Dictionaries

A Key-Value Relationship
Dictionaries

Python Dictionaries are a one-way key-value mapping. They are like a list, but elements are accessed using a key, rather than a numerical index.

```
```

Access is similar to a list, but the key replaces the offset:

```
langs["Python"]
langs["C"]
langs[1995]
```

This results in a `KeyError` exception.

How can we add to a dictionary?

Suppose we wanted to add the FORTRAN language:

```
langs["FORTRAN"] = 1957
```
Dictionaries

Python Dictionaries are a one-way key-value mapping. They are like a list, but elements are accessed using a key, rather than a numerical index.

```python
```

Access is similar to a list, but the key replaces the offset:

- `langs["Python"]` is 1991
- `langs["C"]` is 1972
- `langs[1995]` is Error

This results in a `KeyError` exception.

How can we add to a dictionary?

Suppose we wanted to add the FORTRAN language:

```python
langs["FORTRAN"] = 1957
```
Dictionaries

Python Dictionaries are a one-way key-value mapping. They are like a list, but elements are accessed using a key, rather than a numerical index.

```python
```

Access is similar to a list, but the key replaces the offset:

- `langs["Python"]` is 1991
Python Dictionaries are a one-way key-value mapping. They are like a list, but elements are accessed using a key, rather than a numerical index.

```python
```

Access is similar to a list, but the key replaces the offset:

- `langs["Python"]` is 1991
- `langs["C"]` is 1972
- `langs["Java"]` is 1995
- `langs[1995]` is error

This results in a `KeyError` exception.

How can we add to a dictionary?

Suppose we wanted to add the FORTRAN language:

```python
langs["FORTRAN"] = 1957
```
Python Dictionaries are a one-way key-value mapping. They are like a list, but elements are accessed using a key, rather than a numerical index.

```python
```

Access is similar to a list, but the key replaces the offset:

- `langs["Python"]` is 1991
- `langs["C"]` is 1972
Python Dictionaries are a one-way key-value mapping. They are like a list, but elements are accessed using a key, rather than a numerical index.


Access is similar to a list, but the key replaces the offset:

- langs["Python"] is 1991
- langs["C"] is 1972
- langs[1995] is ?

This results in a **KeyError** exception.
Python Dictionaries are a one-way key-value mapping. They are like a list, but elements are accessed using a key, rather than a numerical index.

```python
```

Access is similar to a list, but the key replaces the offset:

- `langs["Python"]` is 1991
- `langs["C"]` is 1972
- `langs[1995]` is Error
  - This results in a `KeyError` exception
Dictionaries

Python Dictionaries are a one-way key-value mapping. They are like a list, but elements are accessed using a key, rather than a numerical index.

```python
```

Access is similar to a list, but the key replaces the offset:

- `langs["Python"]` is 1991
- `langs["C"]` is 1972
- `langs[1995]` is Error
- This results in a `KeyError` exception

How can we add to a dictionary?

Suppose we wanted to add the FORTRAN language:

```python
langs["FORTRAN"] = 1957
```
Consider Dictionaries Like a Table

Having trouble with dictionaries? Think of them like a table, where the **key** is the column you look up an entry by, and the **value** is the column you are looking for.

<table>
<thead>
<tr>
<th>Name (key)</th>
<th>Phone No. (value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>(123) 456-7890</td>
</tr>
<tr>
<td>Bill</td>
<td>(212) 555-1212</td>
</tr>
<tr>
<td>Jane</td>
<td>(444) 555-6666</td>
</tr>
<tr>
<td>Mary</td>
<td>(890) 123-4567</td>
</tr>
<tr>
<td>John</td>
<td>(791) 234-2255</td>
</tr>
</tbody>
</table>
What types of data can the values of a dictionary be?

The values of a dictionary can be of any type. For example, we can nest lists inside dictionaries:

```python
define = {
    "fruits": ["oranges", "apples"],
    "vegetables": ["broccoli", "kale"]
}
```

Practice:
- Define the dictionary above in your interactive interpreter, then evaluate each of the following. What changes?
  1. `foods["meats"] = ["steak", "chicken"]`
  2. `foods["vegetables"][0] = "yum!"`
  3. `len(foods)`
  4. `len(foods["meats"])`
What types of data can the **values** of a dictionary be?

The **values** of a dictionary can be of **any** type. For example, we can nest lists inside dictionaries:

```python
foods = {
    "fruits": ["oranges", "apples"],
    "vegetables": ["broccoli", "kale"]
}
```

**Practice:** Define the dictionary above in your interactive interpreter, then evaluate each of the following. What changes?

1. `foods["meats"] = ["steak", "chicken"]`
2. `foods["vegetables"][0] = "yum!"`
3. `print(len(foods))`
4. `print(len(foods["meats"]))`
The *keys* of a dictionary can be of any *hashable* type. In other words, any data type that can be stored in a set. For example, this is *not* a valid dictionary.

```
oh_noes = {"a", "list"]: 1234}
```
Iterating over a Dictionary

Calling `.keys()` on a dictionary will give us an iterable of the keys. This allows us to loop like this:

```python
systems = {"Windows NT": 1993,
           "Linux": 1991,
           "Mac OS X": 2001}

for key in systems.keys():
    print(key, systems[key])
```

Windows NT 1993
Linux 1991
Mac OS X 2001
The website has an example program using a dictionary as a phone book. Download it, play with it, and maybe even remix your own.
Don’t forget the documentation!

Python also includes a data type for sets. A set is an unordered collection of and eliminating duplicate entities. Set objects also support mathematical

The *Data Structures* page in the official Python documentation has excellent information and examples on using lists, sets, and dictionaries.

These slides are nowhere near complete! Go forth and read the docs!