Dictionaries

Chapter 11

Today’s Outline

- Couple of warmups on lists.
- Talk about Dictionaries!
- Use cases and metadata.
- Experiment with different functions that allow you to access keys, values, etc.
- Exercises traversing through dictionaries.

List Exercise 1

- Create a list of prime numbers.
    - `prime_nums = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]`
- Write a script that, for each number in the list:
    - Prints the square of the number (e.g. that number times itself)
- My answer:

```
for num in prime_nums:
    print(num**2)
```

List Exercise 2

- Write a Python function to print all the elements in a list. Use the following structure.

```
# define a function that accepts a list as an argument
# for each element in the list
# print the element
```

To call the function

```
# Step 1 - Call the function by creating a list as an argument
# Step 2 - Create a variable and list and pass the variable as
an argument
```

Exercise 2 Answer

- Write a Python function to print all the elements in a list. Use the following structure.

```
def dosomething( thelist ):
    for element in thelist:
            print element
```

To call the function

```
dosomething([1, 2, 3, 4]) # Step 1

alist = ['red','green','blue']  # Step 2
dosomething( alist )
```

List Exercise 3

- Write a Python function that takes a list and returns a new list with unique elements of the first list.
  - **countrylist = ['UGA', 'KEN', 'TZA', 'TZA', 'SDN', 'KEN', 'KEN’]**

```
UGA
KEN
TZA
SDN
```

  - This is a common “data munging” problem: you just want to see the unique values for a given dataset.

List Exercise 3

Line by line structure for the function.

```
# create countrylist. This is your existing list.
# Define a function to accept a list as an argument.
# initialize a new, null list (list with nothing in it).
# traverse all elements - for each element in an existing list:
# check if element exists in null list or not
# if the element is not in the null list:
# append the null list with the element - use null_list.append(element)
# for element in null list
# print element
```

Exercise 3 Answer

```
countrylist = ['UGA', 'KEN', 'TZA', 'TZA', 'SDN', 'KEN', 'KEN’]

  def unique(list1):
      unique_list = []
      for x in list1:              # traverse all elements
          if x not in unique_list:
              unique_list.append(x)
      for x in unique_list:
          print(x)

  unique(countrylist)
```

Compound Data Types

- Strings `'Hello World'`
  - made up of small pieces - charcaters
- Lists `['Hello', 'World']`
  - made up of small pieces - elements
- Dictionaries `{'word1':'Hello', 'word2':'Hello'}`
  - Made up of key / value pairs.
- Tuples `('Hello', 'World')`
  - Similar to lists: made up of small pieces - elements

Dictionaries

- **Lists** store objects in an ordered sequence you access via an index.
- **Dictionaries** use “key-value” pairing instead.
- The syntax for this is:
  - **`{‘key1’:’value1’, ‘key2’:’value2’, ‘key3’:’value3’}`**
- Dictionaries *cannot* be sorted!
- Are an *unordered* way to store objects.
  - **`{‘key1’:’value1’, ‘key3’:’value3’, ‘key2’:’value2’}`**

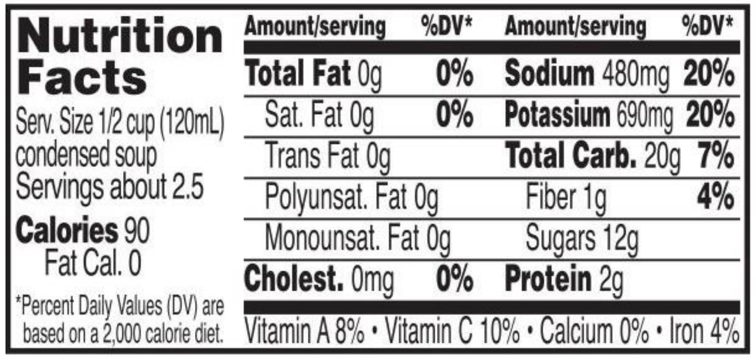
Why use Dictionaries?

- **Dictionaries** are flexible about the data types they can hold: integers, floats, lists other dictionaries.
- Use dictionaries to retrieve a value without needing to know its exact location.
- Dictionary is a good choice to store the data for user inputting name, surname, and age, :
  - `user_info = {'name':'John', 'surname':'Smith', 'age':29} #Integer doesn’t need quotes!`

Why use Dictionaries?

- In this example, we have “meta-data” about the data.
- We know, for example that the value “John” is a name and that the value “Smith” is a last name.
  - `user_info = {'name':'John', 'surname':"Smith', 'age':29}`
- The “key” in this case tells me information about the data.

What is meta data?



Nutrition Facts
Serv. Size 1/2 cup (120mL) condensed soup
Servings about 2.5
Calories 90
Fat Cal. 0

Amount/serving	%DV*	Amount/serving	%DV*
Total Fat 0g	**0%**	**Sodium** 480mg	**20%**
Sat. Fat 0g	**0%**	**Potassium** 690mg	**20%**
Trans Fat 0g		**Total Carb.** 20g	**7%**
Polyunsat. Fat 0g		Fiber 1g	**4%**
Monounsat. Fat 0g		Sugars 12g	
Cholest. 0mg	**0%**	**Protein** 2g	

Vitamin A 8% • Vitamin C 10% • Calcium 0% • Iron 4%

*Percent Daily Values (DV) are based on a 2,000 calorie diet.

Why use Dictionaries?

- Example: using web crawlers and holding data that are related, such as the information contained in an ID or a user profile….
- Create a dictionary for one individual to connect various values with a keyword.
  - `Pat = {'username': 'Pat123', 'online': 'True', 'followers': 987}`
- What about survey responses?
  - `Pat = {'q1':'sometimes', 'q2': 'always', 'q3': 'never'}`

Why use Dictionaries?

- What if I wanted a dictionary of just Twitter user handles?
  - `twitter_users = {'user1': '@Coolio', 'user2': '@Barry', 'user3': '@juniper123'}`
- What if I wanted a dictionary of just Latin tree names?
  - `tree_names_latin= {‘broad_maple’:’acer_amplum', ‘white_oak':’quercus_alba'}`

Create a Dictionary

- Method 1: create a dictionary by providing a list of key-value pairs.

```
price_lookup = {'rice':1.99,'oil':1.99,'chicken':4.89}

print(price_lookup)# What’s the output?

print(price_lookup['oil']) # What’s the output?
```

Create a Dictionary

- Method 2: create Create an empty dictionary and then add elements.

```
eng2fr = {}
```

How do we add key value pairs?

```
eng2fr['one'] = 'une’
eng2fr['two'] = 'deux'
eng2fr[‘three'] = ‘trois'

print(eng2fr)  #What’s the output?
```

Create a Dictionary

- What happens when…

```
eng2fr = {'four':'quatre', 'five':'cinq', 'four':'four'}
print(eng2fr)
```

- When the key already exists, its associated value is replaced. New entries replace old ones with the same key

```
{'four': 'four', 'five': 'cinq'}
```

four:quatre was replaced by four:oven !

Dictionary Operations

- Create a dictionary called "inventory"

```
inventory = {'tents':430, 'stoves': 312, 'beds': 217}
print(inventory)
```

- Delete the 'stoves' element with `del` method.

```
del inventory ['stoves']
print(inventory)
```

- Check the length

```
len(inventory) #in shell
print(len(inventory)) #in script
```

Dictionary Methods

- What's the output for...
- keys()
  - `inventory.keys()                  # In shell`
  - `print(inventory.keys())           # In script window`
  - Returns a list! Notice the square brackets [ ]
- values()
  - `inventory.values()                # In shell`
  - `print(inventory.values())         # In script window`
  - Returns a list! Notice the square brackets [ ]

Dictionary Methods

- What's the output for…
- items()
  - `inventory.items()                # In shell`
  - `print(inventory.items())         # In script window`
  - Returns a tuple! (We'll talk about that next class)

- clear() –
  - `dict.clear(inventory)`

Dictionary Comprehension

- Elegant and concise way to create new dictionary from an iterable in Python.
- Consists of a variable that is an expression inside curly braces `{}` that includes a pair `(key:value)` followed by `for` statement that uses `range()`.
- Example: create dictionary for all squares (number times itself) in a range from 0-5.

```
squares = {x:x*x for x in range(6)}
print(squares)
```

Traversing a Dictionary

- Create this dictionary for a map legend.

```
legend = {0:'no value', 1:'deciduous', 2:'conifers',
3:'industrial', 4:'residential', 5:'water bodies',
6:'agricultural’}
```

- Write a function that will print all the items using the following structure

```
Define a function that takes a dictionary name as an argument:
    for every (key, value) in dictionary - hint: use .items():
        print the key and value
```

Traversing a Dictionary

- Here's my answer

```
def printDictionary(dicInput):
  for (key, value) in dicInput.items():
     print(key, value)

printDictionary(legend)
```

Traversing a Dictionary

Write a function that will print all the keys… using the same structure:

```
def printKeys(dicInput):
  for key in dicInput.keys():
      print(key)

printKeys(legend)
```

Traversing a Dictionary

Write a function that will print all the values… using the same structure:

```
def printValues(dicInput):
  for value in dicInput.values():
        print(value)

printValues(legend)
```

Summary

- Dictionary is a list of key:value pairs.
- Use cases are wanting metadata, or storing related pieces of data.
- Dictionary items are NOT in any real order.
- Dictionary items can be added or deleted or updated.
- Different functions allow you to access keys, values, etc.
- Traversing dictionary examples