Dictionary Fundamentals

Dictionaries are key-value pairs. They are similar to lists but instead of using an index to access a value you use a key. Keys are always strings and values can be anything (where in lists they're normally the same thing.)

Why is this important?

Dictionaries are used everywhere in programming. They're used to store data in a way that makes sense. For example, if you were to store a person's information you could use a dictionary to store their name, age, address, etc. You could then use their name as the key to access their information.

Dictionaries are a data type that is also really flexible which makes it very useful. You can add a key with whatever kind of value you want, the value could be a list, another dictionary, a string, a number, etc.

As well in the future (or other classes) you've probably learned about JSON which is a very similar data type to dictionaries, which makes it easier to parse.

What are we going to do?

We're going to take a look at how to create dictionaries and how to access their values. We'll do this by talking about creating a burger builder program.

Steps

1. Let's Create a dictionary and take a look at how we can access their values.

- Create a file called `burger_builder.py`
- A dictionary is created with curly braces `{}`.
    - the key and value are separated by a colon `:`
    - the key needs to be a string.
    - the value can be anything.

```
# burger dictionary
burger = {
    'patties': 1,
    'patty_type': 'beef',
    'cheese': True,
    'toppings': ['lettuce', 'tomato', 'onion', 'pickles', 'ketchup',
'mustard'],
    'bun': 'sesame seed'
}
```

- Let's talk about the above.
    - everything on the **left side** of the colon is a **key** and they are all strings.
    - everything on the **right side** of the colon is a **value** and has different data types.
    - notify that the toppings' value is a list.

2. Let's access the values in the dictionary.

- The way to access items in a dictionary is very similar to a list except instead of using an index we use the key (that is a string), let's take a look at what this looks like.

```
# burger dictionary
burger = {
    'patties': 1,
    'patty_type': 'beef',
    'cheese': True,
    'toppings': ['lettuce', 'tomato', 'onion', 'pickles', 'ketchup',
'mustard'],
    'bun': 'sesame seed'
}

print("Here's what's in your burger:")
print(f"{burger['patties']} {burger['patty_type']} patties is on your
burger.")
```

- Note that we are using our existing `burger` dictionary and we're using square brackets `burger['patties']` to access the value `1` in our dictionary.

- Here's the output of the application when you run it.

```
$ python burger_builder.py
Here's what's in your burger:
1 beef patties is on your burger.
```

- Let's access some other values in the dictionary.

```
# burger dictionary
burger = {
    'patties': 1,
    'patty_type': 'beef',
    'cheese': True,
    'toppings': ['lettuce', 'tomato', 'onion', 'pickles', 'ketchup',
'mustard'],
    'bun': 'sesame seed'
}

print("Here's what's in your burger:")
print(f"{burger['patties']} {burger['patty_type']} patties is on your
burger.")

print(f"Your burger has {burger['bun']} bun.")
if burger['cheese']:
```

```
    print("Cheese is on your burger.")
else:
    print("No cheese on your burger.")
```

- Here's the output of the application when you run it.

```
$ python burger_builder.py
Here's what's in your burger:
1 beef patties is on your burger.
Your burger has sesame seed bun.
Cheese is on your burger.
```

- You can see in the above and the code that we're using the boolean value for the if statement here `burger['cheese']` as our condition.

- Let's loop through the toppings and print them out.

```
# burger dictionary
burger = {
    'patties': 1,
    'patty_type': 'beef',
    'cheese': True,
    'toppings': ['lettuce', 'tomato', 'onion', 'pickles', 'ketchup',
'mustard'],
    'bun': 'sesame seed'
}

# ... code omitted ...

print("here's a list of toppings.")
for topping in burger['toppings']:
    print(f"- {topping}")
```

- Here's the output of the application

```
$ python burger_builder.py
Here's what's in your burger:
1 beef patties is on your burger.
Your burger has sesame seed bun.
Cheese is on your burger.
here's a list of toppings.
- lettuce
- tomato
- onion
- pickles
```

```
- ketchup
- mustard
```

- You can observe that we're using a for loop to loop through the values under the `burger['toppings']` value.
- Extra Note: Since `burger['toppings']` is a list we can use the index to access items in the list. For example:
  - `burger['toppings'][0]` would return `lettuce`
  - `burger['toppings'][1]` would return `tomato`.
- See here that you just put the `[<index>]` right next to the dictionary access `burger['toppings']` and you can access the value at that index. If that value was an array or a dictionary, the same process would continue.

3. Let's reassign the values in the dictionary.

- Right after the dictionary let's reassign the values for the keys: `patties`, `patty_type`, `cheese`, and `bun`.

```
# burger dictionary
burger = {
    'patties': 1,
    'patty_type': 'beef',
    'cheese': True,
    'toppings': ['lettuce', 'tomato', 'onion', 'pickles', 'ketchup',
'mustard'],
    'bun': 'sesame seed'
}
# changing the values in the dictionary
burger['patties'] = 2
burger['patty_type'] = 'vegetarian'
burger['cheese'] = False
burger['bun'] = 'whole wheat'

# ... rest of code omitted ...
```

- You can see here that reassigning the values is very similar to accessing the values except we're using the assignment operator `=` to assign a new value to the key.
- Let's run our application and we'll see how the values have changes.

```
$ python burger_builder.py
Here's what's in your burger:
2 vegetarian patties is on your burger.
Your burger has whole wheat bun.
No cheese on your burger.
here's a list of toppings.
- lettuce
- tomato
- onion
```

```
- pickles
- ketchup
- mustard
```

- Note if you want to see how those values change you can add a `breakpoint()` before the changing values write `burger` in the debug console and you'll see the values change as you step through the code (with `n` or `next`).

4. Let's access a key that might or might not be there.

- Sometimes you'll want to access a key in a dictionary but you're not sure if it exists or not. You can use the `get` method to do this. Let's see what happens when we try to access a key that doesn't exist without the get

```
# burger dictionary
burger = {
    'patties': 1,
    'patty_type': 'beef',
    'cheese': True,
    'toppings': ['lettuce', 'tomato', 'onion', 'pickles', 'ketchup',
'mustard'],
    'bun': 'sesame seed'
}
# changing the values in the dictionary
burger['patties'] = 2
burger['patty_type'] = 'vegetarian'
burger['cheese'] = False
burger['bun'] = 'whole wheat'

# ... code omitted ...

# with a value that doesn't exist.
gluten_allergy = burger['gluten_allergy']
if gluten_allergy:
    print("we have removed the gluten from the burger")
```

- you'll see that in the output we get a `KeyError: 'gluten_allergy'` because we don't have a key called `gluten_allergy`.

```
Here's what's in your burger:
2 vegetarian patties is on your burger.
Your burger has whole wheat bun.
No cheese on your burger.
here's a list of toppings.
- lettuce
- tomato
- onion
```

```
- pickles
- ketchup
- mustard
Traceback (most recent call last):
  File "C:\Users\dmouris\sdev1001-master-
course\dictionaries\dictionary_fundamentals_start\burger_builder.py", line
29, in <module>
    gluten_allergy = burger['gluten_allergy']
                     ~~~~~~^^^^^^^^^^^^^^^^^^
KeyError: 'gluten_allergy'
```

- let's change the line `gluten_allergy = burger['gluten_allergy']` so that it uses the `get` method with a fallback if it's not there.

```
# ... code omitted ...
gluten_allergy = burger.get('gluten_allergy', False)
if gluten_allergy:
    print("we have removed the gluten from the burger")
```

- Now you can see that we're using the `get` method to access the value of the key `gluten_free` and it doesn't exist so we're using the fallback value of `False`. If you run the program you now get no `KeyError`.

```
$ python burger_builder.py
Here's what's in your burger:
2 vegetarian patties is on your burger.
Your burger has whole wheat bun.
No cheese on your burger.
here's a list of toppings.
- lettuce
- tomato
- onion
- pickles
- ketchup
- mustard
```

- Let's add the key `gluten_allergy` to the `burger` dictionary and see what happens.

```
# ... code omitted ...
burger['gluten_allergy'] = True

gluten_allergy = burger.get('gluten_allergy', False)
if gluten_allergy:
    print("we have removed the gluten from the burger")
```

- Now in the output you can see that there's a new line that says `we have removed the gluten from the burger`.

```
$ python burger_builder.py
Here's what's in your burger:
2 vegetarian patties is on your burger.
Your burger has whole wheat bun.
No cheese on your burger.
here's a list of toppings.
- lettuce
- tomato
- onion
- pickles
- ketchup
- mustard
we have removed the gluten from the burger
```

5. Let's check if a key exists in our dictionary.

- Sometimes you'll want to check if a key exists in a dictionary. You can use the `in` operator to do this. Let's see what happens when we try to access a key that doesn't exist without the get
    - we're going to check to see if there's a key called `tomato_allergy` in our dictionary.

```
# ... code omitted ...

if 'tomato_allergy' in burger:
    if burger['tomato_allergy']:
        print("we have removed the tomatoes from the burger")
    else:
        print("we have added tomatoes to the burger")
```

- In the example above we're checking to see that burger has a key called `tomato_allergy` and in the nested if statement it does we're checking to see if the value is `True` or `False` and printing out a message accordingly.

Conclusion

In this example we've learned how to:

- create a dictionary
- access the values in a dictionary
- reassign the values in a dictionary
- use the `get` method to access a key that might not exist.
- use the `in` operator to check if a key exists in a dictionary.

You can see that dictionaries are very useful and they're used everywhere in programming. They're used to store data in a way that makes sense.

In the next example, we'll take a look on how to loop through the values in a dictionary.