Introduction to Dictionaries

- At the end of this class, students will be able to-
  - Use the variable type – Dictionary
  - Create and modify Dictionaries using Dictionary built – in functions

What Is a Dictionary?

A **dictionary** is a mutable, associative data structure of variable length.

Python dictionary is an unordered collection of items in the form of keys and pairs.

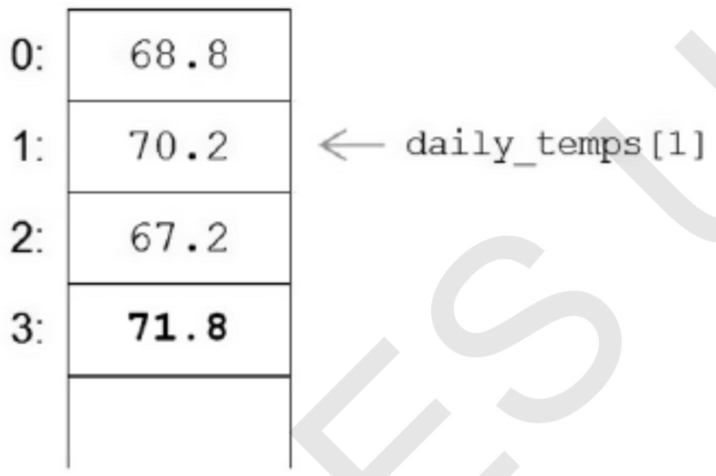
The syntax for declaring dictionaries in Python is given below.

```
daily_temps = {'sun': 68.8, 'mon': 70.2, 'tue': 67.2, 'wed': 71.8,
               'thur': 73.2, 'fri': 75.6, 'sat': 74.0}
```

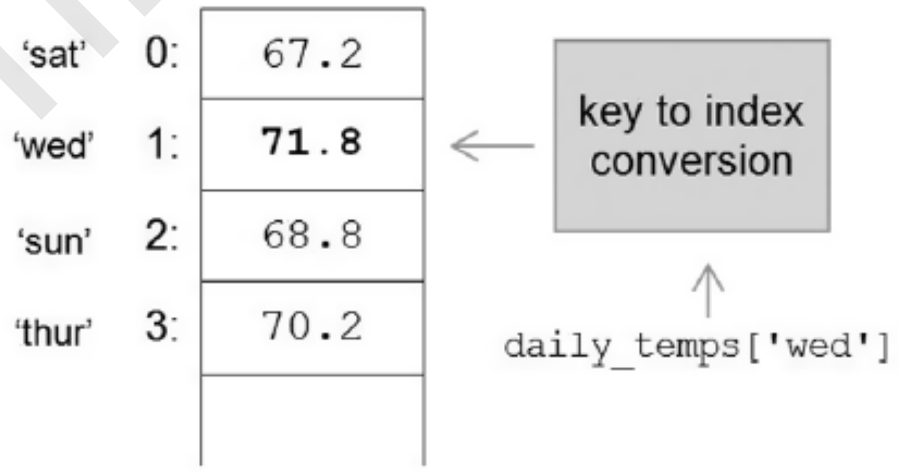
Dictionary daily_temps stores the average temperature for each day of the week, as we did earlier using a list. However, in this case, each temperature has associated with it a unique key value ('sun', 'mon', etc.). Strings are often used as key values.

Accessing the Elements of a Dictionary

The syntax for accessing an element of a dictionary is the same as for accessing elements of sequence types, except that a key value is used within the square brackets instead of an index value:



Indexed Data Structure



Associative Data Structure

The Hashing of Storage Locations

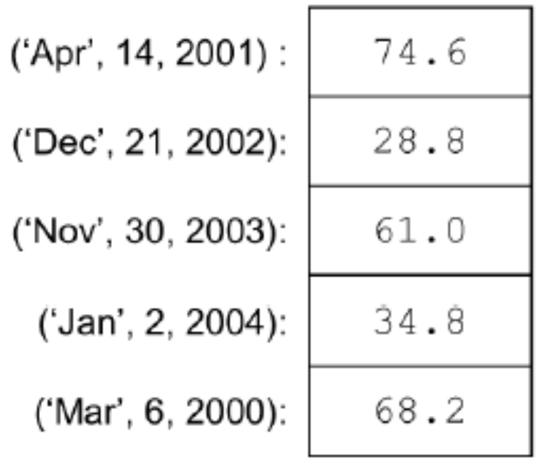
Although the elements of an associative data structure are physically ordered, the ordering is irrelevant to the way that the structure is utilized. The location that an element is stored in and retrieved from within an associative data structure depends only on its key value, thus there is no logical first element, second element, and so forth.

The specific location that a value is stored in is determined by a particular method of converting key values into index values called **hashing**.

Hash Value Types

Although strings are often used as key values, any immutable type may be used as a key value, such as a tuple. In this case, the temperature for a specific date is retrieved by,

```
temps = { ('Jan', 2,2004): 34.8,
          ('Mar', 6, 2000): 68.2,
          ('Nov', 30, 2003): 61.0,
          ('Apr', 14, 2001): 74.6,
          ('Dec', 21, 2002): 28.8}
```



Data Structure: dict : dictionary

- dict is a data structure which has key value pairs.
- Keys are unique
- keys are immutable – cannot be changed
- Keys are hashable
- Key value pairs are stored at the hashed location in some way
- dict like the set are unordered collection

- dict is indexable based on the key – key could be a string, an integer or a tuple or any immutable type
- An empty dict is created by using {} or by the constructor dict()
- we can extract keys using the method dict.keys() and the values using the method dict.values(). Both these return iterable objects. There will be one-to-one mapping between the keys in the dict.keys() and the values in the dict.values()
- we can iterate through a dict – we actually iterate through the keys.

Dynamic Creation and Modification of Dictionaries

Dictionaries may be created and modified dynamically. This is useful when,

- a dictionary contains too many values to hard code into a program, thus the key/value pairs can be read from a file and the dictionary “built” at runtime

- some of the values are not yet known, and therefore the dictionary must be updated at execution time

Creation of dictionary

#Empty Dictionary

a={}

#creating dict with keys and values

a={key1:value1,key2:value2…..}

Key should be immutable like number, string and tuple;Key should be unique

a={“ australia”:2015, “india”:2011}

>>> a["australia"]

2015

PES UNIVERSITY

Accessing Values in Dictionary

You can access the items of a dictionary by referring to its key name

```
a={"key1":"value1","key2":"value2","key3":"value3"}
print(a["key1"])
for i in a:
        print(i,"===>",a[i])
```

Output:

```
value1
key1 ===> value1
key2 ===> value2
key3 ===> value3
```

Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
items()	Returns a list containing the a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys

Dictionary Methods

Method	Description
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

clear() Method

Remove all elements from the dict:

```
team={'England': {'Australia', 'Pakistan'}, 'SriLanka':
  {'Netherlands'}, 'SouthAfrica': {'Australia', 'India',
  'WestIndies'}}
print(id(team))
team.clear()
print(team)
print(id(team))
```

Output:

```
2277301334664
{}
2277301334664
```

PES UNIVERSITY

copy() Method

Copy the dictionary:

```
team={'England': {'Australia', 'Pakistan'}, 'SriLanka':
  {'Netherlands'}}
print(id(team))
b=team.copy()
print(b)
print(id(team))
```

Output:

```
3171916558768
{'England': {'Pakistan', 'Australia'},
'SriLanka': {'Netherlands'}}
3171916558768
```

fromkeys() Method

The fromkeys() method creates a new dictionary from the given sequence of elements with a value provided by the user.

```
str123=("name","marks","section")
dict1=dict.fromkeys(str123,"value")
print(dict1
```

Output:

```
{'name': 'value', 'marks':
'value', 'section': 'value'}
```

fromkeys() Method

```
key=("name","marks","section")
dict1=dict.fromkeys(key,"value")
print(dict1)
dict1["name"]="xyz"
dict1["marks"]=45
dict1["section"]="B"
print(dict1)
```

Output:

```
{'name': 'value', 'marks': 'value', 'section': 'value'}
{'name': 'xyz', 'marks': 45, 'section': 'B'}
```

get() Method

The method **get()** returns a value for the given key. If key is not available then returns default value None.

```
dict1={'name': ‘xyz', 'marks': 45, 'section': 'B'}
a=dict1.get("name")
print(a)
print(dict1.get("hello"))
print(dict1["hello"])
```

Output:

```
xyz
None
KeyError: 'hello'
```

items() Method

The method **items()** returns a list of dict's (key, value) tuple pairs

```
dict1={'name': 'xyz', 'marks': 45, 'section': 'B'}
a=dict1.items()
print(a)
```

Output:

```
dict_items([('name', 'monika'), ('marks', 45), ('section', 'B')])
```

keys() Method

The method **keys()** returns a list of all the available keys in the dictionary.

```
dict1={'name': 'xyz', 'marks': 45, 'section': 'B'}
a=dict1.keys()
print(a)
```

Output:

```
dict_keys(['name', 'marks', 'section'])
```

pop() Method

The pop() method removes and returns an element from a dictionary having the given key.

```
dict1={'name': 'xyz', 'marks': 45, 'section': 'B'}
a=dict1.pop("name")
print(a)
print(dict1)
```

Output:

```
xyz
{'marks': 45, 'section': 'B'}
```

popitem() Method

The popitem() returns and removes an arbitrary element (key, value) pair from the dictionary.

```
dict1={'name': 'xyz', 'marks': 45, 'section': 'B'}
a=dict1.popitem()
print(a)
print(dict1)
```

Output:

```
('section', 'B')
{'name': 'xyz', 'marks': 45}
```

update() Method

The method **update()** adds dictionary dict2's key values pairs in to dict. This function does not return anything.

```
a={"python":["a",45]}
b={"python1":["b",46]}
a.update(b)
print(a)
```

Output:

```
{'python': ['a', 45], 'python1': ['b', 46]}
```

values() Method

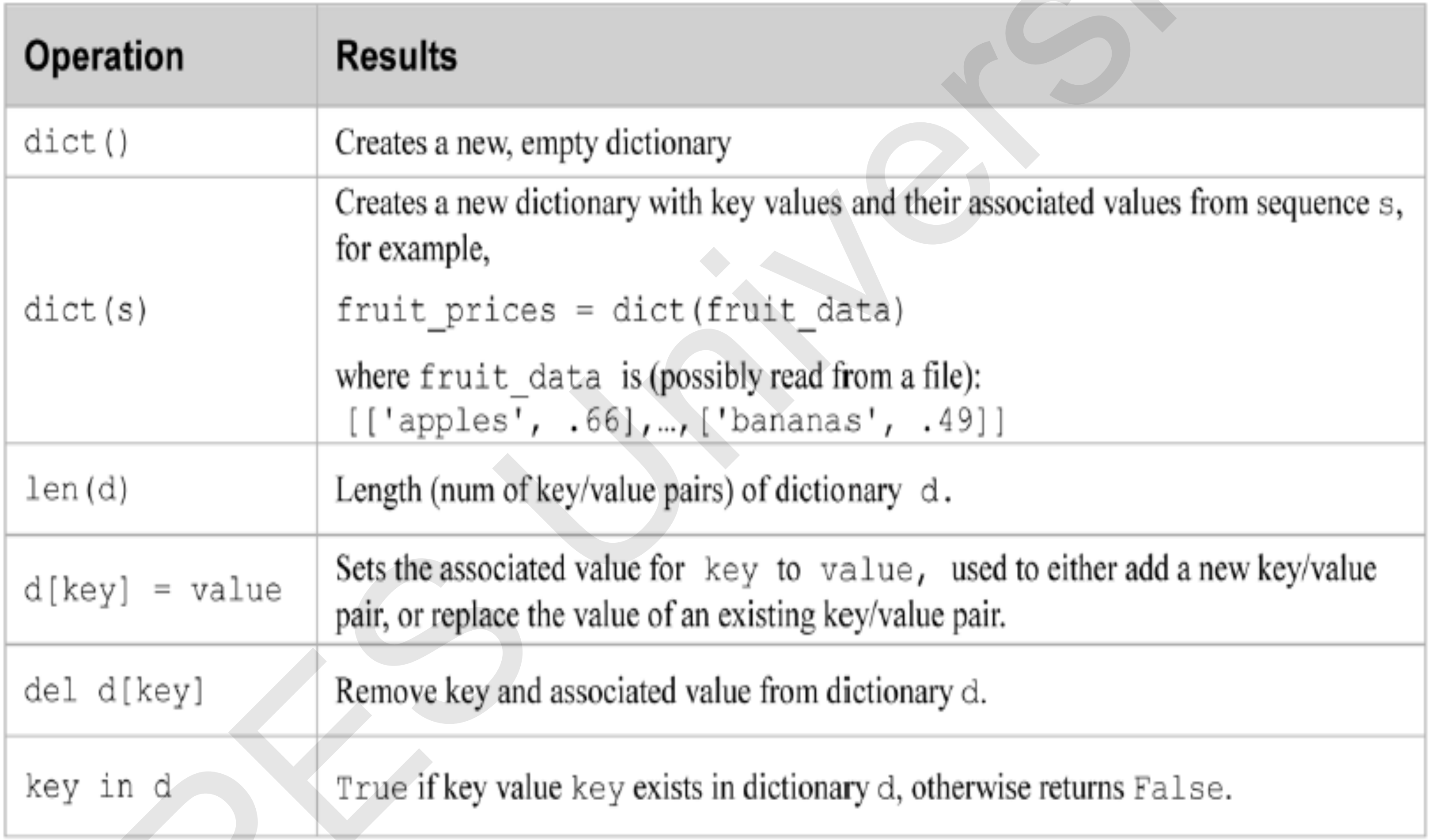
The method **values()** returns a list of all the values available in a given dictionary.

```
dict1={'name': 'xyz', 'marks': 45, 'section': 'B'}
print(dict1.values())
```

Output:

```
dict_values(['xyz', 45, 'B'])
```

Operations for Dynamically Manipulating Dictionaries



Operation	Results
`dict()`	Creates a new, empty dictionary

`dict(s)`	Creates a new dictionary with key values and their associated values from sequence `s`, for example,
	`fruit_prices = dict(fruit_data)`
	where `fruit_data` is (possibly read from a file):
	`[['apples', .66],…,['bananas', .49]]`

`len(d)`	Length (num of key/value pairs) of dictionary `d`.
`d[key] = value`	Sets the associated value for `key` to `value`, used to either add a new key/value pair, or replace the value of an existing key/value pair.
`del d[key]`	Remove key and associated value from dictionary `d`.
`key in d`	`True` if key value `key` exists in dictionary `d`, otherwise returns `False`.

Give a program segment that creates an initially empty dictionary named password_lookup,   prompting one-by-one for usernames and passwords (until a username of 'z' is read) entering each   into the dictionary.

```
password_lookup = {}
finished = False
while not finished:
        name = input('Enter name: ')
         if name.lower()!= 'z':
                passwd = input('Enter password: ')
                password_lookup[name] = passwd
        else:
                finished = True
print(password_lookup)
```

PES UNIVERSITY

Consider the following text for the problems below.

Example (Wining) (Losing) (score)

```
highest_inings="""England Australia 481
England Pakistan 444
SriLanka Netherlands 443
SouthAfrica WestIndies 439
SouthAfrica Australia 438
SouthAfrica India 438”””
```

- Consider the previous example and print the

Wining teams along with their opponent teams

Name and the count .

Example:

OP:

```
{'England': {'Pakistan', 'Australia'}, 'SriLanka': {'Netherlands'},
   'SouthAfrica': {'India', 'Australia', 'WestIndies'}}
England ====> 2
SriLanka ====> 1
SouthAfrica ====> 3
```

```
highest_inings="""England Australia 481
England Pakistan 444
SriLanka Netherlands 443
SouthAfrica WestIndies 439
SouthAfrica Australia 438
SouthAfrica India 438"""
team=set()
winning_team={}
for i in highest_inings.split("\n"):
        team=i.split()[0]
        losing_team=i.split()[1]
        if team not in winning_team:
                winning_team[team]=set()
        winning_team[team].add(losing_team)
print(winning_team)
for i in winning_team:
        print(i,"===>",len(winning_team[i]))
```

Consider the text below:

CBT1="""BSection Nikhil Python 14

ASection Aryan Python 25

BSection Aparna Chemistry 28

ASection Shurthi Chemistry 32"""“

Find the number of Sections and print?

```
CBT1="""BSection Nikhil Python 14
ASection Aryan Python 25
BSection Aparna Chemistry 28
ASection Shurthi Chemistry 32"""
# find the no of Sections
a=CBT1.split("\n")
c=set()
for i in a:
        c.add(i.split()[0])
print(c,"no of sections is",len(c))

for l in enumerate(CBT1.split('\n')) :
        print(l)
```

Output:

```
{'ASection', 'BSection'} no of sections
is 2
(0, 'BSection Nikhil Python 14')
(1, 'ASection Aryan Python 25')
(2, 'BSection Aparna Chemistry 28')
(3, 'ASection Shurthi Chemistry 32')
```

```
all = "“” sanskrit kalidasa shakuntala
english r_k_narayan malgudi_days
kannada kuvempu ramayanadarshanam
sanskrit bhasa swapnavasavadatta
kannada kuvempu malegalalli_madumagalu
english r_k_narayan dateless_diary
kannada karanta chomanadudi
sanskrit baana harshacharita
kannada karanta sarasatammana_Samadhi
sanskrit kalidasa malavikagnimitra
sanskrit kalidasa raghuvamsha
sanskrit baana kadambari
sanskrit bhasa pratijnayogandhararayana”””
```

# find the no of books

```
print(“no of books : ", len(all.split('\n')))
```

# find the number of languages

```
langset = set()
for line in all.split('\n'):
    #print(line.split()[0])
    langset.add(line.split()[0])
#print(langset)
print(“no  of lang : ", len(langset))
```

Output:

```
no of books :  13
no  of lang :  3
```

PES UNIVERSITY

```
# count the number of books in each language
lang_book_count = {}
for line in all.split('\n'):
    lang = line.split()[0]
    #print(lang)
    if lang not in lang_book_count :
            lang_book_count[lang] = 0
    lang_book_count[lang] += 1

for lang in lang_book_count :
    print(lang, " => ",  lang_book_count[lang])
```

Output:

```
sanskrit  =>  7
english  =>  2
kannada  =>  4
```

```
# find list of authors for each language
# dict of sets
lang_author = {}
for line in all.split('\n'):
   (lang, author) = line.split()[:2]
# print(lang, author)
   if lang not in lang_author:
         lang_author[lang] = set()
   lang_author[lang].add(author)

for lang in lang_author:
   print(lang)
   for author in lang_author[lang]:
         print("\t", author)
```

Output:

```
sanskrit
        bhasa
        kalidasa
        baana
english
        r_k_narayan
kannada
        karanta
        kuvempu
```

```
# find # of books of each author in each language
# soln: dict of dict of int

lang_author = {}
for line in all.split('\n'):
    (lang, author) = line.split()[:2]
    if lang not in lang_author :
            lang_author[lang] = {}
    if author not in lang_author[lang] :
            lang_author[lang][author] = 0
    lang_author[lang][author] += 1

for lang in lang_author:
    print(lang)
    for author in lang_author[lang]:
            print("\t", author, "=>",
                            lang_author[lang][author])
```

Output:

```
sanskrit
        kalidasa => 3
        bhasa => 2
        baana => 2
english
        r_k_narayan => 2
kannada
        kuvempu => 2
        karanta => 2
```

# find list of titles of each author of each lang

# soln: dict of dict of list

```
info = {}
for line in all.split('\n'):
   (lang, author, title) = line.split()
   if lang not in info :
            info[lang] = {}
   if author not in info[lang] :
            info[lang][author] = []
   info[lang][author].append(title)

for lang in info:
   print(lang)
   for author in info[lang]:
            print("\t", author)
            for title in info[lang][author]:
                        print("\t\t", title)
```

Output:

```
sanskrit
        kalidasa
                shakuntala
                malavikagnimitra
                raghuvamsha
        bhasa
                swapnavasavadatta
                pratijnayogandhararayana
        baana
                harshacharita
                kadambari
english
        r_k_narayan
                malgudi_days
                dateless_diary
kannada
        kuvempu
                ramayanadarshanam
                malegalalli_madumagalu
        karanta
                chomanadudi
                sarasatammana_Samadhi
```

Dictionary Comprehension

```
squares = {x: x*x for x in range(6)}
print(squares)
```

Output:

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Summary

- A **dictionary** is a mutable, associative data structure of variable length.
- Python dictionary is an unordered collection of items in the form of keys and pairs