# Dictionaries in Python

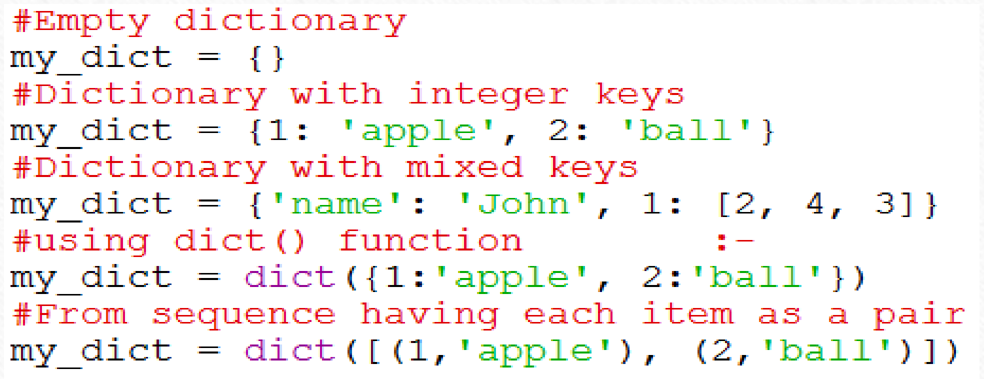
# Definition of  the Dictionary

- Dictionary is a data structure in which we store values as a pair of key and value.
- Each key is separated from its value by a colon(:) and consecutive items are separated by commas.
- Dictionary is an unordered collection of key-value pairs and it is generally used when we have a huge amount of data.
- The entire items in a dictionary are enclosed in curly brackets { }.

  **Syntax : -** dict_name={ key1:val1, key2:val2 ,key3:val3 }
- Dictionary keys are case- sensitive .Two keys with same name but in different case are not the same in python.

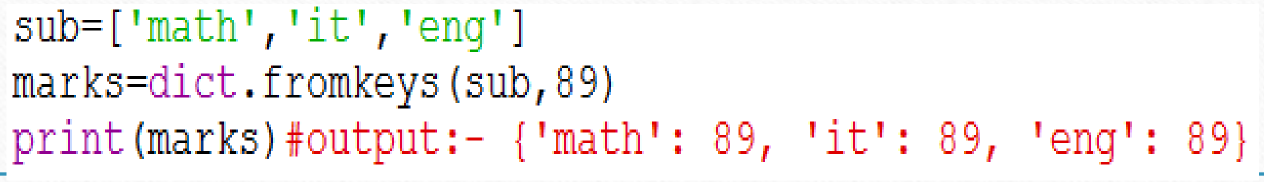
# Creating a Dictionary

- Creating a dictionary is as simple as placing items inside curly braces {} separated by comma.
- An item has a key and the corresponding value expressed as a pair, key: value.
- While values can be of any data type and can repeat, keys must be of immutable type (string, number or tuple with immutable elements) and must be unique.



```
#Empty dictionary
my_dict = {}
#Dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}
#Dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}
#using dict() function        :-
my_dict = dict({1:'apple', 2:'ball'})
#From sequence having each item as a pair
my_dict = dict([(1,'apple'), (2,'ball')])
```

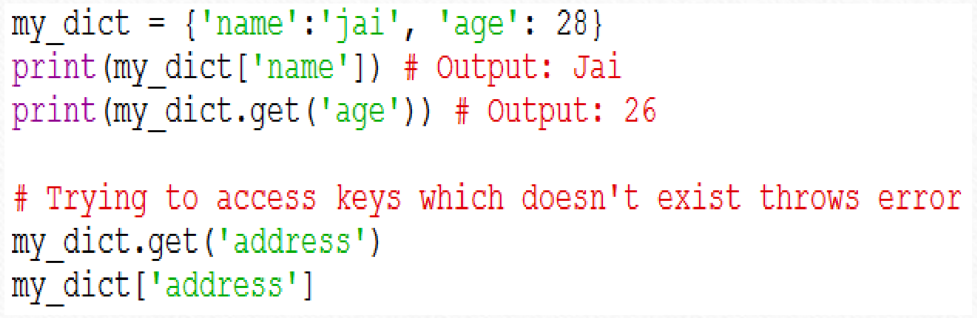
- Dict.fromkeys(seq[,val]):- Create a new dictionary with keys from seq and values set  to val. If no val is specified then, none is assigned as default value.



```
sub=['math','it','eng']
marks=dict.fromkeys(sub,89)
print(marks)#output:- {'math': 89, 'it': 89, 'eng': 89}
```

# Access Elements from a Dictionary

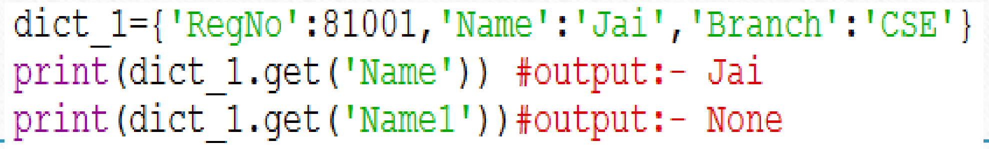
- To access values in dictionary, get() Method  or square brackets  are used along with key to obtain its value.
- Note that if  you try to access an item with key ,which is not specified in the dictionary , a keyError is generated .
- The difference while using get() is that it returns None instead of  KeyError, if  the key is not found.



```
my_dict = {'name':'jai', 'age': 28}
print(my_dict['name']) # Output: Jai
print(my_dict.get('age')) # Output: 26

# Trying to access keys which doesn't exist throws error
my_dict.get('address')
my_dict['address']
```

Dict.get(key,[value]):- Returns the value for the key passed as argument. If  the key is not present in dictionary. It will return the default value is specified then  it will return none.

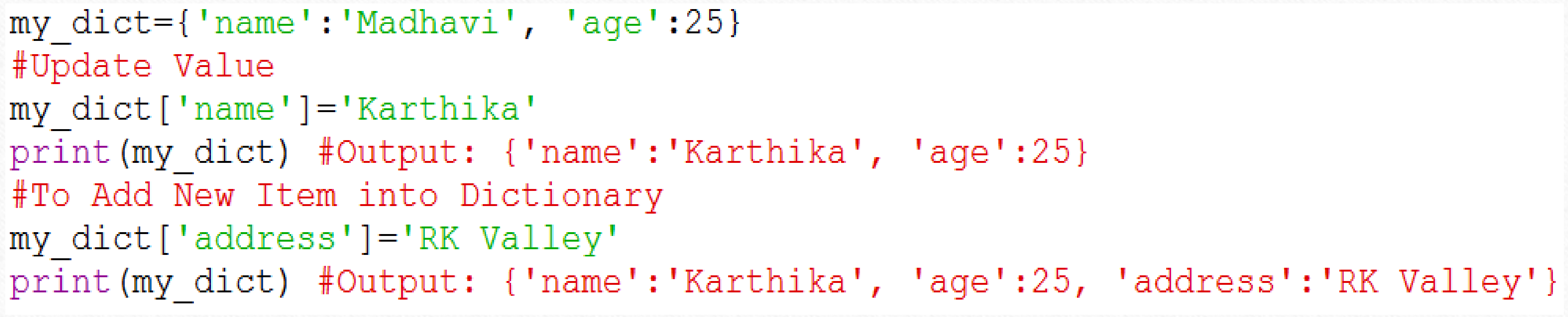


```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
print(dict_1.get('Name')) #output:- Jai
print(dict_1.get('Name1'))#output:- None
```

# Adding and Modifying an Item in a Dictionary

- Dictionaries are mutable. We can add new items or change the value of existing items using assignment operator.

- If the key is already present, value gets updated, else a new key: value pair is added to the dictionary.

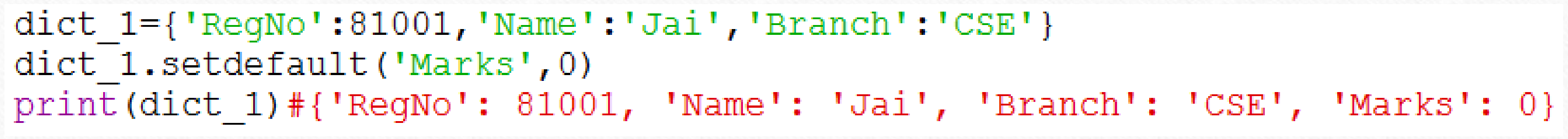


```
my_dict={'name':'Madhavi', 'age':25}
#Update Value
my_dict['name']='Karthika'
print(my_dict) #Output: {'name':'Karthika', 'age':25}
#To Add New Item into Dictionary
my_dict['address']='RK Valley'
print(my_dict) #Output: {'name':'Karthika', 'age':25, 'address':'RK Valley'}
```

- Dictionary is an associative array also known as hashes since any key of the dictionary can be associated or mapped to a value.

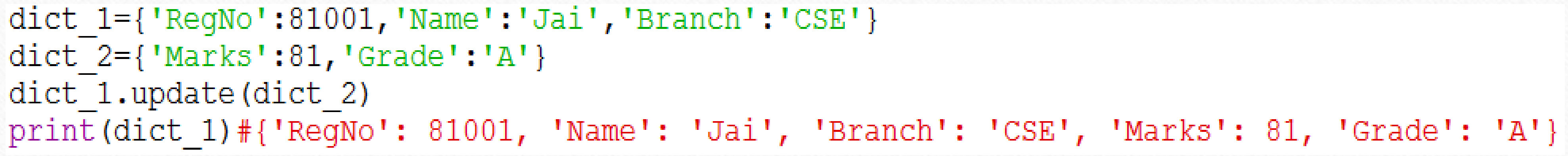
# Adding and Modifying an Item in a Dictionary

Dict.setdefault(key[,value]): - Sets a default value for a key that is not present in the Dict



```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
dict_1.setdefault('Marks',0)
print(dict_1)#{'RegNo': 81001, 'Name': 'Jai', 'Branch': 'CSE', 'Marks': 0}
```

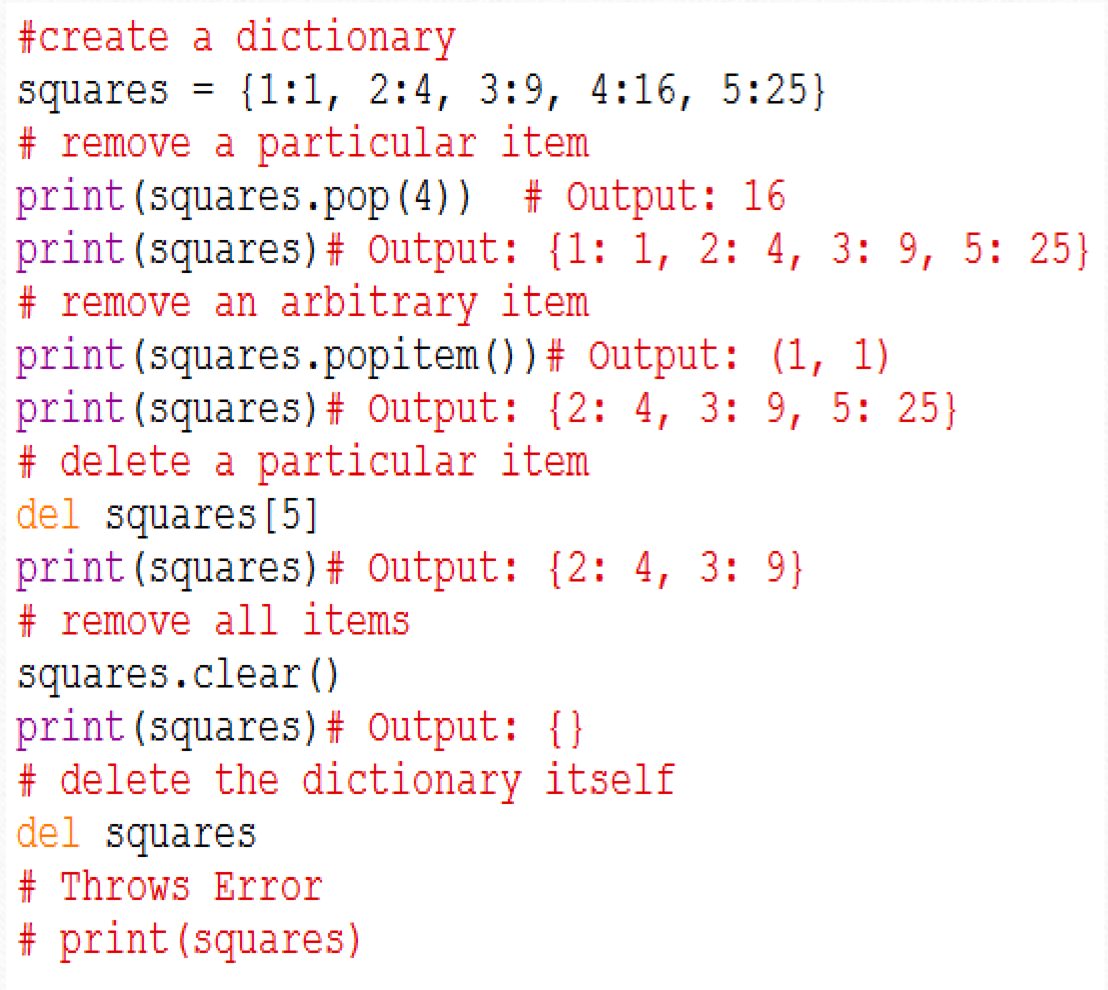
Dict1.update(Dict2):- Adds the key-value pairs of Dict2 to the key-value pairs of Dict1



```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
dict_2={'Marks':81,'Grade':'A'}
dict_1.update(dict_2)
print(dict_1)#{'RegNo': 81001, 'Name': 'Jai', 'Branch': 'CSE', 'Marks': 81, 'Grade': 'A'}
```

# Deleting or removing an Item in a Dictionary

- We can remove a particular item in a dictionary by using the method **pop()**. This method removes as item with the provided key and returns the value.
- The method, **popitem()** can  also be used to remove and return an arbitrary **item (key, value)** form the dictionary.
- All the items can be removed at once using the **clear()** method.
- We can also use the del keyword to remove individual items or the entire dictionary itself.

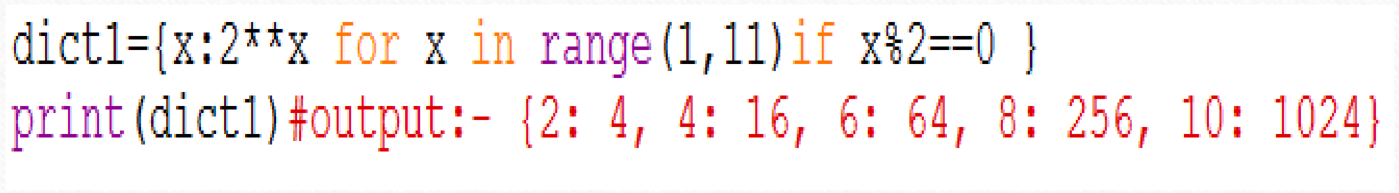


```
#create a dictionary
squares = {1:1, 2:4, 3:9, 4:16, 5:25}
# remove a particular item
print(squares.pop(4))  # Output: 16
print(squares)# Output: {1: 1, 2: 4, 3: 9, 5: 25}
# remove an arbitrary item
print(squares.popitem())# Output: (1, 1)
print(squares)# Output: {2: 4, 3: 9, 5: 25}
# delete a particular item
del squares[5]
print(squares)# Output: {2: 4, 3: 9}
# remove all items
squares.clear()
print(squares)# Output: {}
# delete the dictionary itself
del squares
# Throws Error
# print(squares)
```

# Dictionary Comprehension

- Dictionary comprehension is another way of Creating a dictionary.
- Dictionary comprehension a syntactic construct which creates a dictionary based on existing dictionary.

  Syntax: - D= {expression for  variable in sequence [if condition]}
- The Dictionary comprehension  is placed within curly brackets .
- It has three parts – **for loop , condition** and **expression .**
- First ,the for loop is used to go through  the sequence .
- The if condition is optional and if specified , only those values in the sequence are evaluated using the expression which satisfy the condition.



```
dict1={x:2**x for x in range(1,11)if x%2==0 }
print(dict1)#output:- {2: 4, 4: 16, 6: 64, 8: 256, 10: 1024}
```

# Key points to Remember

- Keys must have unique values, not even a single key can be duplicated in a dictionary.
- If you try to add a duplicate key ,then the last assignment is retained.

```
my_dict={'name': 'Karthika', 'age': 25, 'name':'Madhavi'}
print(my_dict['name']) # Output : Madhavi
```

- In dictionary, keys should be strictly of a type that is immutable, this means that a key can be of strings ,number, or tuple type but it cannot be a list which is mutable.

```
my_dict={[1,2]: 'Karthika', 'age': 25, 'name':'Madhavi'}
print(my_dict) # TypeError: unhashable type: 'list'
```

- In case you try to make your keys of a mutable type then a typeError will generated.
- The **in** or **not in** keyword can be used to check whether a single key is present in the dictionary(**Membership Operator**).

```
my_dict = {'name':'jai', 'age': 28'}
if 'name' in my_dict: print(my_dict['name'])#output: jai
```

## Sorting and Looping Items in a Dictionary

- The **keys()** method of dictionary returns a list of all the keys used in the dictionary in an arbitrary order and the **sorted()** function is used to sort the keys

```
my_dict={'name': 'Karthika', 'age': 25, 'address':'RK Valley'}
print(sorted(my_dict.keys())) #['address', 'age', 'name']
```

- We can access only values ,only keys and both using with **for loop**

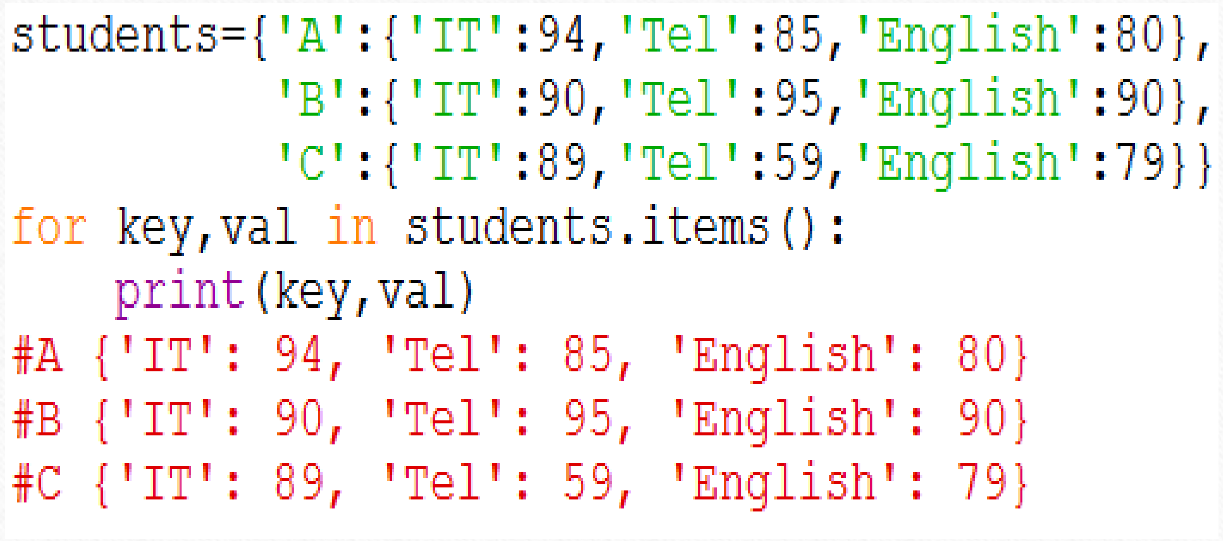
```
my_dict={'name': 'Karthika', 'age': 25, 'address':'RK Valley'}
for key in my_dict:
    print(key,end=" ") #Accessing keys(Output: name age address)
print()
for val in my_dict.values():
    print(val,end=" ") #Accessing values(Output: Karthika 25 RK Valley)
print()
for key,val in my_dict.items():
    print(key,val,end=" ") #Accessing keys and values
                          #(Output: name Karthika age 25 address RK Valley)
```

- Here **values()** method returns only values and **items()** method returns both keys and values in a dictionary

## Nested Dictionaries

- We can also create nested dictionary like nested lists or nested tuples.

- To define a dictionary inside another dictionary.



```
students={'A':{'IT':94,'Tel':85,'English':80},
          'B':{'IT':90,'Tel':95,'English':90},
          'C':{'IT':89,'Tel':59,'English':79}}
for key,val in students.items():
    print(key,val)
#A {'IT': 94, 'Tel': 85, 'English': 80}
#B {'IT': 90, 'Tel': 95, 'English': 90}
#C {'IT': 89, 'Tel': 59, 'English': 79}
```

## Built-in Dictionary Functions and Methods

Len(Dict) :- Returns the length of dictionary ,that is number of items(key-value pairs)

```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
print(len(dict_1))#output:- 3
```

str(Dict) :- Returns a string representation of the dictionary

```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
print(str(dict_1))#{'RegNo': 81001, 'Name': 'Jai', 'Branch': 'CSE'}
```

Dict.clear() :- Delete all entries in the dictionary

```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
dict_1.clear()
print(dict_1) #output:- {}
```

Dict.copy() :- Returns a shallow copy of the dictionary, i.e., the dictionary returned will not have a duplicate copy of Dict but will have the same reference.

```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
dict_2=dict_1.copy()
print(dict_2) #Output:{'RegNo': 81001, 'Name': 'Jai', 'Branch': 'CSE'}
dict_2['Name']='Abhiram'
print(dict_1) #Output:{'RegNo': 81001, 'Name': 'Jai', 'Branch': 'CSE'}
print(dict_2) #Output:{'RegNo': 81001, 'Name': 'Abhiram', 'Branch': 'CSE'}
```

Dict.fromkeys(seq[,val]):- Create a new dictionary with keys from seq and values set to val. If no val is specified then, none is assigned as default value.

```
sub=['math','it','eng']
marks=dict.fromkeys(sub,89)
print(marks)#output:- {'math': 89, 'it': 89, 'eng': 89}
```

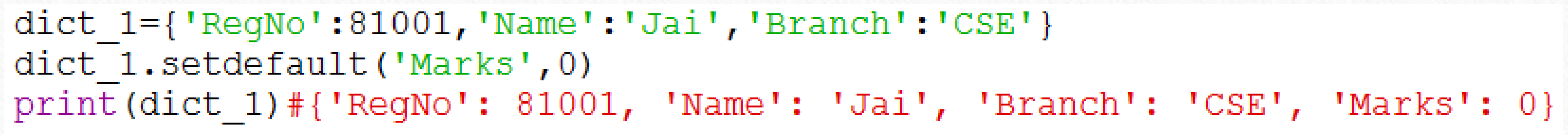
Dict.get(key):- Returns the value for the key passed as argument. If the key is not present in dictionary. It will return the default value is specified then it will return none.

```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
print(dict_1.get('Name')) #output:- Jai
print(dict_1.get('Name1'))#output:- None
```

Dict.items():- Returns a list of tuples (key-value pair),keys() returns list of keys and values() returns list of values in the dictionary,

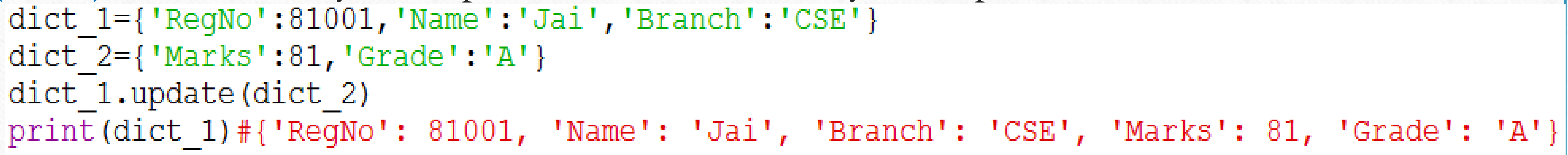
```
print(dict_1.items())#dict_items([('RegNo', 81001), ('Name', 'Jai'), ('Branch', 'CSE')])
print(dict_1.keys())#dict_keys(['RegNo', 'Name', 'Branch'])
print(dict_1.values())#dict_values([81001, 'Jai', 'CSE'])
```

Dict.setdefault(key,value): - Sets a default value for a key that is not present in the Dict



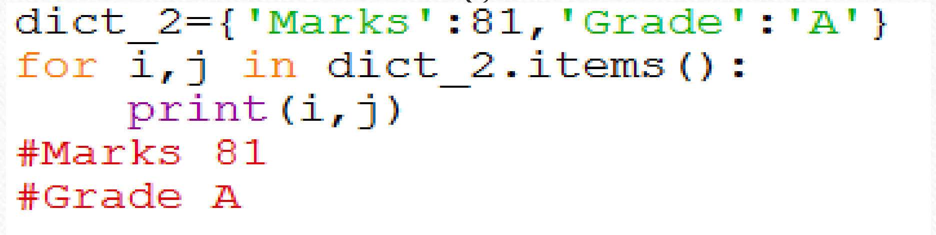
```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
dict_1.setdefault('Marks',0)
print(dict_1)#{'RegNo': 81001, 'Name': 'Jai', 'Branch': 'CSE', 'Marks': 0}
```

Dict1.update(Dict2):- Adds the key-value pairs of Dict2 to the key-value pairs of Dict1



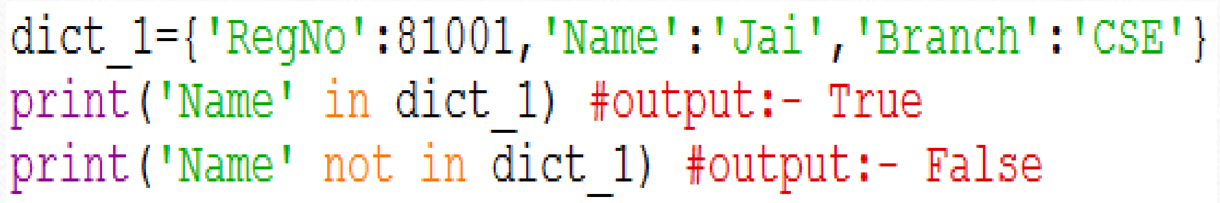
```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
dict_2={'Marks':81,'Grade':'A'}
dict_1.update(dict_2)
print(dict_1)#{'RegNo': 81001, 'Name': 'Jai', 'Branch': 'CSE', 'Marks': 81, 'Grade': 'A'}
```

Dict.iteritems():- Used to iterate through items in the dictionary.



```
dict_2={'Marks':81,'Grade':'A'}
for i,j in dict_2.items():
    print(i,j)
#Marks 81
#Grade A
```

in and not in :- Check whether a given key is present in dictionary or not



```
dict_1={'RegNo':81001,'Name':'Jai','Branch':'CSE'}
print('Name' in dict_1) #output:- True
print('Name' not in dict_1) #output:- False
```

All(): -  Return True if all keys of the dictionary are true  or if the dictionary is empty

Any():- Return True if all and any  key of the dictionary is true or if the dictionary  is empty it returns False

## Difference between List and Dictionary

- A list is an ordered set of items, But a dictionary is a data structure that is used for matching one item(key) with another(value).
- In lists, you can use indexing to access a particular item, But these indexes should be a number.
- In dictionaries, you can use any type (immutable) of value as an index.

  For Example, here when we write **Dict['Name']**, Name acts as an index but it is not a number but a string.
- Lists are used to look up a value whereas a dictionary is used take one value and loop up another value . For this reason , dictionary is also known as a lookup table.

## Advantages of a Dictionary

- In fact , the main advantage of a dictionary is that you don't need to search for a value one by one in the entire set of values, you can find a value instantly.
- The key-value pair may not be displayed in the order in which it was specified while defining the dictionary.
- This is because python uses complex algorithms(called hashing) to provide fast access to the items stored in the dictionary.
- This also makes dictionary preferable to use over a list of tuples.

# String Formatting with Dictionary

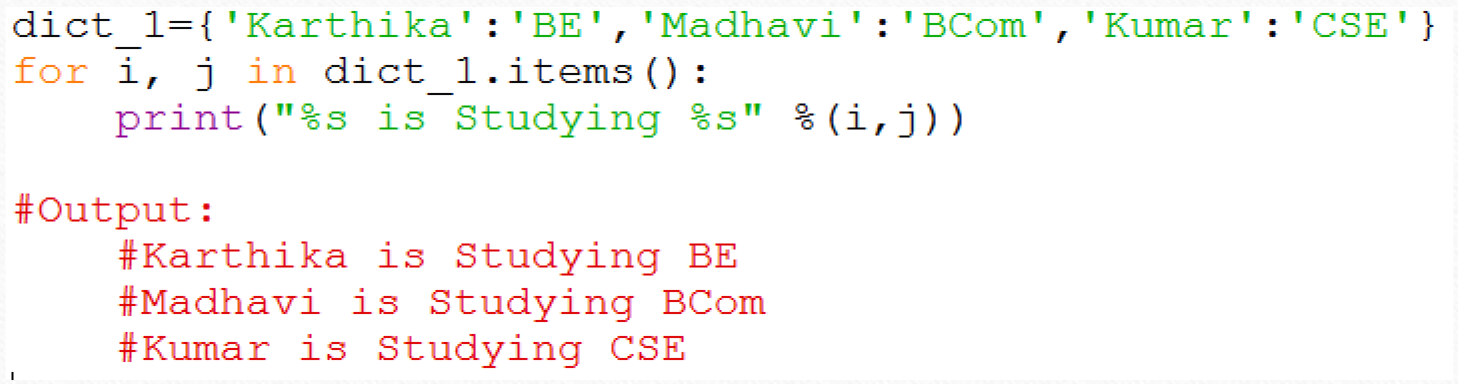
- Python also allow you to use string formatting feature with dictionaries.

- So you can use the following representation  and etc..

%s → Strings

%d→ Integer

%f→ Float point numbers



```
dict_1={'Karthika':'BE','Madhavi':'BCom','Kumar':'CSE'}
for i, j in dict_1.items():
    print("%s is Studying %s" %(i,j))

#Output:
    #Karthika is Studying BE
    #Madhavi is Studying BCom
    #Kumar is Studying CSE
```

## When to use which Data Structure

- Use lists to store a collection of data that does not need random access.
- Use lists if the data has to be modified frequently.
- Use a set if you want to ensure that every element in the data structure must be unique.
- Use tuples when you want that your data should not be altered
- Use dictionary to accessing very fast to the items stored in the dictionary.
- Dictionaries are the best data structure  for frequent lookup operations.

## Practice Problems on Dictionaries

1. Write a program that combines the lists to a dictionary?
2. Write a program to count the numbers of characters in the given string and store them in a dictionary data structure?
3. Write a program that print a histogram of frequencies of characters occurring in message?
4. Write a program that has dictionary of names of students and a list of their marks in 4 subjects. Create another dictionary from this dictionary that has name of the students and their total marks .Find out the topper and his/her score?
5. Write a program that inverts a dictionary .That is , it makes key of one dictionary value of another and vice versa?
6. Write a program that creates a dictionary of cubes of odd numbers in the range of **n**?

7. Write a program that calculates fib(n) using dictionary?
8. Write a program that has set of even numbers and their corresponding squares, to define another dictionary that has a list of even numbers and their corresponding cubes and take all values from dict_1 and dict_2 and display squares and cube values ?
9. Write a program that creates two dictionaries one that stores conversion values from meters to centimeters and the other that stores values from centimeters to meters?
10. Write a program that creates a dictionary of radius of a circle and its circumference using break statements ?