Dictionaries

Python
Dictionaries

- Python has an efficient association data structure — the dictionary
  - Dictionary pairs keys with values
    - Useful for: indices
    - Useful for: translations
    - Useful for: quick lookups
      - E.g.: first letters -> full email address
      - E.g.: human-readable URL -> IP address
      - ...

Dictionaries

- Dictionaries are key-value stores
  - Keys — anything, but needs to be immutable
    - Remember: Lists are mutable, strings are immutable
  - Value — anything
Dictionaries

- Dictionaries are created by using curly brackets
  - Can use lists
    
    ```python
    dicc = {1: 'uno', 2: 'dos', 3: 'tres'}
    ```
  - Or can use assignment
    
    ```python
    dicc = {}
    dicc[1] = "uno"
    dicc[2] = "dos"
    dicc[3] = "tres"
    ```
- Values are assigned / retrieved using the bracket notation
Dictionary

- Dictionary `dicc={}`

- Accessing values:
  
  `dicc[‘key’]`

- With default value

  `dicc.get(key, default_value)`

- Or with if - else

  `if key in dicc:`

- Creating / changing values

  `dicc[‘key’] = value`
Dictionary

- Deleting from a dictionary
  
  ```python
  dicc = {}
  ```

- Use the `del` keyword
  - Raises a key error if the key is not in the dictionary
    ```python
    if key in dicc:
        del dicc[key]
    ```

- Use the `pop` method, which returns the value
  ```python
  value = dicc.pop(key)
  value = dicc.pop(key, default)
  ```
Dictionary

• Checking for existence
  • Use the “in” keyword

```python
>>> dicc = {1: "uno", 2: "dos", 3: "tres"}
>>> 1 in dicc
True
>>> 4 not in dicc
True
```
Dictionaries

- A simple program that “learns” Spanish words

```python
def test():
    dicc = {}
    while True:
        astr = input("Enter an English word: ")
        if astr == "Stop it":
            return
        elif astr in dicc:
            print(dicc[astr])
        else:
            print("I have not yet learned this word")
            val = input("Please enter the Spanish word: ")
            dicc[astr] = val
```
Dictionaries

- Dictionaries have an internal structure
  - You will learn in Data Structures how to build dictionaries yourselves
  - For the moment, enjoy their power
- You can print dictionaries
  - You will notice that they change structure after inserts and not reflect the order in which you inserted elements
  - This is because they optimize access
Dictionaries

• Deleting all entries in a dictionary
  • use the `clear()` method

• Deleting an entry without fear of creating a key error
  • Use an if statement
  • Use `pop` with a second argument `None`
    • `dicc.pop(1, None)`
Dictionaries

- Looping over keys
  - Simplest:
    - for number in dicc:
  - iterkeys() or iter works the same way
    - for number in dicc.iterkeys():
    - for number in iter(dicc):
Some Uses of Dictionaries

• Dictionaries can be used to count things.
  • Example: Count the number of letters in a file.
  • We open the file with encoding latin-1 so that there are no encoding errors

```python
alphabet = "abcdefghijklmnopqrstuvwxyz"

with open("alice.txt", encoding = "latin-1") as infile:
    dicc = {}
    for letter in alphabet:
        dicc[letter]=0
```
Some Uses of Dictionaries

- Create and initialize a dictionary
  - We are only interested in letters

```python
alphabet = "abcdefghijklmnopqrstuvwxyz"

with open("alice.txt", encoding = "latin-1") as infile:
    dicc = {}
    for letter in alphabet:
        dicc[letter] = 0
```
Some Uses of Dictionaries

• Read the file line by line.
  • Read each letter in the line
    • After changing to lower case, update dictionary

```python
alphabet = "abcdefghijklmnopqrstuvwxyz"

with open("alice.txt", encoding = "latin-1") as infile:
    dicc = {}
    for letter in alphabet:
        dicc[letter]=0
    for line in infile:
        for letter in line:
            letter=letter.lower()
            if letter in alphabet:
                dicc[letter]+=1
```
Some Uses of Dictionaries

- Now process the dictionary
  - Calculate the sum of values (i.e. the counts)
  - Pretty-print the results

```python
for letter in alphabet:
    cum += dicc[letter]
for letter in alphabet:
    print("{:1s} {:5d} {:5.2f}%".format(
        letter, dicc[letter], dicc[letter]/cum*100))
```
Some Uses of Dictionaries

• Using lists as dictionary values
  • in order to create an index of words in a file
Some Uses of Dictionaries

- Open file with encoding “latin-1”
- Read file line by line
- Break line into words
- Normalize words by stripping and lowering

```python
with open("alice.txt", encoding = "latin-1") as infile:
    index = {}
    word_count = 0
    for line in infile:
        for word in line.split():
            word = word.lower().strip(".,;:?[]'-\"")
            word_count += 1
            index[word] = index.get(word, 0) + 1
```
Some Uses of Dictionaries

• Add word to dictionary if long enough

```python
with open("alice.txt", encoding = "latin-1") as infile:
    index = {}
    word_count = 0
    for line in infile:
        for word in line.split():
            word_count += 1
            word = word.lower().strip(",,.;:!?[]'-\"")
            if len(word)>7:
                if word in index:
                    index[word].append(word_count)
                else:
                    index[word] = [word_count]
```
Some Uses of Dictionaries

- Print out results if word is frequent enough

```python
for word in index:
    if len(index[word]) > 2:
        print(word, index[word])
```