Dictionary in Python

Click to add text

Introduction to Dictionaries

- Dictionaries are used to store data values in key:value pairs.
- Each pair has key and value
- Keys should be unique
- Key and value are separated by :
- Each pair is separated by ,

Example:

```
dict = {'A' : 1234, 'B' : 1235}
dict = {1 : 1234, 2 : 1235}
dict=[1234,1235]
```

Properties of Dictionaries

- unordered **mutable collections**;
- items are **stored** and fetched ***by key***,
- Accessed by key, **not offset position**
- Unordered collections of arbitrary objects
- Variable-length, **heterogeneous**, and arbitrarily **nestable**
- dictionaries are written with curly brackets
- does not allow duplicates

Creating a Dictionary

- Creating an EMPTY dictionary

  dictname = {}

- **Example:**

  Dict1 = {}

  MyDict = {}

  Books = {}

- Creating a dictionary with items

dictname =

{ key1:val1, key2:val2, ….}

Example:

MyDict = { 1 : ‘Chocolate’, 2 : ‘Icecream’}

MyCourse = {‘MS’ : ‘Python’, ‘IT’ : ‘C’, ‘CSE’ : ‘C++’, ‘MCA’ : ‘Java’}

MyCircle = {‘Hubby’:9486028245, ‘Mom’:9486301601}

More examples

```
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
  }
  print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Accessing Values

- >>>MyDict = { 1 : ‘Chocolate’, 2 : ‘Icecream’}

- Using keys within square brackets

  >>> print **MyDict[1]**

  ‘Chocholate’

  >>> print **MyCourse[‘CSE’]**

  ‘C++’

Updating Elements

- **update** by adding a new item (key-value) pair
- **modify an existing entry**

```
>>>MyDict[1] = 'Pizza'
>>>MyCourse['MCA'] = 'UML'
```

Deleting Elements

- remove an element in a dictionary using the key

  >>>**del MyCourse['IT']**

- remove all the elements

  >>>MyCourse**.clear()**

- delete the dictionary

  >>>**del** MyCourse

Basic Operations

```
>>> D = {'spam': 2, 'ham': 1, 'eggs': 3}

>>> len(D) # Number of entries in dictionary 3

>>> 'ham' in D # Key membership test

True

>>> L=list(D.keys())  # Create a new list of D's keys

['eggs', 'spam', 'ham']
```

Basic Operations

```
>>> list(D.values())
[2, 1, 3]
>>> list(D.items())
[('eggs', 3), ('spam', 2), ('ham', 1)]
>>> D.get('spam')           # A key that is there
2
>>> print(D.get('toast'))   # A key that is missing
None
```

Update Method

```
>>> D

{'eggs': 3, 'spam': 2, 'ham': 1}

>>> D2 = {'toast':4, 'muffin':5}

>>> D.update(D2)

>>> D

{'eggs': 3, 'muffin': 5, 'toast': 4, 'spam': 2, 'ham': 1}

#unordered
```

Pop Method

Delete and return value for a given key

```
>>> D = {'eggs': 3, 'muffin': 5, 'toast': 4, 'spam': 2,
   'ham': 1}
>>> D.pop('muffin')
5
>>> D.pop('toast‘)
4
>>> D
{'eggs': 3, 'spam': 2, 'ham': 1}
```

Nesting in dictionaries

```
>>> jobs = []
>>> jobs.append('developer')
>>> jobs.append(‘manager‘)
rec = {}
>>> rec['name'] = 'Bob'
>>> rec['age']  = 40.5
>>> rec['job']  = jobs
```

Nesting in dictionaries

```
>>> rec
{'name': 'Bob', 'age': 40.5, 'job': ['developer', 'manager‘, ’lead’]}
>>>print(rec(‘name’))
‘Bob’
>>>print(rec(‘age’))
40.5
>>>print(rec(‘job’))
['developer', 'manager‘, ’lead’]
>>>print(rec[‘job’][2])
‘lead’
```

Nesting in dictionaries

```
>>> rec['name']
'Bob'
>>> rec['job']
['developer', 'manager']
>>> rec['job'][1]
'manager’
```

Other Ways to Make Dictionaries

```
D = {'name': 'Bob', 'age': 40}
```

```
D = {}              # Assign by keys dynamically
D['name'] = 'Bob'
D['age']  = 40
```

Comprehensions in Dictionaries

This is the general template you can follow for dictionary comprehension in Python:

```
dict_variable = {key:value for (key,value) in
    dictonary.items()}
```

Comprehensions in Dictionaries

```
dict1 = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
# Double each value in the dictionary
double_dict1 = {k:v*2 for (k,v) in dict1.items()}
print(double_dict1)
```

```
{'e': 10, 'a': 2, 'c': 6, 'b': 4, 'd': 8}
```

Comprehensions in Dictionaries

- You can also make changes to the key values. For example, let's create the same dictionary as above but also change the names of the key.

```
dict1_keys = {k*2:v for (k,v) in dict1.items()}
print(dict1_keys)
{'dd': 4, 'ee': 5, 'aa': 1, 'bb': 2, 'cc': 3}
```

Comprehensions in Dictionaries

```
>>> D = {c: c * 4 for c in 'SPAM'}

>>> D

{'S': 'SSSS', 'P': 'PPPP', 'A': 'AAAA', 'M': 'MMMM'}
```

Dictionary methods

- `<dict>.items()`
  - displays the items in the dictionary (pair of keys and values)
- `<dict>.keys()`
  - display the keys in the dictionary
- `<dict>.values()`
  - displays the values in the dictionary
- `<dict>.pop()`
  - removes the last item from the dictionary
- `<dict2> = <dict1>.copy()`
  - copies the items from dict1 to dict2
- `<dict>.clear()`
  - removes all the items from the dictionary

Getting Inputs from user to a Dictionary

```
n = int(input("enter a n value:"))
d = {}

for i in range(n):
    keys = input() # here i have taken keys as strings
    values = input() # here i have taken values as integers
    d[keys] = values
print(d)
```

Problem

Write a kids play program that prints the capital of a country given the name of the country.

PAC For Quiz Problem

Input	Processing	Output
A set of question/ answer pairs and a question	Map each question to the corresponding answer. Find the answer for the given question	Answer for the question

Pseudocode

```
READ num_of_countries
FOR i=0 to num_of_countries
        READ name_of_country
        READ capital_of_country
        MAP name_of_country to capital_of_country
END FOR
READ country_asked
GET capital for country_asked
PRINT capital
```

Already we know

- To read values from user
- Print values
- We have to yet know to
  - Map a pair of values

Python provides Dictionaries to Map a pair of values