**CREATING DICTIONARIES**

**Introduction to Python Dictionaries**

A [*dictionary*](https://www.codecademy.com/resources/docs/python/dictionaries?page_ref=catalog) is an unordered set of key: value pairs.

It provides us with a way to map pieces of data to each other so that we can quickly find values that are associated with one another.

Suppose we want to store the prices of various items sold at a cafe:

* Avocado Toast is 6 dollars
* Carrot Juice is 5 dollars
* Blueberry Muffin is 2 dollars

In Python, we can create a dictionary called menu to store this data:

menu = {"avocado toast": 6, "carrot juice": 5, "blueberry muffin": 2}

Notice that:

1. A dictionary begins and ends with curly braces { and }.
2. Each item consists of a key ("avocado toast") and a value (6).
3. Each key: value pair is separated by a comma.

It’s considered good practice to insert a space () after each comma, but our code will still run without the space.

**Instructions**

**1.**

Suppose we have a dictionary of temperature sensors in the house and what temperatures they read. We’ve just added a sensor to the "pantry", and it reads 22 degrees.

Add this pair to the dictionary on line 1.

Checkpoint 2 Passed

Hint

It should look something like:

sensors = {"living room": 21, "kitchen": 23, "bedroom": 20, "pantry": 22}

**2.**

Remove the # in front of the definition of the dictionary num\_cameras, which represents the number of cameras in each area around the house.

If you run this code, you’ll get an error:

SyntaxError: invalid syntax

Try to find and fix the syntax error to make this code run.

Checkpoint 3 Passed

Hint

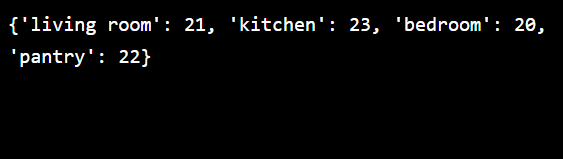
Add commas (,) to num\_cameras so that it runs without errors.

**script.py**

sensors =  {"living room": 21, "kitchen": 23, "bedroom": 20, "pantry": 22}

num\_cameras = {"backyard": 6,  "garage": 2, "driveway": 1}

print(sensors)

****

**Make a Dictionary**

In the previous exercise, we saw a dictionary that maps strings to numbers (i.e., "avocado toast": 6). However, the keys can be numbers as well.

For example, if we were mapping restaurant bill subtotals to the bill total after tip, a dictionary could look like:

subtotal\_to\_total = {20: 24, 10: 12, 5: 6, 15: 18}

Values can be of any [type](https://www.codecademy.com/resources/docs/python/data-types?page_ref=catalog). We can use a string, a number, a list, or even another dictionary as the value associated with a key!

For example:

students\_in\_classes = {"software design": ["Aaron", "Delila", "Samson"], "cartography": ["Christopher", "Juan", "Marco"], "philosophy": ["Frederica", "Manuel"]}

The list ["Aaron", "Delila", "Samson"], which is the value for the key "software design", represents the students in that class.

We can also mix and match key and value types. For example:

person = {"name": "Shuri", "age": 18, "family": ["T'Chaka", "Ramonda"]}

**Instructions**

**1.**

Create a dictionary called translations that maps the following words in English to their definitions in Sindarin (the language of the elves):

| **English** | **Sindarin** |
| --- | --- |
| mountain | orod |
| bread | bass |
| friend | mellon |
| horse | roch |

Checkpoint 2 Passed

Hint

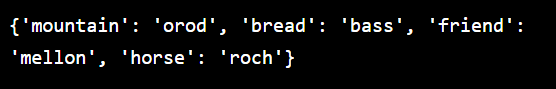
It should look something like:

translations = {"mountain": "orod", "bread": "bass", ...}

**script.py**

translations = {"mountain":"orod", "bread":"bass", "friend":"mellon", "horse":"roch"}

print(translations)

****

**Invalid Keys**

We can have a list or a dictionary as a *value* of an item in a dictionary, but we cannot use these data types as keys of the dictionary. If we try to, we will get a TypeError.

For example:

powers = {[1, 2, 4, 8, 16]: 2, [1, 3, 9, 27, 81]: 3}

This code will yield:

TypeError: unhashable type: 'list'

The word “unhashable” in this context means that this ‘list’ is an object that can be changed.

Dictionaries in Python rely on each key having a *hash value*, a specific identifier for the key. If the key can change, that hash value would not be reliable. So the keys must always be unchangeable, hashable data types, like numbers or strings.

**Instructions**

**1.**

Run the code inside **script.py**. You should get an error:

TypeError: unhashable type

Make the code run without errors by flipping the items in the dictionary so that the strings are the keys and the lists are the values

Checkpoint 2 Passed

Hint

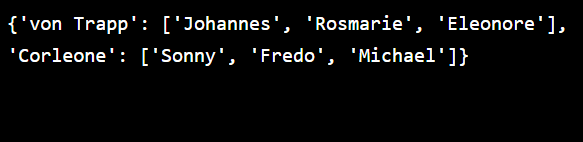
It should look something like:

children = {"von Trapp": ["Johannes", "Rosmarie", "Eleonore"], ...}

**script.py**

children = {"von Trapp": ["Johannes", "Rosmarie", "Eleonore"], "Corleone": ["Sonny", "Fredo", "Michael"]}

print(children)

****

**Empty Dictionary**

A dictionary doesn’t have to contain anything. Sometimes we need to create an empty dictionary when we plan to fill it later based on some other input.

We can create an empty dictionary like this:

empty\_dict = {}

We will explore ways to fill a dictionary in the next exercise.

**Instructions**

**1.**

Create an empty dictionary called my\_empty\_dictionary.

Checkpoint 2 Passed

Hint

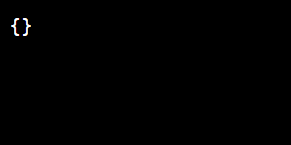
It should look something like:

name = {}

**script.py**

my\_empty\_dictionary = {}

print(my\_empty\_dictionary)

****

**Add A Key**

To add a single key: value pair to a dictionary, we can use the syntax:

dictionary[key] = value

For example, if we had our menu dictionary from the first exercise:

menu = {"oatmeal": 3, "avocado toast": 6, "carrot juice": 5, "blueberry muffin": 2}

And we wanted to add a new item, "cheesecake" for 8 dollars, we could use:

menu["cheesecake"] = 8

Now, menu looks like:

{"oatmeal": 3, "avocado toast": 6, "carrot juice": 5, "blueberry muffin": 2, "cheesecake": 8}

**Instructions**

**1.**

Create an empty dictionary called animals\_in\_zoo.

Checkpoint 2 Passed

**2.**

Walking around the zoo, you see 8 zebras. Add "zebras" to animals\_in\_zoo as a key with a value of 8.

Checkpoint 3 Passed

**3.**

The primate house was bananas! Add "monkeys" to animals\_in\_zoo as a key with a value of 12.

Checkpoint 4 Passed

**4.**

As you leave the zoo, you are saddened that you did not see any dinosaurs. Add "dinosaurs" to animals\_in\_zoo as a key with a value of 0.

Checkpoint 5 Passed

**5.**

Print animals\_in\_zoo.

**script.py**

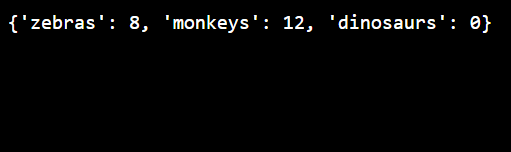
animals\_in\_zoo = {}

animals\_in\_zoo["zebras"] = 8

animals\_in\_zoo["monkeys"] = 12

animals\_in\_zoo["dinosaurs"] = 0

print(animals\_in\_zoo)

****

**Add Multiple Keys**

If we wanted to add multiple key : value pairs to a dictionary at once, we can use the [.update()](https://www.codecademy.com/resources/docs/python/dictionaries/update?page_ref=catalog) method.

Looking at our sensors object from a previous exercise:

sensors = {"living room": 21, "kitchen": 23, "bedroom": 20}

If we wanted to add 3 new rooms, we could use:

sensors.update({"pantry": 22, "guest room": 25, "patio": 34})

This would add all three items to the sensors dictionary.

Now, sensors looks like:

{"living room": 21, "kitchen": 23, "bedroom": 20, "pantry": 22, "guest room": 25, "patio": 34}

**Instructions**

**1.**

In one line of code, add two new users to the user\_ids dictionary:

* theLooper, with an id of 138475
* stringQueen, with an id of 85739

Checkpoint 2 Passed

Hint

It should look something like:

user\_ids.update({"username1": 100, "username2": 101})

**2.**

Print user\_ids.

Checkpoint 3 Passed

Hint

It should look something like:

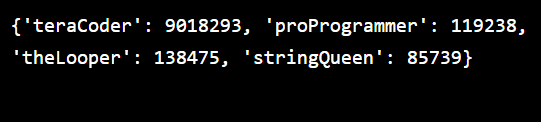
print(name)

**script.py**

user\_ids = {"teraCoder": 9018293, "proProgrammer": 119238}

user\_ids.update({"theLooper": 138475, "stringQueen": 85739})

print(user\_ids)

****

**Overwrite Values**

We know that we can add a key by using the following syntax:

menu["banana"] = 3

This will create a key "banana" and set its value to 3. But what if we used a key that already has an entry in the menu dictionary?

In that case, our value assignment would overwrite the existing value attached to that key. We can overwrite the value of "oatmeal" like this:

menu = {"oatmeal": 3, "avocado toast": 6, "carrot juice": 5, "blueberry muffin": 2}  
menu["oatmeal"] = 5  
print(menu)

This would yield:

{"oatmeal": 5, "avocado toast": 6, "carrot juice": 5, "blueberry muffin": 2}

Notice the value of "oatmeal" has now changed to 5.

**Instructions**

**1.**

Add the key "Supporting Actress" and set the value to "Viola Davis".

Checkpoint 2 Passed

**2.**

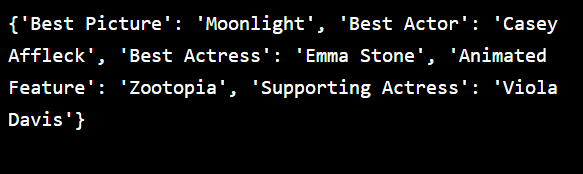
Without changing the definition of the dictionary oscar\_winners, change the value associated with the key "Best Picture" to "Moonlight".

**script.py**

oscar\_winners = {"Best Picture": "La La Land", "Best Actor": "Casey Affleck", "Best Actress": "Emma Stone", "Animated Feature": "Zootopia"}

oscar\_winners["Supporting Actress"] = "Viola Davis"

oscar\_winners["Best Picture"] = "Moonlight"

****

**Dict Comprehensions**

Let’s say we have two lists that we want to combine into a dictionary, like a list of students and a list of their heights, in inches:

names = ['Jenny', 'Alexus', 'Sam', 'Grace']  
heights = [61, 70, 67, 64]

Python allows you to create a dictionary using a dict comprehension, with this syntax:

students = {key:value for key, value in zip(names, heights)}  
#students is now {'Jenny': 61, 'Alexus': 70, 'Sam': 67, 'Grace': 64}

Remember that [zip()](https://www.codecademy.com/resources/docs/python/built-in-functions/zip?page_ref=catalog) combines two lists into an iterator of tuples with the list elements paired together. This dict comprehension:

1. Takes a pair from the iterator of tuples
2. Names the elements in the pair key (the one originally from the names list) and value (the one originally from the heights list)
3. Creates a key : value item in the students dictionary
4. Repeats steps 1-3 for the entire iterator of pairs

**Instructions**

**1.**

You have two lists, representing some drinks sold at a coffee shop and the milligrams of caffeine in each. First, create a variable called zipped\_drinks that is an iterator of pairs between the drinks list and the caffeine list.

Checkpoint 2 Passed

Hint

For example, to create an iterator of pairs between names and heights:

names = ['Jenny', 'Alexus', 'Sam', 'Grace']  
heights = [61, 70, 67, 64]  
   
zipped\_students = zip(names, heights)

**2.**

Create a dictionary called drinks\_to\_caffeine by using a dict comprehension that goes through the zipped\_drinks iterator and turns each tuple pair into a key:value item.

Checkpoint 3 Passed

Hint

Use the following syntax:

dict\_variable = {key: value for key, value in zip\_iterator}

**script.py**

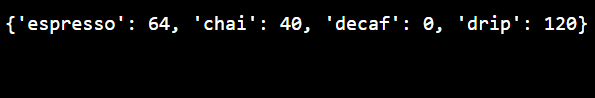
drinks = ["espresso", "chai", "decaf", "drip"]

caffeine = [64, 40, 0, 120]

zipped\_drinks = zip(drinks, caffeine)

drinks\_to\_caffeine = {key:value for key, value in zipped\_drinks}

print(drinks\_to\_caffeine)

****