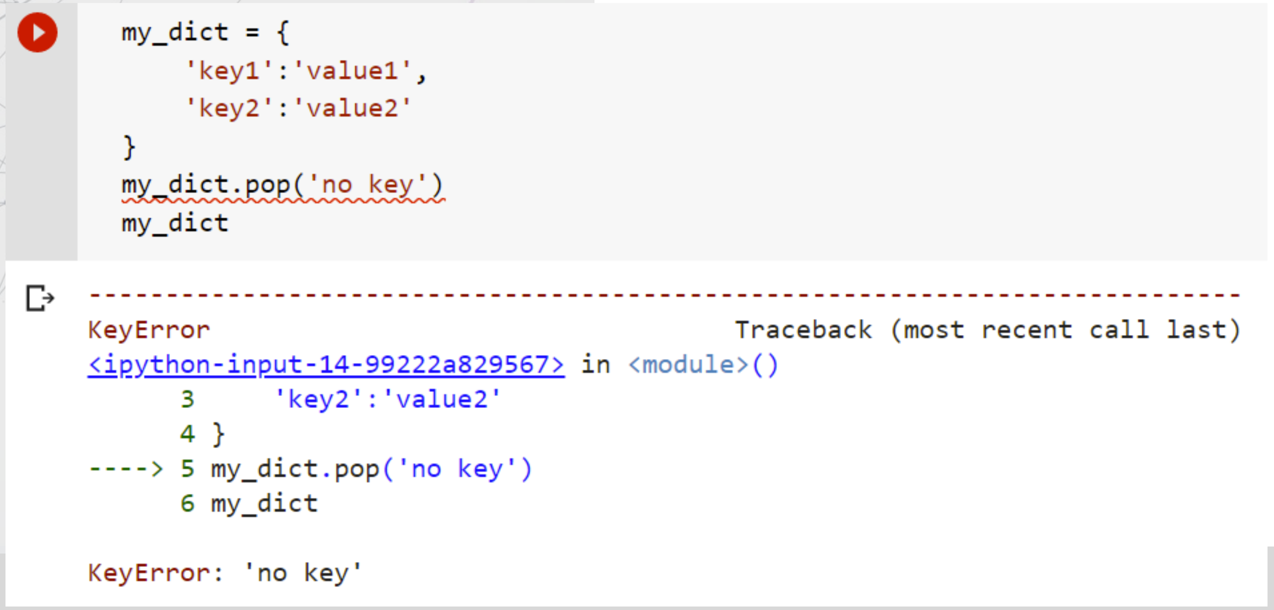
```
dict.pop('key')
```

- If key is not in the dictionary, use default.



```
my_dict = {
    'key1':'value1',
    'key2':'value2'
}
my_dict.pop('key1')
my_dict
```

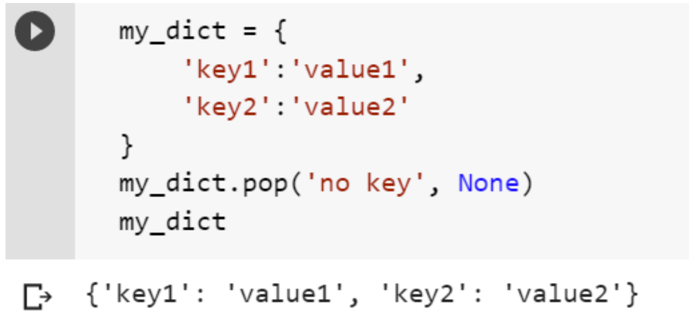
```
{'key2': 'value2'}
```



```
my_dict = {
    'key1':'value1',
    'key2':'value2'
}
my_dict.pop('no key')
my_dict
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-14-99222a829567> in <module>()
      3     'key2':'value2'
      4 }
----> 5 my_dict.pop('no key')
      6 my_dict

KeyError: 'no key'
```



```
my_dict = {
    'key1':'value1',
    'key2':'value2'
}
my_dict.pop('no key', None)
my_dict
```

```
{'key1': 'value1', 'key2': 'value2'}
```

Dictionary – keys and values

- Get all keys:

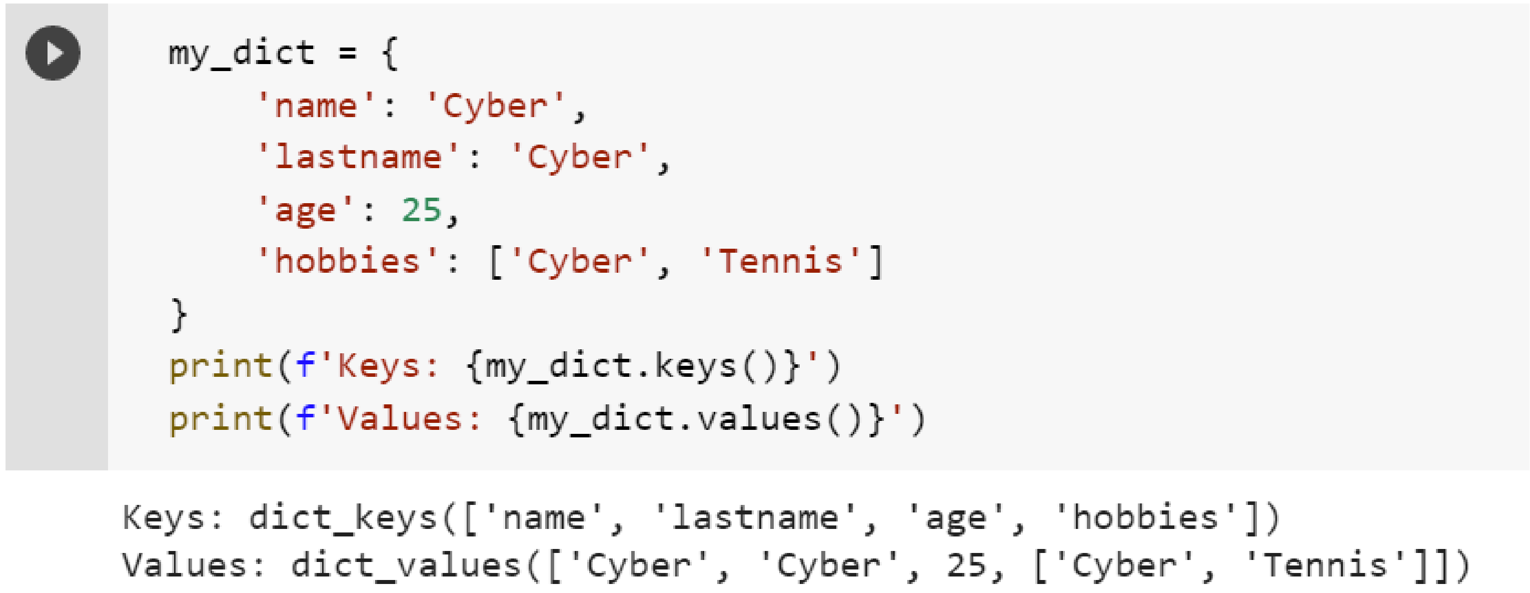
`dict.keys()`

- Get all values:

`dict.values()`

- Get all items

`dict.items()`



```
my_dict = {
    'name': 'Cyber',
    'lastname': 'Cyber',
    'age': 25,
    'hobbies': ['Cyber', 'Tennis']
}
print(f'Keys: {my_dict.keys()}')
print(f'Values: {my_dict.values()}')
```

```
Keys: dict_keys(['name', 'lastname', 'age', 'hobbies'])
Values: dict_values(['Cyber', 'Cyber', 25, ['Cyber', 'Tennis']])
```

Exercise

1. Build a dictionary with the following creatures and their food choices:
    1. A cat drinks milk
    2. A chicken eats seeds
    3. A cow eats grass
    4. A dragon eats people
2. Get a list of the keys in your dictionary (i.e., the names of the creatures).
3. The cat now likes fish. Update the new food choice
4. Try retrieving the value for the key ‘unicorn’.
5. Try removing the key ‘axolotl’ from your dictionary.

Questions?