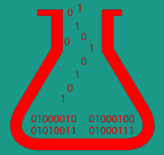
Python Collection Types

Dictionaries



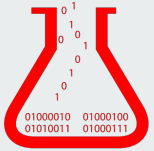
Dictionary (dict)

- A mutable unordered set of key:value pairs, with unique keys
- Syntax: sound_dict = {“cat”: “meow”, “duck”: “quack”}
- To access a given value, call the key:
    - sound_dict[“duck”] → “quack”
- To assign a new key:value pair or reassign an existing key:
    - sound_dict[“cow”] = “moo”
    - print(sound_dict) → {“cat”: “meow”, “duck”: “quack”, “cow”: “moo”}

Dictionaries

- Dictionaries are key:value pairs.
- Think of them like lists, but instead of numeric indexes (0,1,2,3,4…), the keys are arbitrary strings of text (or other hashable type).
- There are multiple syntaxes to make dictionaries
- Keys are unique and immutable (strings)*
- The dictionary is mutable (key:value pairs can be added or removed)

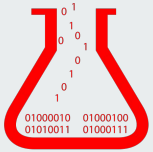
* Dict keys do not have to be strings, though the full definition of what can be used as a key is a bit beyond the scope of this class. Check out this page for more info: https://wiki.python.org/moin/DictionaryKeys



Dictionary instantiation

We can instantiate dictionaries with a few different syntaxes

```
barn_animalweights = {'Cat':10, 'Dog':25, 'Elephant':2000, 'Giraffe':1000}
```

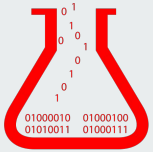


Dictionary instantiation

We can lookup values with bracket notation

```
barn_animalweights = {'Cat':10, 'Dog':25, 'Elephant':2000, 'Giraffe':1000}

print(barn_animalweights['Cat'])
```



Dictionary instantiation

We can lookup values with bracket notation

```
barn_animalweights = {'Cat':10, 'Dog':25, 'Elephant':2000, 'Giraffe':1000}

print(barn_animalweights['Cat'])
```

```
...: print(barn_animalweights['Cat'])
10
```

Dictionary instantiation

We can use bracket notation to modify, a dictionary

```
barn_animalweights = {'Cat':10, 'Dog':25, 'Elephant':2000, 'Giraffe':1000}
```

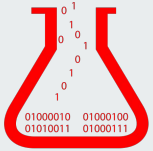
```
 [10]: barn_animalweights['Dog'] = 45
  ...: print(barn_animalweights)
Cat': 10, 'Dog': 45, 'Elephant': 2000, 'Giraffe': 1000}
```

Dictionary instantiation

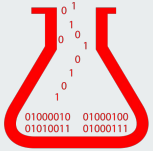
We can also use bracket notation to create a dictionary

```
barn_animalweights = {}
print(barn_animalweights)
#prints an empty dictionary
barn_animalweights['Cat']=10
barn_animalweights['Dog']=25
barn_animalweights['Elephant']=2000
barn_animalweights['Giraffe']=1000
print(barn_animalweights)
```

```
{}
{'Cat': 10, 'Dog': 25, 'Elephant': 2000, 'Giraffe': 1000}
```



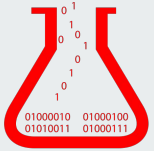
Dictionaries



Dictionaries

- Dictionaries are key:value pairs.
- Think of them like lists, but instead of numeric indexes (0,1,2,3,4…), there are arbitrary strings of text.
- There are multiple syntaxes to make dictionaries
- Keys are unique and immutable (strings)*
- The dictionary is mutable (key:value pairs can be added or removed)

- * Dict keys do not have to be strings, though the full definition of what can be used as a key is a bit beyond the scope of this class. Check out this page for more info: https://wiki.python.org/moin/DictionaryKeys

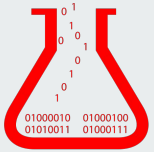


Dictionary exploration

- We made 3 dictionaries
- Consider the bound method ".update()" that can merge 2 dictionaries

```
1 print(farm_dict)
2 print(alt_farm_dict)
3 print(alt_farm_dict2)
```

```
{'donkey': 5, 'horse': 2, 'pig': 10}
{'hippo': 2, 'chicken': 200}
{'horse': 2, 'chicken': 47}
```



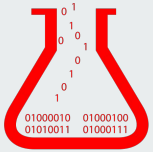
Dictionary mutation

- We change the dictionary by merging a second dictionary with it

```
2 # use the bound method update to change these in place
3 farm_dict.update(alt_farm_dict)
```

```
1 farm_dict
```

```
{'chicken': 200, 'donkey': 5, 'hippo': 2, 'horse': 2, 'pig': 10}
```

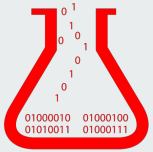


Dictionary exploration

- Get keys, values, or both back

```
4 print(farm_dict.keys())
5 print(farm_dict.values())
6 print(farm_dict.items())
7
```

```
dict_keys(['donkey', 'horse', 'pig', 'hippo', 'chicken'])
dict_values([5, 2, 10, 2, 200])
dict_items([('donkey', 5), ('horse', 2), ('pig', 10), ('hippo', 2), ('chicken', 200)])
```



Dictionary exploration

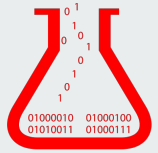
- Recall an item based on its keys this can be done with dictionary notation or using the “get” method

```
1 print("this uses the get method")
2
3 print(farm_dict.get('donkey'))
4
5 print("this is dictionary notation:")
6
7 print(farm_dict['donkey'])
8
```

```
this uses the get method
5
this is dictionary notation:
5
```

Dictionary Methods and Operations

- sound_dict = {“cat”: “meow”, “duck”: “quack”, “cow”: “moo”}
- .keys() - returns a list of the keys of the dictionary
  - sound_dict.keys() → dict_keys(['cat', 'duck', 'cow'])
- .values() - returns a list of the values of the dictionary
  - sound_dict.values() → dict_values(['meow', 'quack', 'moo'])
- .items() - returns a list of tuples of (key,value)
  - sound_dict.items() → dict_items([('cat', 'meow'), ('duck', 'quack'), ('cow', 'moo')])



Advanced Dictionary Method - zip

- **zip()**
- Used to pair up the elements of two lists (or other iterable) based on shared index
  - odd = (1,3,5), even = (2,4,6)
  - print(list(zip(odd, even))) → [(1,2),(3,4),(5,6)]
- Can also be used with dictionaries:
  - students = [“Matt”, “Jane”, “Bob”], grades = [82, 97, 70]
  - print(dict(zip(names, grades))) → {“Matt”:82, “Jane”:97, “Bob”:70}