Week 5

Intro Python

http://cadtx.pw/intropy5

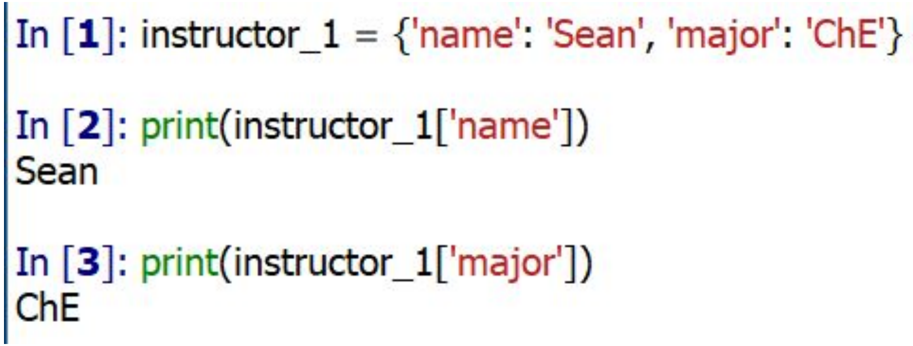
Objectives

Chapter 6

- Dictionaries
- Looping through dictionaries
- Nesting

Dictionaries

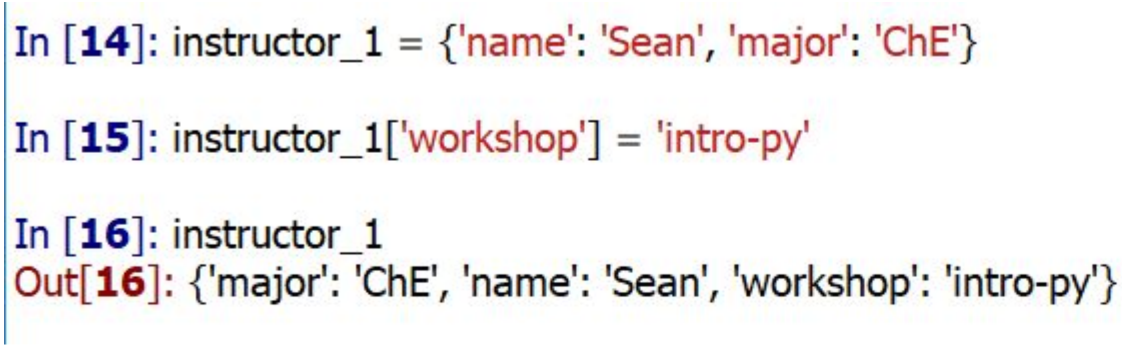
- Dictionaries in Python are collections of *key-value* pairs.
- Keys can be numbers, strings, or even tuples
- Values can be anything - numbers, strings, lists, even another dictionary
- Defined using {} with a series of key-value pairs inside
- Access values using dictionary[key]



```
In [1]: instructor_1 = {'name': 'Sean', 'major': 'ChE'}

In [2]: print(instructor_1['name'])
Sean

In [3]: print(instructor_1['major'])
ChE
```

Adding new Key-Value Pairs

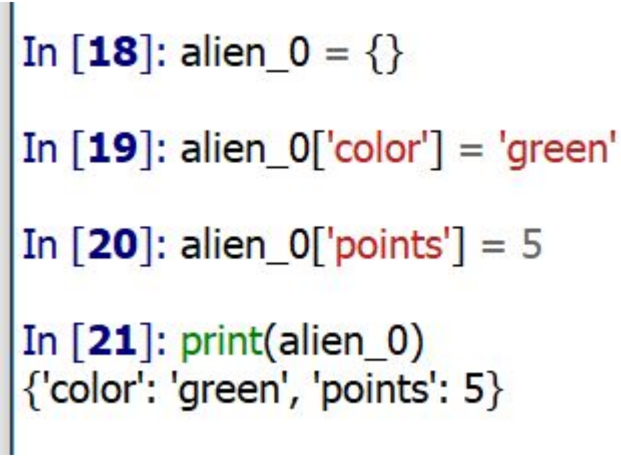
- Like lists, dictionaries are dynamic - you can add new key-value pairs at any time
- Uses the following syntax:
  - Dictionary_name[new_key] = new_value
- Notice that the keys aren’t in the same order we defined them
  - Dictionaries in Python are **unordered**
  - Keys have no index, the only connection is between keys and their values



```
In [14]: instructor_1 = {'name': 'Sean', 'major': 'ChE'}

In [15]: instructor_1['workshop'] = 'intro-py'

In [16]: instructor_1
Out[16]: {'major': 'ChE', 'name': 'Sean', 'workshop': 'intro-py'}
```

Starting with an Empty Dictionary

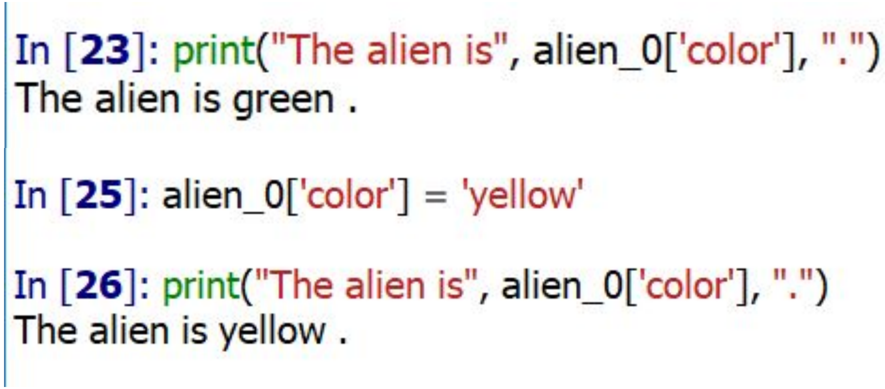
- Sometimes it’s convenient or even necessary to start with an empty dictionary and then add new values to do it
- To define an empty dictionary set a variable equal to {}
- Then add values like we have been doing
- You’ll typically do this when storing user input or code that generates data



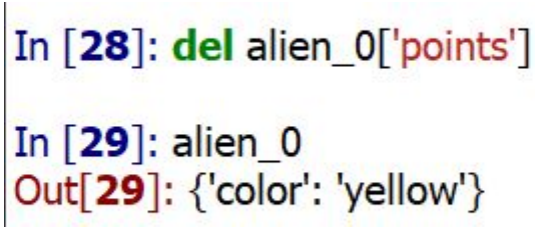
```
In [18]: alien_0 = {}

In [19]: alien_0['color'] = 'green'

In [20]: alien_0['points'] = 5

In [21]: print(alien_0)
{'color': 'green', 'points': 5}
```

Modifying Values in a Dictionary

- You can modify values by simply reassigning the key to a new value
  - Dictionary[key] = new_value
- You can remove key-value pairs using the **del** keyword
  - del dictionary[key]



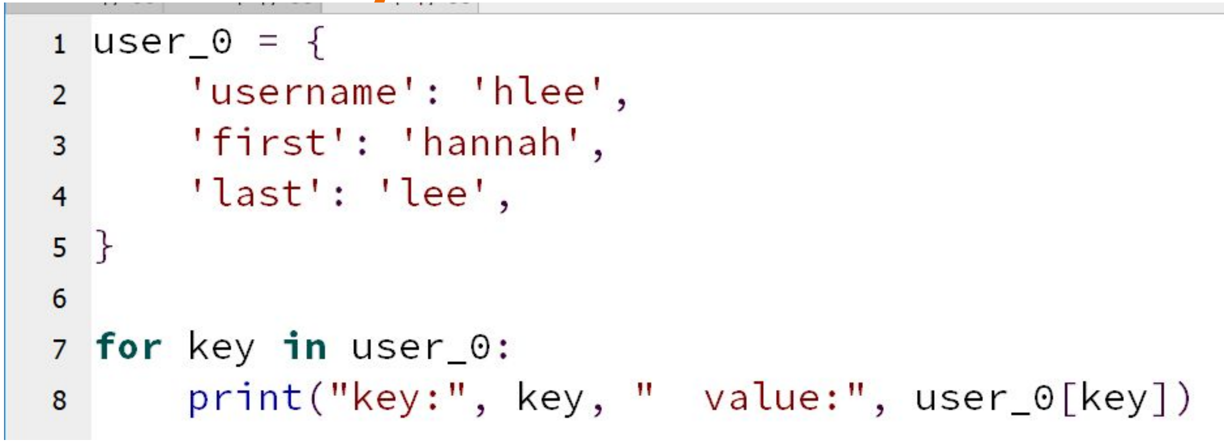
```
In [23]: print("The alien is", alien_0['color'], ".")
The alien is green .

In [25]: alien_0['color'] = 'yellow'

In [26]: print("The alien is", alien_0['color'], ".")
The alien is yellow .
```



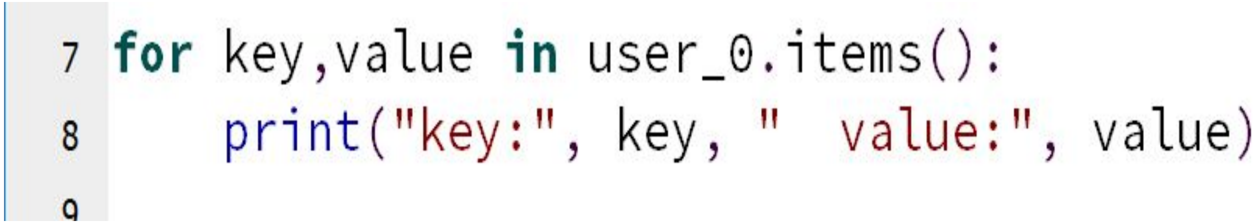
```
In [28]: del alien_0['points']

In [29]: alien_0
Out[29]: {'color': 'yellow'}
```

Looping Through a Dictionary

- We can loop through dictionaries using for loops like before
- By default keys are looped over
- Typing out the dictionary’s name like this everytime is cumbersome - a more efficient way to loop over dictionaries is to use .items()
-

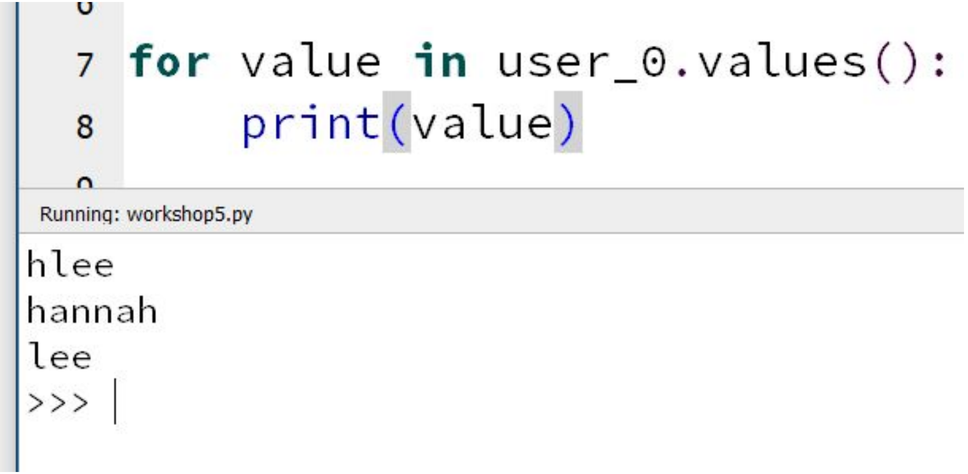


```
user_0 = {
    'username': 'hlee',
    'first': 'hannah',
    'last': 'lee',
}

for key in user_0:
    print("key:", key, "  value:", user_0[key])
```



```
for key,value in user_0.items():
    print("key:", key, "  value:", value)
```

- You can also loop through a dictionary’s values using .values()



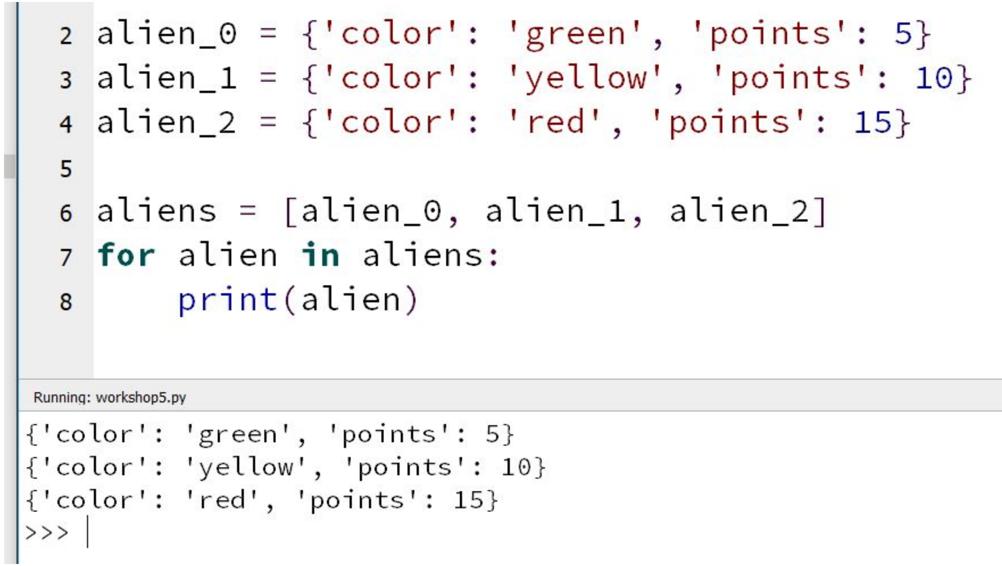
```python
for value in user_0.values():
    print(value)
```

Running: workshop5.py

```
hlee
hannah
lee
>>> 
```

Nesting

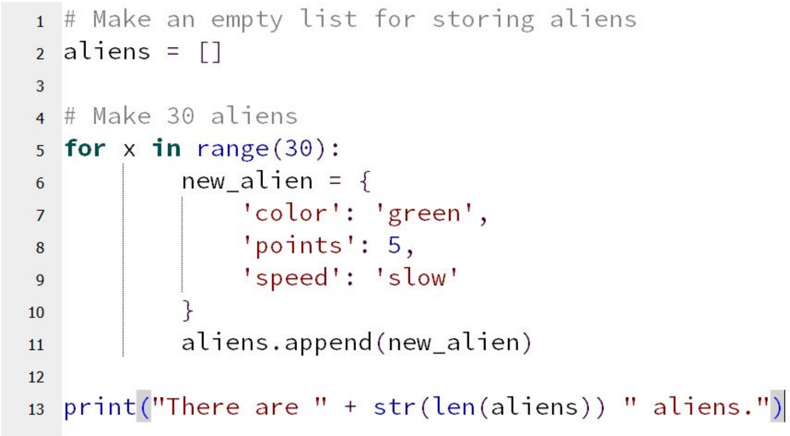
- Sometimes you'll want to store a set of dictionaries in a list or a list of items as a value
    - You can nest lists or dictionaries within a dictionary
- For example, if you wanted to keep track of all of the aliens in a game



```python
alien_0 = {'color': 'green', 'points': 5}
alien_1 = {'color': 'yellow', 'points': 10}
alien_2 = {'color': 'red', 'points': 15}

aliens = [alien_0, alien_1, alien_2]
for alien in aliens:
    print(alien)
```

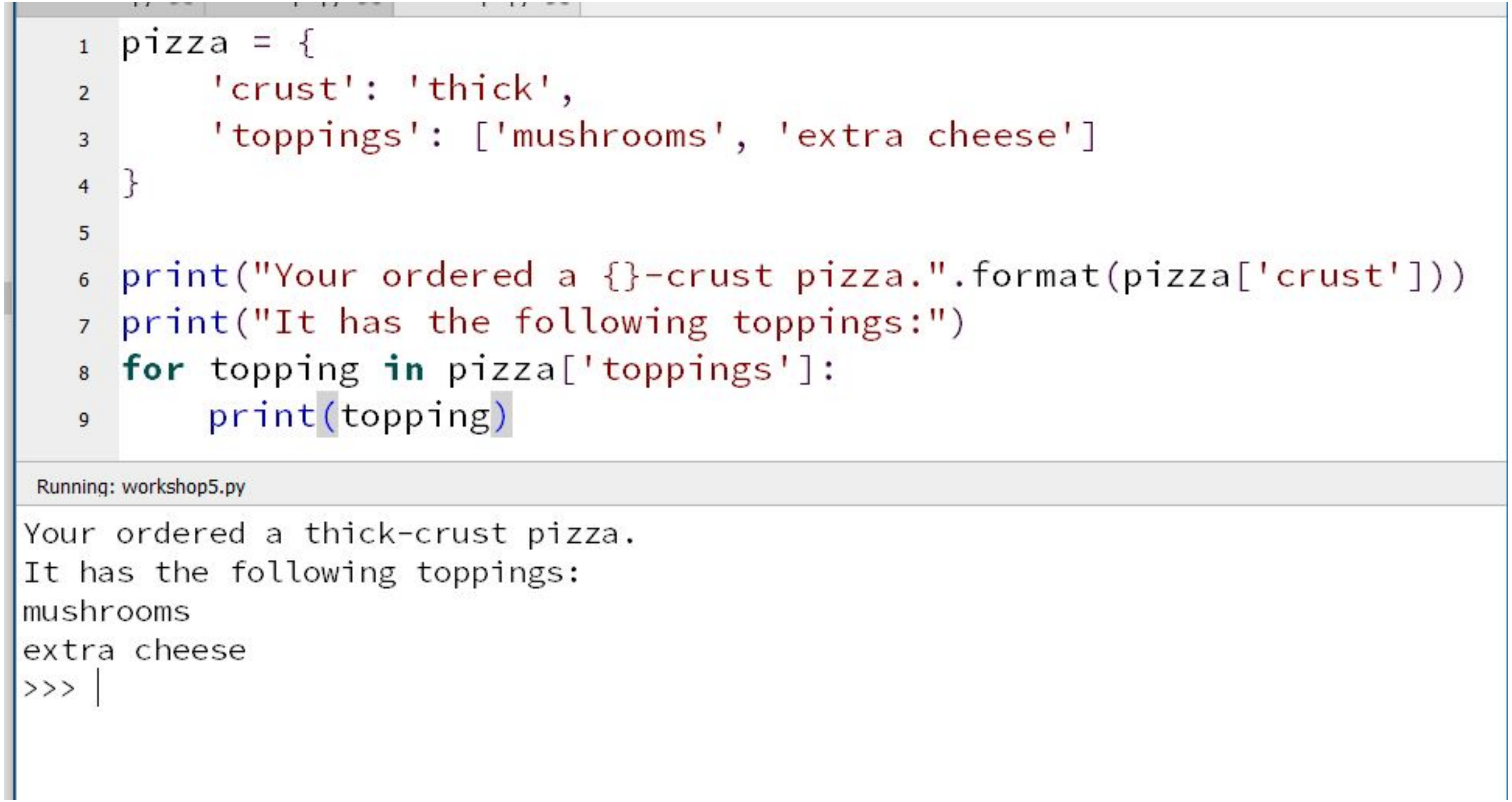
Running: workshop5.py

```
{'color': 'green', 'points': 5}
{'color': 'yellow', 'points': 10}
{'color': 'red', 'points': 15}
>>> 
```



```python
# Make an empty list for storing aliens
aliens = []

# Make 30 aliens
for x in range(30):
        new_alien = {
            'color': 'green',
            'points': 5,
            'speed': 'slow'
        }
        aliens.append(new_alien)

print("There are " + str(len(aliens)) " aliens.")
```

- You can also nest lists within dictionaries

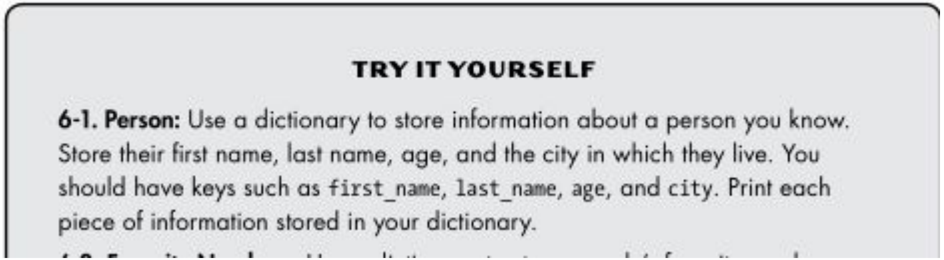


```
pizza = {
    'crust': 'thick',
    'toppings': ['mushrooms', 'extra cheese']
}

print("Your ordered a {}-crust pizza.".format(pizza['crust']))
print("It has the following toppings:")
for topping in pizza['toppings']:
    print(topping)
```

Running: workshop5.py

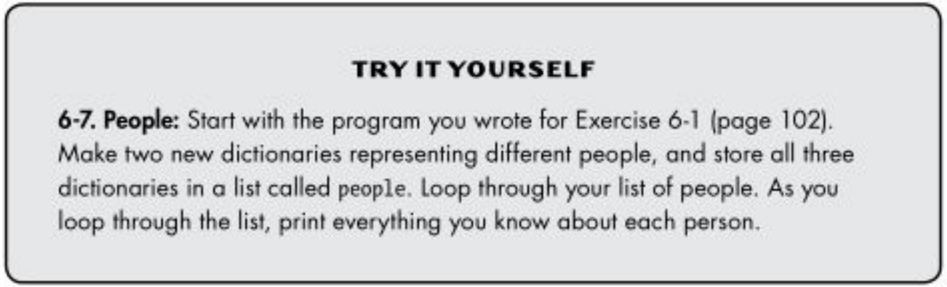
```
Your ordered a thick-crust pizza.
It has the following toppings:
mushrooms
extra cheese
>>> 
```

Exercises



TRY IT YOURSELF

6-1. Person: Use a dictionary to store information about a person you know. Store their first name, last name, age, and the city in which they live. You should have keys such as `first_name`, `last_name`, `age`, and `city`. Print each piece of information stored in your dictionary.



TRY IT YOURSELF

6-7. People: Start with the program you wrote for Exercise 6-1 (page 102). Make two new dictionaries representing different people, and store all three dictionaries in a list called `people`. Loop through your list of people. As you loop through the list, print everything you know about each person.