Introduction to Python

Dictionaries

Topics

1) enumerate()
2) Dictionaries
3) Iterating over a dictionary

Dictionaries

Python lists are useful but in some applications, it is nice to have a different indexing scheme than the integers. For example, consider a database of students' names and their grades:

Mike Smith: [70,81, 84]

Sarah Johnson: [88,71,85]

…

Suppose that this database has hundreds of records. It is hard to access these students' grades using 0-based integer indexing.

Python dictionaries allow "values" to be accessed by meaningful "keys". In the example above, we can access the database of grades by name(keys) instead of integer index.

Dictionaries

Dictionaries are extremely flexible mappings of keys to values, and form the basis of much of Python's internal implementation.

They can be created via a comma-separated list of key:value pairs within curly braces. The "keys" must be distinct.

```python
data = {"Mike":3.1, "Sarah":3.6, "John":3.4}
print(data["Mike"])             # 3.1
gpa = data["John"]
print(gpa)                      # 3.4
print(len(data))                # 3
```

Dictionaries

New items can be added to the dictionary using indexing as well.

```
data = {"Mike":3.1, "Sarah":3.6, "John":3.4}
data['Andy'] = 2.9
print(data)

Output:
{'Mike': 3.1, 'Sarah': 3.6, 'John': 3.4, 'Andy': 2.9}

print(data['Courtney'])  # KeyError
                         # 'Courtney' not in set of keys
```

Dictionaries

Modifying dictionary.

```
data = {"Mike":3.1, "Sarah":3.6, "John":3.4}
data['Mike'] = 3.2
print(data)    # {'Mike': 3.2, 'Sarah': 3.6, 'John': 3.4}
data['Sarah'] += 0.2
print(data)    # {'Mike': 3.2, 'Sarah': 3.8, 'John': 3.4}
```

Membership Operations

By default, membership operations checks keys of a dictionary.

```
scores = {'Mike':5, 'John':2, 'Sarah':4}

print('Mike' in scores)              # True
print(5 in scores)                   # False
print('Michele' not in scores)       # True
```

This will allow use to loop through a dictionary.

Iterating over keys a dictionary

It is easy to iterate over keys of the dictionary. The default loop iterates over the keys.

```python
grades = {"Mike":3.1, "Sarah":3.6, "John":3.4}
for x in grades:
     print(x, end=" ")
```

Output:

```
Mike Sarah John
```

Iterating over keys a dictionary

The following compute the average GPA.

```
grades = {"Mike":3.1, "Sarah":3.6, "John":3.4}
sum = 0
for student in grades:
     sum += grades[student]
average = sum/len(grades)
```

Note: This [] syntax is the same as the syntax for lists and strings!

Other languages, like Java, has different syntax for accessing different data structures.

Example of a Use for Dictionaries

One use of a dictionary is keep track of frequency count.

```
words = ['baby','shark','do','do','do','do','do','do']

frequency = {}
for word in words:
     if word not in frequency:
          frequency[word] = 1
     else:
          frequency[word] += 1
print(frequency)
```

Output:

```
{'baby': 1, 'shark': 1, 'do': 6}
```

We will use this code to do word frequency analysis of the works of Shakespeare!

References

1) Vanderplas, Jake, A Whirlwind Tour of Python, O’reilly Media.