**USING DICTIONARIES**

**Using Dictionaries**

Now that we know how to create a dictionary, we can start using already created dictionaries to solve problems.

In this lesson, you’ll learn how to:

* Use a key to get a value from a dictionary
* Check for existence of keys
* Iterate through keys and values in dictionaries

**Get A Key**

Once you have a dictionary, you can access the values in it by providing the key. For example, let’s imagine we have a dictionary that maps buildings to their heights, in meters:

building\_heights = {"Burj Khalifa": 828, "Shanghai Tower": 632, "Abraj Al Bait": 601, "Ping An": 599, "Lotte World Tower": 554.5, "One World Trade": 541.3}

Then we can access the data in it like this:

>>> building\_heights["Burj Khalifa"]  
828  
>>> building\_heights["Ping An"]  
599

**Instructions**

**1.**

We have provided a dictionary that maps the elements of astrology to the zodiac signs. Print out the list of zodiac signs associated with the "earth" element.

Checkpoint 2 Passed

**2.**

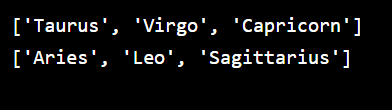
Print out the list of the "fire" signs.

**script.py**

zodiac\_elements = {"water": ["Cancer", "Scorpio", "Pisces"], "fire": ["Aries", "Leo", "Sagittarius"], "earth": ["Taurus", "Virgo", "Capricorn"], "air":["Gemini", "Libra", "Aquarius"]}

print(zodiac\_elements["earth"])

print(zodiac\_elements["fire"])

****

**Get an Invalid Key**

Let’s say we have our dictionary of building heights from the last exercise:

building\_heights = {"Burj Khalifa": 828, "Shanghai Tower": 632, "Abraj Al Bait": 601, "Ping An": 599, "Lotte World Tower": 554.5, "One World Trade": 541.3}

What if we wanted to know the height of the Landmark 81 in Ho Chi Minh City? We could try:

print(building\_heights["Landmark 81"])

But "Landmark 81" does not exist as a key in the building\_heights dictionary! So this will throw a KeyError:

KeyError: 'Landmark 81'

One way to avoid this error is to first check if the key exists in the dictionary:

key\_to\_check = "Landmark 81"  
  
if key\_to\_check in building\_heights:  
  print(building\_heights["Landmark 81"])

This will not throw an error, because key\_to\_check in building\_heights will return False, and so we never try to access the key.

**Instructions**

**1.**

Review the code in the editor and predict what the output will be. Run the code to see if you are correct!

Checkpoint 2 Passed

**2.**

Because "energy" is not a key in zodiac\_elements, a KeyError is thrown in the terminal!

Using an if statement, check if "energy" is a key in zodiac\_elements. Nest the existing print() statement within the if statement so that it will only execute if "energy" is a key.

Run your code again. This time, there should be no errors in the terminal!

Checkpoint 3 Passed

**3.**

Add the key "energy" to the zodiac\_elements. It should map to a value of "Not a Zodiac element". Run the code. Since "energy" is now a key, its value prints to the terminal!

**script.py**

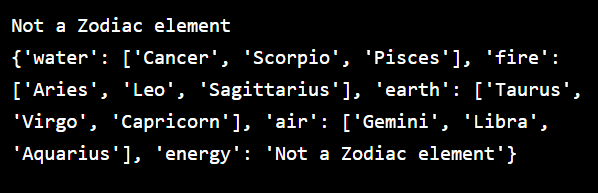
zodiac\_elements = {"water": ["Cancer", "Scorpio", "Pisces"], "fire": ["Aries", "Leo", "Sagittarius"], "earth": ["Taurus", "Virgo", "Capricorn"], "air":["Gemini", "Libra", "Aquarius"]}

zodiac\_elements["energy"] = "Not a Zodiac element"

if "energy" in zodiac\_elements:

  print(zodiac\_elements["energy"])

print(zodiac\_elements)

****

**Try/Except to Get a Key**

We saw that we can avoid KeyErrors by checking if a key is in a dictionary first. Another method we could use is a try/except:

key\_to\_check = "Landmark 81"  
try:  
  print(building\_heights[key\_to\_check])  
except KeyError:  
  print("That key doesn't exist!")

When we try to access a key that doesn’t exist, the program will go into the except block and print "That key doesn't exist!".

**Instructions**

**1.**

Use a try block to try to print the caffeine level of "matcha". If there is a KeyError, print "Unknown Caffeine Level".

Checkpoint 2 Passed

**2.**

Above the try block, add "matcha" to the dictionary with a value of 30.

**script.py**

caffeine\_level = {"espresso": 64, "chai": 40, "decaf": 0, "drip": 120}

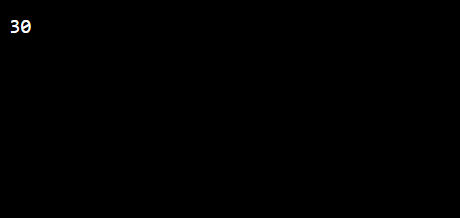
caffeine\_level["matcha"] = 30

try:

  print(caffeine\_level["matcha"])

except KeyError:

  print("Unknown Caffeine Level")

****

**Safely Get a Key**

We saw in the last exercise that we had to add a key:value pair to a dictionary in order to avoid a KeyError. This solution is not sustainable. We can’t predict every key a user may call and add all of those placeholder values to our dictionary!

Dictionaries have a .get() method to search for a value instead of the my\_dict[key] notation we have been using. If the key you are trying to .get() does not exist, it will return None by default:

building\_heights = {"Burj Khalifa": 828, "Shanghai Tower": 632, "Abraj Al Bait": 601, "Ping An": 599, "Lotte World Tower": 554.5, "One World Trade": 541.3}  
  
#this line will return 632:  
building\_heights.get("Shanghai Tower")  
  
#this line will return None:  
building\_heights.get("My House")

You can also specify a value to return if the key doesn’t exist. For example, we might want to return a building height of 0 if our desired building is not in the dictionary:

>>> building\_heights.get('Shanghai Tower', 0)  
632  
>>> building\_heights.get('Mt Olympus', 0)  
0  
>>> building\_heights.get('Kilimanjaro', 'No Value')  
'No Value'

**Instructions**

**1.**

Use .get() to get the value of "teraCoder"‘s user ID, with 100000 as a default value if the user doesn’t exist. Store it in a variable called tc\_id. Print tc\_id to the console.

Checkpoint 2 Passed

**2.**

Use .get() to get the value of "superStackSmash"‘s user ID, with 100000 as a default value if the user doesn’t exist. Store it in a variable called stack\_id. Print stack\_id to the console.

**script.py**

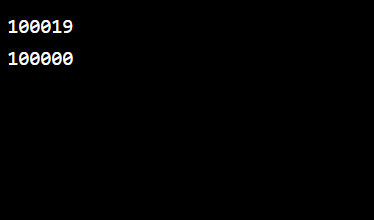
user\_ids = {"teraCoder": 100019, "pythonGuy": 182921, "samTheJavaMaam": 123112, "lyleLoop": 102931, "keysmithKeith": 129384}

tc\_id = user\_ids.get("teraCoder", 1000)

print(tc\_id)

stack\_id = user\_ids.get("superStackSmash", 100000)

print(stack\_id)

****