Struktur Data

Dictionaries

Husni

husni@trunojoyo.ac.id

http://husni.trunojoyo.ac.id

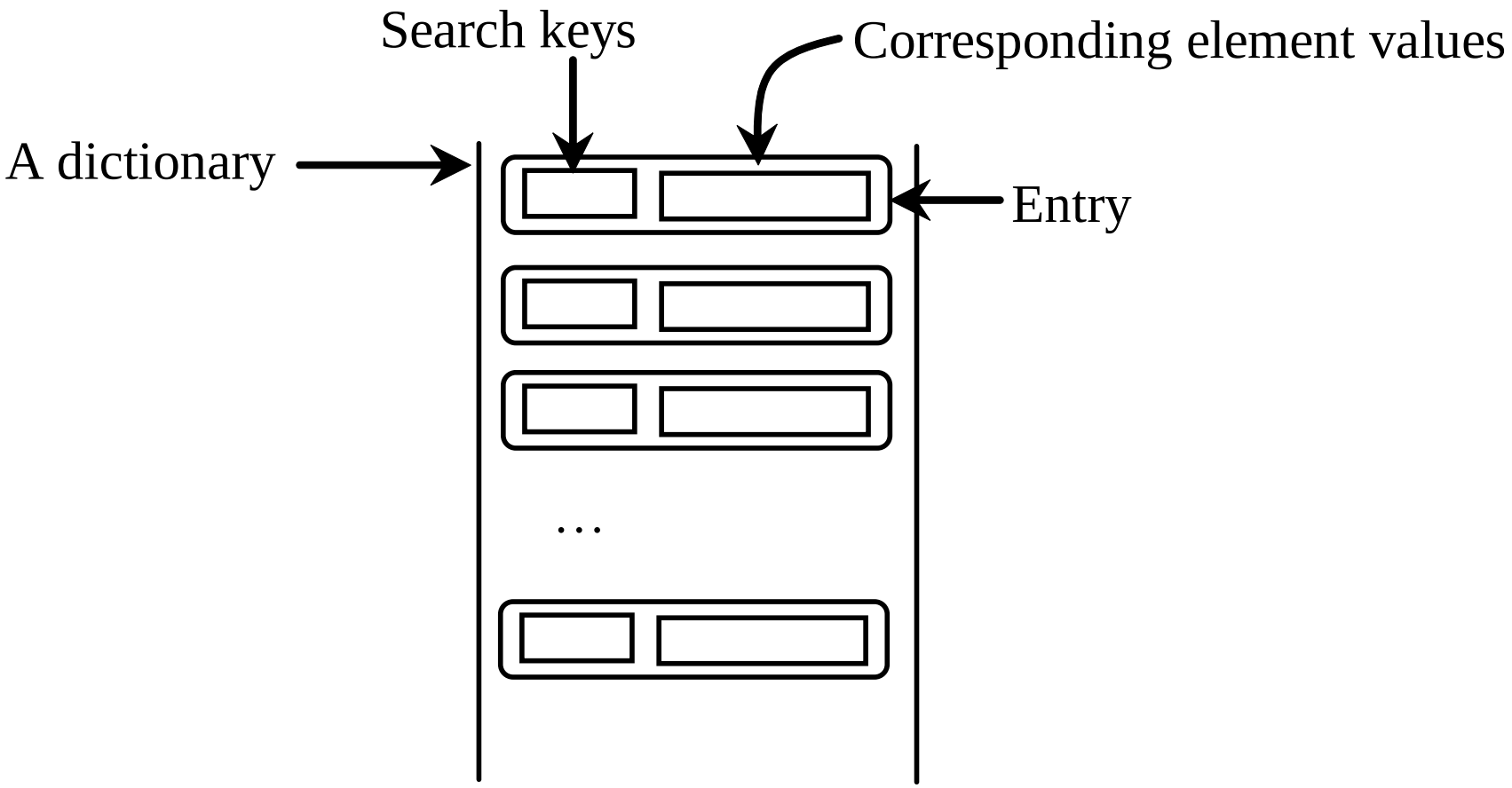
Objectives

- To store key/value pairs in a dictionary and access value using the key.
- To use dictionaries to develop applications.

Dictionary

- Why dictionary?
- Suppose your program stores a million students and frequently searches for a student using the social security number. An efficient data structure for this task is the *dictionary*. A dictionary is a collection that stores the elements along with the keys. The keys are like an indexer.

Key/value pairs



Creating a Dictionary

```
dictionary = {} # Create an empty dictionary
dictionary = {"john":40, "peter":45} # Create a dictionary
```

```
>>> a = dict(one=1, two=2, three=3)
>>> b = {'one': 1, 'two': 2, 'three': 3}
>>> c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
>>> d = dict([('two', 2), ('one', 1), ('three', 3)])
>>> e = dict({'three': 3, 'one': 1, 'two': 2})
>>> a == b == c == d == e
True
```

Adding/Modifying Entries

To add an entry to a dictionary, use

```
dictionary[key] = value
```

For example,

```
dictionary["susan"] = 50
```

Deleting Entries

To delete an entry from a dictionary, use

```
del dictionary[key]
```

For example,

```
del dictionary["susan"]
```

Looping Entries

```
for key in dictionary:
    print(key + ":" + str(dictionary[key]))
```

The len and in operators

len(dictionary) returns the number of the elements in the dictionary.

```
>>> dictionary = {"john":40, "peter":45}
>>> "john" in dictionary
True
>>> "johnson" in dictionary
False
```

The Dictionary Methods

dict	
keys(): tuple	Returns a sequence of keys.
values(): tuple	Returns a sequence of values.
items(): tuple	Returns a sequence of tuples (key, value).
clear(): void	Deletes all entries.
get(key): value	Returns the value for the key.
pop(key): value	Removes the entry for the key and returns its value.
popitem(): tuple	Returns a randomly-selected key/value pair as a tuple and removes the selected entry.

Other Built-in Dictionary Functions & Methods:

1. **cmp(dict1, dict2)** - Compares elements of both dict.
2. **len(dict)** - Gives the total length of the dictionary.
3. **str(dict)** - Produces a printable string representation of a dictionary
4. **type(variable)** - Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type.
5. **dict.copy()** - Returns a shallow copy of dictionary dict
6. **dict.fromkeys()** - Create a new dictionary with keys from seq and values set to value.
7. **dict.has_key(key)** - Returns true if key in dictionary dict, false otherwise
8. **dict.setdefault(key, default=None)** - Similar to get(), but will set dict[key]=default if key is not already in dict
9. **dict.update(dict2)** - Adds dictionary dict2's key-values pairs to dict

Built-in Dictionary Functions & Methods

Python includes the following dictionary functions:

1. cmp(dict1, dict2) Compares elements of both dict.
2. len(dict) Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.
3. str(dict) Produces a printable string representation of a dictionary
4. type(variable) Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type.

Python includes following dictionary methods

1. dict.clear() Removes all elements of dictionary dict
2. dict.copy() Returns a shallow copy of dictionary dict
3. dict.fromkeys() Create a new dictionary with keys from seq and values set to value.
4. dict.get(key, default=None) For key key, returns value or default if key not in dictionary
5. dict.has_key(key) Returns true if key in dictionary dict, false otherwise
6. dict.items() Returns a list of dict's (key, value) tuple pairs
7. dict.keys() Returns list of dictionary dict's keys
8. dict.setdefault(key, default=None) Similar to get(), but will set dict[key]=default if key is not already in dict
9. dict.update(dict2) Adds dictionary dict2's key-values pairs to dict
10. dict.values() Returns list of dictionary dict's values

Dictionaries

```
dict = {}
for i in range(ord('A'), ord('Z')+1):
    dict[i-65]= chr(i)

for i in range(len(dict)):
    print(dict[i])
```

Dictionary – Sample Code

```
import random

myDict = {}
MAX = 1000
def makeRandomDict():
    dict = {}
    for i in range(26):
        key = random.randrange(1, MAX)
        dict[key]= chr(i+65)
    return dict

def makeDict(dict):
    for i in range(26):
        dict[i]= chr(i+65)

def printDict(dict):
    keyList = list(dict)
    for i in keyList:
        print(dict[i])
```

```
def searchDict(searchKey, dict):
    keyList = list(dict)
    print(keyList)

    if(searchKey in keyList):
        idx = keyList.index(searchKey)
        print("FOUND key: ", searchKey,
                "Value: ", dict[searchKey])

def main():
    print(myDict)
    makeDict(myDict)
    printDict(myDict)
    searchDict(7, myDict)
    dic = makeRandomDict()
    printDict(dic)
    searchDict(628, dic)

main()
```

Case Studies: Occurrences of Words

This case study writes a program that counts the occurrences of words in a text file and displays the words and their occurrences in alphabetical order of words. The program uses a dictionary to store an entry consisting of a word and its count. For each word, check whether it is already a key in the dictionary. If not, add to the dictionary an entry with the word as the key and value 1. Otherwise, increase the value for the word (key) by 1 in the dictionary.