DICTIONARY BASICS

Dictionaries store key-value pairs, where keys are used to look up values

- **Keys** must be unique & immutable *(simple data types like strings are immutable)*
- **Values** do not need to be unique and can be any data type

EXAMPLE *Looking up the inventory status for goggles*

```
inventory_status = {'skis': 'in stock',
                    'snowboard': 'sold out',
                    'goggles': 'sold out',
                    'boots': 'in stock'}

inventory_status['goggles']
```

Dictionaries are created with curly braces {}
Keys and values are separated by colons :
Key-value pairs are separated by commas ,

To retrieve dictionary values, simply enter the associated key

```
'sold out'
```

- *NOTE: You cannot look up dictionary values or indices*

*The **KeyError** will be returned if a given key is not in the dictionary*

```
inventory_status['in stock']
KeyError: 'in stock'

inventory_status[2]
KeyError: 2
```

DICTIONARY BASICS

Dictionary values can be lists, so to access individual attributes:

1. Retrieve the **list** by looking up its **key**
2. Retrieve the **list element** by using its **index**

EXAMPLE *Looking up the stock quantity for skis*

```
item_details = {'skis': [249.99, 10, 'in stock'],
                'snowboard': [219.99, 0, 'sold out'],
                'goggles': [99.99, 0, 'sold out'],
                'boots': [79.99, 7, 'in stock']}

item_details['skis']
```

```
[249.99, 10, 'in stock']
```

The key is the item name, and the value is a list storing item price, inventory, and inventory status

```
item_details['skis'][1]
```

```
10
```

This returns the second element (index of 1) in the list stored as the value for the key 'skis'

DICTIONARY BASICS

- List Limitations
- Dictionary Basics
- Modifying Dictionaries
- Dictionary Methods
- Nested Dictionaries
- Sets

You can conduct **membership tests** on dictionary keys

```
item_details = {'skis': [249.99, 10, 'in stock'],
                'snowboard': [219.99, 0, 'sold out'],
                'goggles': [99.99, 0, 'sold out'],
                'boots': [79.99, 7, 'in stock']}

print('skis' in item_details)
print('bindings' in item_details)
```

```
True
False
```

And you can **loop through*** them

```
for item in item_details:
    print(item, item_details[item])
```

```
skis [249.99, 10, 'in stock']
snowboard [219.99, 0, 'sold out']
goggles [99.99, 0, 'sold out']
boots [79.99, 7, 'in stock']
```

Note that the items that are being looped over are the dictionary keys

*Prior to Python 3.7, python dictionaries were not ordered – which meant the order of key-value pairs could change each time we iterated through them This isn't the case anymore, but in some workplaces, you may encounter an older version of Python being used