DICTIONARIES

Objectives

- Describe, create and access a dictionary data structure
- Use built in methods to modify and copy dictionaries
- Iterate over dictionaries using loops and dictionary comprehensions
- Compare and contrast dictionaries and lists

Limitations of Lists

```
instructor = ["Colt", True, 4, "Python", False]
```

Not enough information!

We want to describe this data in more detail!

Introducing....

A dictionary!

A data structure that consists of key value pairs.

We use the keys to describe our data and the values to represent the data

Our First Dictionary

```
instructor = {
    "name": "Colt",
    "owns_dog": True,
    "num_courses": 4,
    "favorite_language": "Python",
    "is_hilarious": False,
    44: "my favorite number!"
}
```

keys and values separated by a colon

Our keys are almost always numbers or strings

Our values can be anything!

Another Way

Another approach is to use the **dict** function. You assign values to keys by passing in keys and values separated by an =

```
another_dictionary = dict(key = 'value')
another_dictionary # {'key': 'value'}
```

Accessing Individual Values

```
instructor = {
    "name": "Colt",
    "owns_dog": True,
    "num_courses": 4,
    "favorite_language": "Python",
    "is_hilarious": False,
    44: "my favorite number!"
}
```

```
instructor["name"] # "Colt"

instructor["thing"] # KeyError
```

There is a more forgiving dictionary method we'll learn about later!

Let's get all the values!

```
instructor = {
    "name": "Colt",
    "owns_dog": True,
    "num_courses": 4,
    "favorite_language": "Python",
    "is_hilarious": False,
    44: "my favorite number!"
}
```

```
print(instructor["name"])
print(instructor["owns_dog"])
print(instructor["num_courses"])
print(instructor["favorite_language"])
print(instructor["is_hilarious"])
print(instructor[44])
```

No, no, no.....

Accessing All Values in a Dictionary

Use **.values()**

```
instructor = {
    "name": "Colt",
    "owns_dog": True,
    "num_courses": 4,
    "favorite_language": "Python",
    "is_hilarious": False,
    44: "my favorite number!"
}
```

```
for value in instructor.values():
    print(value)

# "Colt"
# True
# 4
# "Python"
# False
# "my favorite number!"
```

Accessing All Keys in a Dictionary

Use **.keys()**

```
instructor = {
    "name": "Colt",
    "owns_dog": True,
    "num_courses": 4,
    "favorite_language": "Python",
    "is_hilarious": False,
    44: "my favorite number!"
}
```

```
for key in instructor.keys():
    print(key)

# name
# owns_dog
# num_courses
# favorite_language
# is_hilarious
# 44
```

Accessing All Keys and Values

Use **.items()**

```
instructor = {
    "name": "Colt",
    "owns_dog": True,
    "num_courses": 4,
    "favorite_language": "Python",
    "is_hilarious": False,
    44: "my favorite number!"
}
```

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

# name "Colt"
# owns_dog True
# num_courses 4
# favorite_language "Python"
# is_hilarious False
# 44 "my favorite number!"
```

Given this dictionary...

```
instructor = {
    "name": "Colt",
    "owns_dog": True,
    "num_courses": 4,
    "favorite_language": "Python",
    "is_hilarious": False,
    44: "my favorite number!"
}
```

Does a dictionary have a key?

```
"name" in instructor # True
"awesome" in instructor # False
```

Does a dictionary have a value?

```
"Colt" in instructor.values() # True
"Nope!" in instructor.values() # False
```

How are Dictionaries Useful?

Dictionaries are a fundamental data structure for organizing and describing data into key-value pairs

Just remember, there's no ordering!

YOUR TURN

Dictionary Methods

Working with dictionaries is very common - there are quite a few things we can do!

clear

Clears all the keys and values in a dictionary:

```
d = dict(a=1,b=2,c=3)
d.clear()
d # {}
```

copy

Makes a copy of a dictionary

```
d = dict(a=1,b=2,c=3)
c = d.copy()
c # {'a': 1, 'b': 2, 'c': 3}
c is d # False
```

fromkeys

Creates key-value pairs from comma separated values:

```
{}.fromkeys("a","b") # {'a': 'b'}
{}.fromkeys(["email"], 'unknown') # {'email': 'unknown'}
{}.fromkeys("a",[1,2,3,4,5]) # {'a': [1, 2, 3, 4, 5]}
```

get

Retrieves a key in an object and return None instead of a KeyError if the key does not exist:

```
d = dict(a=1,b=2,c=3)
d['a'] # 1
d.get('a') # 1
d['b'] # 2
d.get('b') # 2
d['no_key'] # KeyError
d.get('no_key') # None
```

pop

Takes a single argument corresponding to a key and removes that key-value pair from the dictionary. Returns the value corresponding to the key that was removed.

```
d = dict(a=1,b=2,c=3)
d.pop() # TypeError: pop expected at least 1 arguments, got 0
d.pop('a') # 1
d # {'c': 3, 'b': 2}
d.pop('e') # KeyError
```

popitem

Removes a random key in a dictionary:

```
d = dict(a=1,b=2,c=3,d=4,e=5)
d.popitem() # ('d', 4)
d.popitem('a') # TypeError: popitem() takes no arguments (1 given)
```

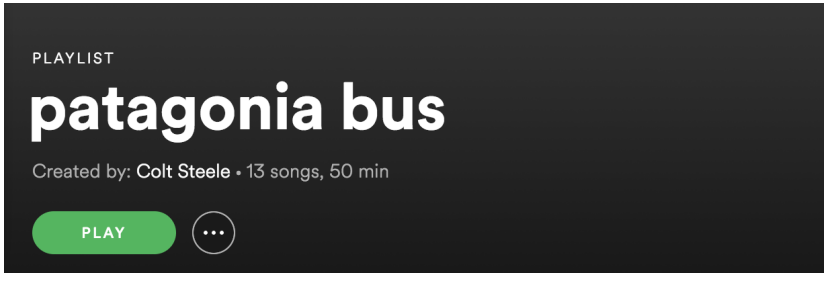
update

Update keys and values in a dictionary with another set of key value pairs.

```
first = dict(a=1,b=2,c=3,d=4,e=5)
second = {}

second.update(first)
second # {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

# let's overwrite an existng key
second['a'] = "AMAZING"

# if we update again
second.update(first) # {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

# the update overrides our values
second # {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
```

DATA MODELING

PLAYLIST MODELING



Turn It Off	Culture Abuse	Peach	2017-10-31	3:37
Eating Hooks - Siriusmo Remix - Solomun Edit	Moderat, Siriusmo, Solomun	Eating Hooks (Siriusmo Remix - Solomun Edit)	2017-11-06	7:02
Nights Off	Siriusmo	Mosaik	2017-11-06	4:08
Tilted - Paradis Remix	Christine and the Queens, Paradis	Tilted (Paradis Remix)	2017-11-06	5:50
Don't Stop	Knightlife	Cut Copy Presents: Oceans Apart	2017-11-06	6:13
Each Moment Like The First	James Holden, The Animal Spirits	The Animal Spirits	2017-11-07	4:57
Renata	James Holden	The Inheritors	2017-11-07	5:57
Sad Saturdays	JOBA	Sad Saturdays	2017-11-07	4:27
Give It To Me (Like You Mean It)	Cub Sport	BATS	2017-11-10	2:50
Wasted Days	Cloud Nothings	Attack On Memory	2017-11-14	8:54

Dictionary Comprehension

the syntax

{ ____:____ for ___ in ____}

- iterates over keys by default
- to iterate over keys and values using .items()

our first example

```
numbers = dict(first=1, second=2, third=3)

squared_numbers = {key: value ** 2 for key,value in numbers.items()}

print(squared_numbers) # {'first': 1, 'second': 4, 'third': 9}
```

more examples

```
{num: num**2 for num in [1,2,3,4,5]}
```

```
str1 = "ABC"
str2 = "123"
combo = {str1[i]: str2[i] for i in range(0,len(str1))}
print(combo) # # {'A': '1', 'B': '2', 'C': '3'}
```

conditional logic with dictionaries

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

{ num:("even" if num % 2 == 0 else "odd") for num in num_list }

# {1: 'odd', 2: 'even', 3: 'odd', 4: 'even'}
```

YOUR
TURN

Recap

- dictionaries are key value pairs that are useful when describing collections and order is not important
- you can create dictionaries with curly braces or the dict function
- you can iterate over dictionaries using keys(), values() and items()
- use methods like get to handle errors more gracefully than directly accessing keys in a dictionary
- dictionary comprehension is useful for creating dictionaries from other data structures