Contents

• Dictionary
Dictionary - a reference source in print or electronic form containing words usually alphabetically arranged along with information about their forms, pronunciations, functions, etymologies, meanings, and syntactical and idiomatic uses.

Electronic dictionary:

**Cat** --- Scientific name: *felis catus*. A cat is a small, meat-eating animal that is often kept as a pet.

**Python** --- Scientific name: genus: *python*. Pythons are non-venomous snakes that usually kill their prey by wrapping around them and suffocating them.
Dictionary

Dictionary – is a list of pairs (word ~ definition)

Python dictionary – is a list of pairs (key : value)

• The Python dictionary works the same way as the actual dictionary - you look up a key and get its value(s).

• The idea is to organize information for fast retrieval.

• Very efficient implementation.
A Python dictionary is an extremely useful data storage construct for organizing information in pairs (key: value). The association of a key and a value is called a **key-value pair** or an item.

 Analog to a hashtable in Java. A **hash** is a function that takes a value (of any kind) and returns an integer (that is the memory location of the key). Dictionaries use these integers, called hash values, to store and look up key-value pairs.

 The difference is that in Python the **key can be any immutable type** (string, number, tuple).

 Any key is a **unique** string, number or tuple. A key can have associated one or more values in a list. The values of a dictionary can be any Python data type.

 Since dictionaries are mutable, they can’t be used as keys, but they **can** be used as values.
Dictionary

• Syntax similar to list indexing
• Like lists dictionaries can easily be changed, can be shrunked and grown at run time.
• Dictionaries can be contained in lists and vice versa.
• But what's the difference between lists and dictionaries?
  – Lists are ordered sets of objects, whereas dictionaries are unordered sets.
  – But the main difference is that items in dictionaries are accessed via keys and not via their position.
• Rich set of built-in operations
Dictionary

• Dictionary is another data type in Python. It is a collection of items that maps a "key" to a "value".

• Creating a dictionary

dictionary = {key1:value1, key2:value2, key3:value3}

Example:
mydictionary1 = {0:"first",1:"second",2:"third"}
mydictionary2 = {"one":1,"three":3,"two":2}

• The empty dictionary: emptyDict = {}

• Dictionary with a single item: dict1 = {"abc": 45.6}
Access dictionary elements

data = {'Name':'Zara','Age':7,'Class':'First'}

# obtaining value at key 2
print(data['Age'])  # 7

# Get all values
data.values()  # returns list of the values

# obtaining a value at key key
data.get('Age')  # returns None if no such key exists

# Get all keys
data.keys()  # returns list of the keys
Access dictionary elements

• **items()** - returns a list of all items. Each item is a (key, value) pair or tuple.

• This is the most efficient way to examine all the key value data in the dictionary.

• All of these lists can be passed to the sorted() function.

• Basic syntax for looping through dictionaries (keys and values)

```python
for key in data:
    print(data[key])
```
Access dictionary elements

data = {
    'key1': 'value1',
    'key2': 'value2',
    'key3': 'value3'
}

for key, value in data.items():
    print(key, value)

for value in data.values():
    print(value)
Access dictionary elements

# build up a dictionary by starting with the empty dictionary `{}` and storing key/value pairs into the dictionary like this:

diction = {}
diction['a'] = 'alpha'  diction['g'] = 'gamma'  diction['o'] = 'omega'

# get a value

### get the values() list of values

### get the keys() list
Access dictionary elements

# build up a dictionary by starting with the empty dictionary {} and storing key/value pairs into the dictionary like this:

```python
import this

diction = {}
diction['a'] = 'alpha'  diction['g'] = 'gamma'  diction['o'] = 'omega'

# get a value
print(diction['a'])
```

```python
## get the values() list of values
print(list(diction.values()))
['alpha', 'omega', 'gamma']
```

```python
## get the keys() list
print(list(diction.keys()))
['a', 'o', 'g']
```
Iterate dictionary elements

## items() is the dict expressed as (key, value) tuples

```python
print(list(diction.items()))
[('a', 'alpha'), ('o', 'omega'), ('g', 'gamma')]
```

## Note that the keys are in a random order.

```python
for key in diction:
    print(key)
```

```python
a g o
```

## Exactly the same as above

```python
for key in diction.keys():
    print(key)
```

## Common case -- loop over the keys in sorted order, accessing each key/value

```python
for key in sorted(diction.keys()):
    print(key, diction[key])
```
Updating a dictionary

- You can update a dictionary by:
  - adding a new entry or item (i.e., a key-value pair) only if the key is not already in dictionary (**unique keys**)
  - modifying an existing entry -> if the key is already present, overwrites the previous value
  - deleting an existing entry

```python
data = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
data['Age'] = 8                    # update existing entry
data['School'] = "DPS School"      # Add new entry
```
Deleting from dictionary

- **del** lets you delete individual items from a dictionary by key.

  ```python
del data[‘Age’]
  ```

- **clear** deletes all items from a dictionary. Note that the set of empty curly braces signifies a dictionary without any items.

  ```python
data.clear()
```
Dictionary Operations

eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}

• Length of dictionary – `len()` it returns the number of key-value pairs

  `len(eng2sp)`

• `in` operator - tells you whether something appears as a key in the dictionary

  `'one' in eng2sp`

  True

  `'uno' in eng2sp`

  False
Dictionary Operations

```python
eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}

# make a copy of the dictionary with copy
copyDict = eng2sp.copy()

# using the if statement to get the values
if "one" in eng2sp:
    print(eng2sp['one'])
```
Combining two dictionaries

- You can combine two dictionaries by using the `update` method of the primary dictionary. Note that the update method will merge existing elements if they conflict.

```python
d = {'apples': 1, 'oranges': 3, 'pears': 2}
ud = {'pears': 4, 'grapes': 5, 'lemons': 6}
d.update(ud)
print(d)
{'grapes': 5, 'pears': 4, 'lemons': 6, 'apples': 1, 'oranges': 3}
```
Building a Dictionary

- The `dict()` constructor builds dictionaries directly from sequences of key-value pairs:
  ```python
dict([(\'sape\', 4139), (\'guido\', 4127), (\'jack\', 4098)])
{'sape': 4139, 'jack': 4098, 'guido': 4127}
```

- When the keys are simple strings, it is sometimes easier to specify pairs using keyword arguments:
  ```python
dict(sape=4139, guido=4127, jack=4098)
{\'sape\': 4139, \'jack\': 4098, \'guido\': 4127}
```
More on Dictionary

• Tuple as keys are ok:

```python
dic = { (1,2,3):"abc", 3.1415:"abc"}
print(dic[(1,2,3)])
```

```
abc
```

• Lists as values are ok:

```python
grades = {'fall2011':['A','B','A-'], 'spring2013':['B+','A']}
print(grades['fall2011'])
```

```
['A','B','A-']
```
Accessing non Existing Keys

• If you try to access a key which doesn't exist, you will get an error message (syntactical error).

• You can prevent this by using the "in" operator:

```python
words = {"house" : "Haus", "cat" : "Katze"}
if "car" in words:
    print(words["car"])
...
Lists from Dictionaries

```python
w={"house":"Haus","cat":"Katze","red":"rot"}

Create lists from dictionaries with

- `items()` - creates a list consisting of tuples of (key,value) pairs

```python
w.items()  # print(list(w.items()))
```

[['house', 'Haus'], ['red', 'rot'], ['cat', 'Katze']]

- `keys()` - creates a list of the keys in the dictionary

```python
w.keys()
```

['house', 'red', 'cat']

- `values()` - creates a list of the values

```python
w.values()
```

['Haus', 'rot', 'Katze']

```
Dictionaries from Lists

dishes = ["pizza", "sauerkraut", "paella", "Hamburger"]
countries = ["Italy", "Germany", "Spain", "USA"]

Use `zip()` then `dict()` to combine the two lists into a dictionary

country_specialities = zip(countries, dishes)
print(list(country_specialities))
[['Italy', 'pizza'], ['Germany', 'sauerkraut'], ['Spain', 'paella'], ['USA', 'Hamburger']]
country_specialities_dict = dict(country_specialities)
print(country_specialities_dict)
{'Germany': 'sauerkraut', 'Spain': 'paella', 'Italy': 'pizza', 'USA': 'Hamburger'}
Dictionaries from Lists

What happens if the two lists do not have the same number of elements?
The superfluous elements will not be used!

countries = ["Italy", "Germany", "Spain", "USA", "Switzerland"]
dishes = ["pizza", "sauerkraut", "paella", "Hamburger"]
country_specialities = zip(countries, dishes)
print(country_specialities)
[('Italy', 'pizza'), ('Germany', 'sauerkraut'), ('Spain', 'paella'), ('USA', 'Hamburger')]

Sorry… The food specialty of Switzerland is not included in today’s class!
Next lecture ...

• File manipulation
• Read/ Write from/to large files