SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 120: Introduction to Computing Science and Programming 1

Dictionaries

python™

Course Topics

1. General introduction
2. Algorithms, flow charts and pseudocode
3. Procedural programming in Python
4. Data types and Control Structures
5. Binary encodings
6. Fundamental algorithms
7. Basics of (Functions and) Recursion (Turtle Graphics)
8. Basics of Data File management
9. **Basics of computability and complexity: searching /sorting**
10. **More Data Types: Dict**

Today's Topics

1. Dictionaries
2. Sets

Dictionaries

- We have used variables and lists to store data previously.

  For example, **quiz_1 = 14** or

  marks_list = [12, 15, 40, 30]

- **Dictionary**: is another object in Python that stores a **collection of data**.
- We use **{ }** to define data in a dictionary.
- Each element in a dictionary consists of a **key** and a **value.**

  Format: <dictionary_name> = {key1:val1, key2:val2, …}

- Often referred to as mapping of key to value
- To retrieve a specific value, use the **key** associated with it.

Retrieving a Value from a Dictionary

- To retrieve a specific value, use the **key** associated with it.
- General **format** to retrieve a from a dictionary: **dictionary_name[key]**
- If **key** is in the dictionary, associated value is returned, otherwise, **KeyError** exception is raised.
- To test whether a **key** is in a dictionary use the **in** and **not in** operators.
  - These operators can helps prevent **KeyError** exceptions.
- Elements in dictionary are unsorted

Example 1

```
# Declare and define a dictionary.
country_population = {'Canada' : 36624199, 'USA' : 324459463}
```


```
print(country_population["Canada"])
print("{:,}".format(country_population["USA"]))
print(country_population])
```


Adding Elements to an Existing Dictionary

- Dictionaries are **mutable** objects
- To add a new **key-value** pair: `dictionary_name[key] = value`
  - If **key** exists in the dictionary, the value associated with it will be changed. Else, added.

```
country_population = {'Canada' : 36624199, 'USA' : 324459463}

country_population['Mexico'] = 129163276

print(country_population)

{'Canada' : 36624199, 'USA' : 324459463, 'Mexico' : 129163276}
```

Deleting Elements From an Existing Dictionary

- To delete a **key-value** pair: **del dictionary_name[key]**
  - If key is not in the dictionary, KeyError exception is raised

```
country_population = {'Canada' : 36624199, 'USA' : 324459463, 'Mexico' : 129163276}

del country_population['Mexico']

print(country_population)
```

{'Canada' : 36624199, 'USA' : 324459463}

Getting the Number of Elements and Mixing Data Types

- **len** function: Gets you the number of elements in a dictionary.
- **Keys** are immutable objects, but associated **values** are mutable and can be any type of object.
- Values stored in a single dictionary can be of **different types**.

```
country_population['CA_Flag'] = 'Red and White'
print(country_population)
```

Integer type

String

```
{'Canada' : 36624199, 'USA' : 324459463, , 'CA_Flag' : 'Red and white'}
```

Creating Empty Dictionary & Using for Loop to Iterate Over

- To create an empty dictionary: either use **{ }** or use built-in function **dict()**

  empty_dict = { } **or,** **another_empty_dict = dict()**

- We can add elements to the dictionary as program executes,
- Use a for loop to iterate over a dictionary
- General format: **for** key **in** dictionary_name:

For example:

```
country_population = {'Canada' : 36624199, 'USA' : 324459463, 'Mexico' : 129163276}
for  key in country_population:
        print(key)
```


Some Dictionary Methods

- **clear()** method: Deletes all the elements in a dictionary, leaving it empty.
- Format: **dictionary_name.clear()**
- **get()** method: Gets you a **value** associated with specified the specified **key**.
- Format: **dictionary_name.get(key, default)**
  - **default** is returned if the key is not found.

    print(country_population.get('China', 'No Value Found'))
  - Alternative to [ ] operator.
  - Cannot raise KeyError exception.

Some Dictionary Methods - 2

- **items()** method: Returns all the dictionaries **keys** and associated **values**.
- Format: **dictionary_name.items()**
- Returned as a **dictionary view**.
- Each element in **dictionary view** is a **tuple** which contains a key and its associated **value**.
- Use a for loop to iterate over the tuples in the sequence.
- Can use a variable which receives a tuple, or can use two variables which receive key and value.

```
for  key, valuein country_population.items():
        print(key, value)
```


Some Dictionary Methods - 3

- **`keys()`** method: It returns all the dictionaries keys as a sequence.
  - Format: **`dictionary_name.keys()`**

```
for key in country_population.keys()
    print(key)
```

- **`pop()`** method: It returns a value associated with the specified key and **removes** that **key-value** pair from the dictionary.
  - Format: **`dictionary_name.pop(key, default)`**
    - *`default`* is returned if *`key`* is not found

```
value = country_population.pop('Australia', 'Not Found')
```

Some Dictionary Methods - 4

- **`popitem()`** method: It returns a randomly selected key-value pair and removes that key-value pair from the dictionary.
  - Format: **`dictionary_name.popitem()`**

```
Key_value_tuple = country_population.popitem()
```

- **`values()`** method: returns all the dictionaries values as a sequence.
  - Format: **`dictionary_name.values()`**

```
    value = country_population.pop('Australia', 'Not Found')
for  value in country_population.values():
    print(value)
```


Questions?