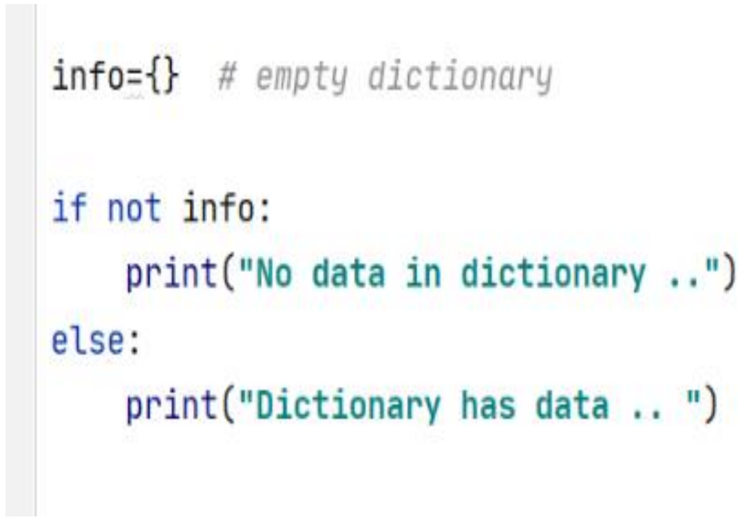
Dictionary

Reference: w3schools.com

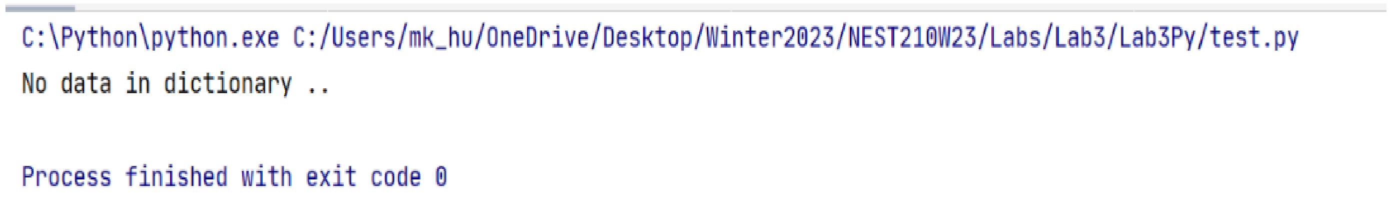
Topics Covered – Dictionaries and Files

- Creating and traversing dictionary items
- Adding items (key:value) in dictionary
- Removing items (key:value) in dictionary (using pop() and del)
- Clear entire Dictionary using clear() Dictionary API
- Copy Dictionary to another Dictionary (using copy() and dict() constructor)
- Nested Dictionary
- Files
- Modes of opening file
- File write and read operation
- File functions

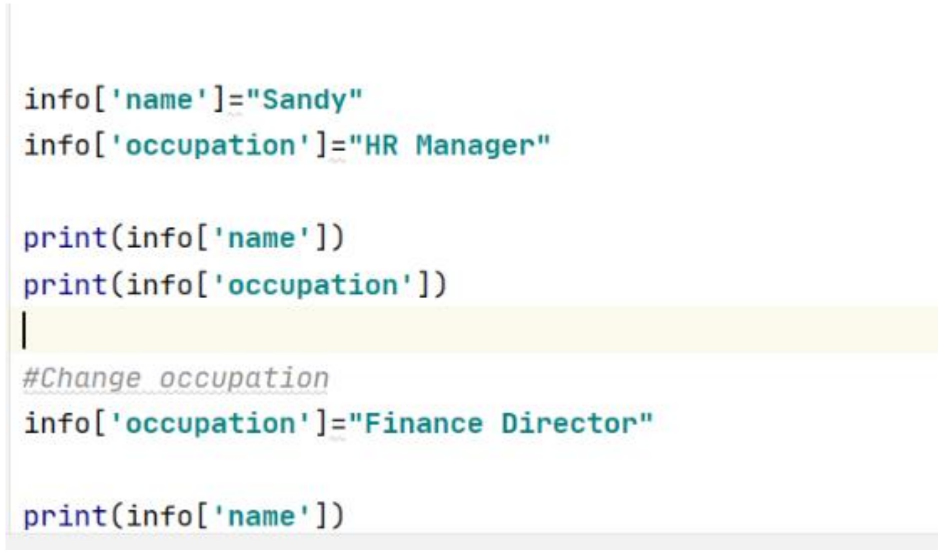
Create Dictionary – and add ‘key’ and ‘elements’



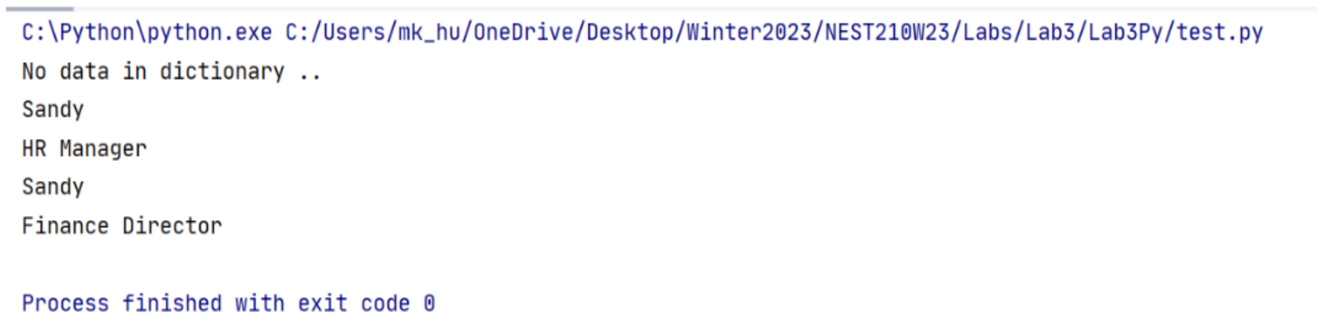
```
info={}   # empty dictionary

if not info:
    print("No data in dictionary ..")
else:
    print("Dictionary has data .. ")
```



```
C:\Python\python.exe C:/Users/mk_hu/OneDrive/Desktop/Winter2023/NEST210W23/Labs/Lab3/Lab3Py/test.py
No data in dictionary ..

Process finished with exit code 0
```



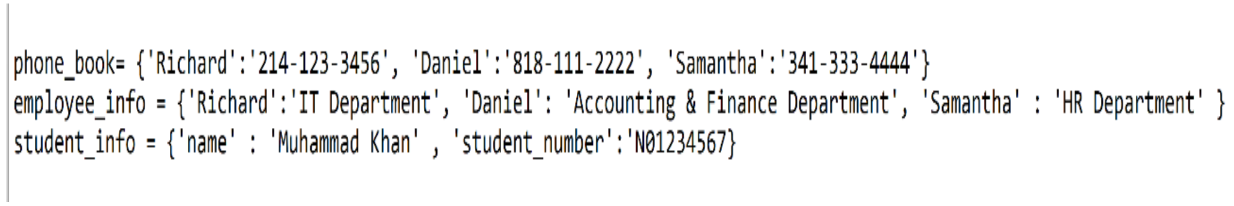
```
info['name']="Sandy"
info['occupation']="HR Manager"

print(info['name'])
print(info['occupation'])

#Change occupation
info['occupation']="Finance Director"

print(info['name'])
```



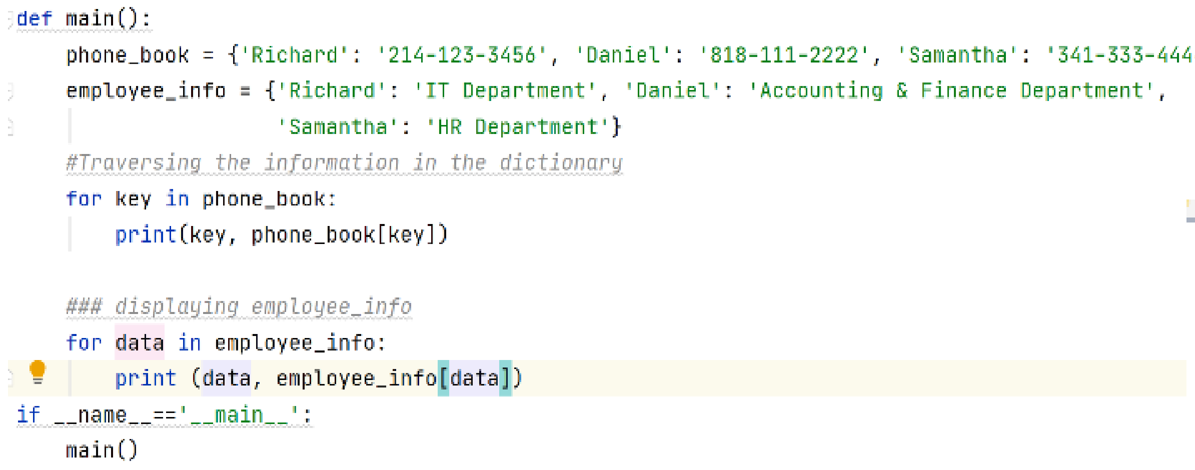
```
C:\Python\python.exe C:/Users/mk_hu/OneDrive/Desktop/Winter2023/NEST210W23/Labs/Lab3/Lab3Py/test.py
No data in dictionary ..
Sandy
HR Manager
Sandy
Finance Director

Process finished with exit code 0
```

Format of Dictionary data type

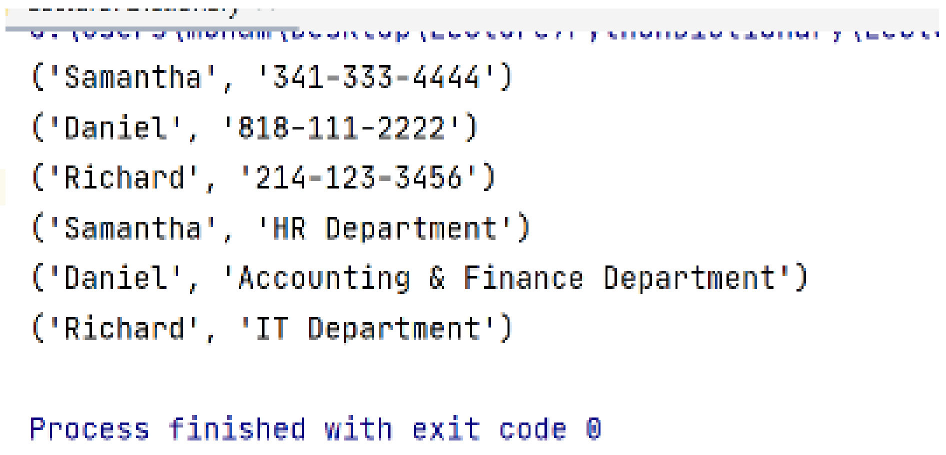
w3schools.com



```
phone_book= {'Richard':'214-123-3456', 'Daniel':'818-111-2222', 'Samantha':'341-333-4444'}
employee_info = {'Richard':'IT Department', 'Daniel': 'Accounting & Finance Department', 'Samantha' : 'HR Department' }
student_info = {'name' : 'Muhammad Khan' , 'student_number':'N01234567}
```

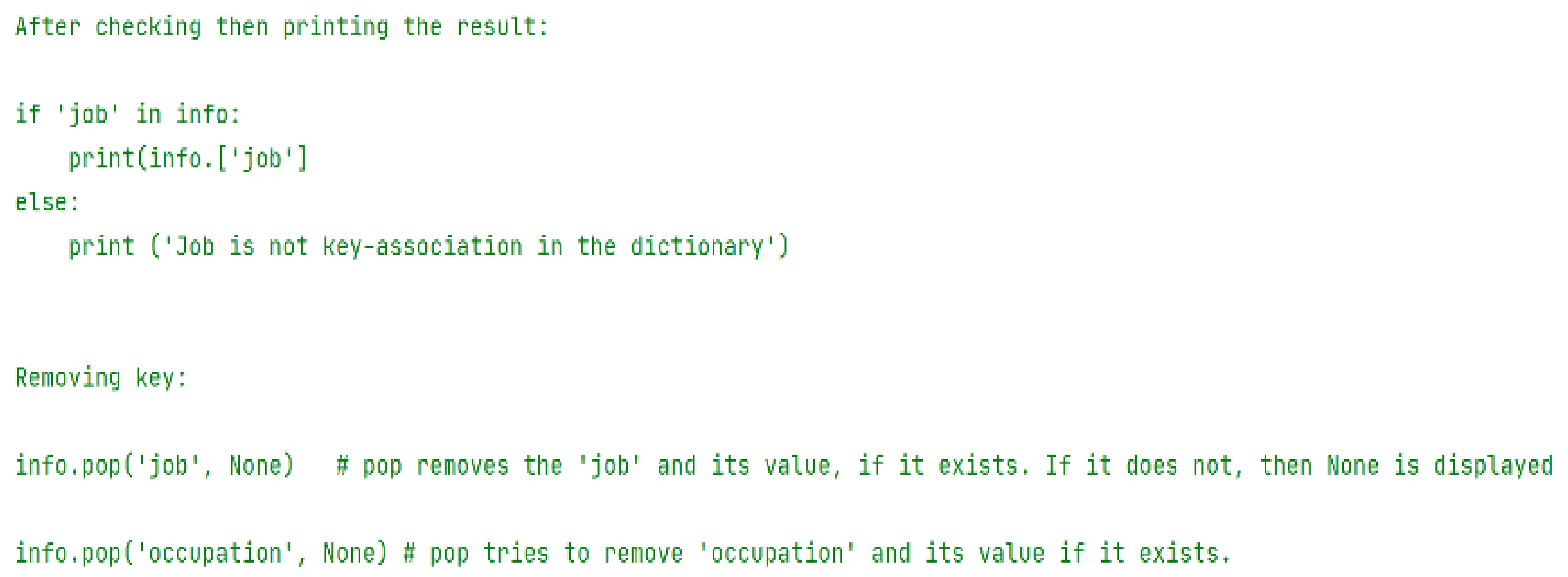


```python
def main():
    phone_book = {'Richard': '214-123-3456', 'Daniel': '818-111-2222', 'Samantha': '341-333-4444'}
    employee_info = {'Richard': 'IT Department', 'Daniel': 'Accounting & Finance Department',
                     'Samantha': 'HR Department'}
    #Traversing the information in the dictionary
    for key in phone_book:
        print(key, phone_book[key])

    ### displaying employee_info
    for data in employee_info:
        print (data, employee_info[data])
if __name__=='__main__':
    main()
```



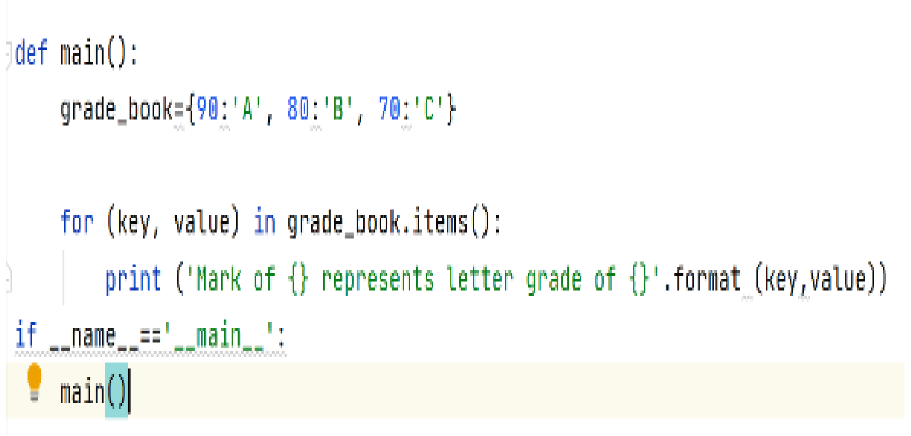
```
('Samantha', '341-333-4444')
('Daniel', '818-111-2222')
('Richard', '214-123-3456')
('Samantha', 'HR Department')
('Daniel', 'Accounting & Finance Department')
('Richard', 'IT Department')

Process finished with exit code 0
```

Checking ‘key’ in a dictionary

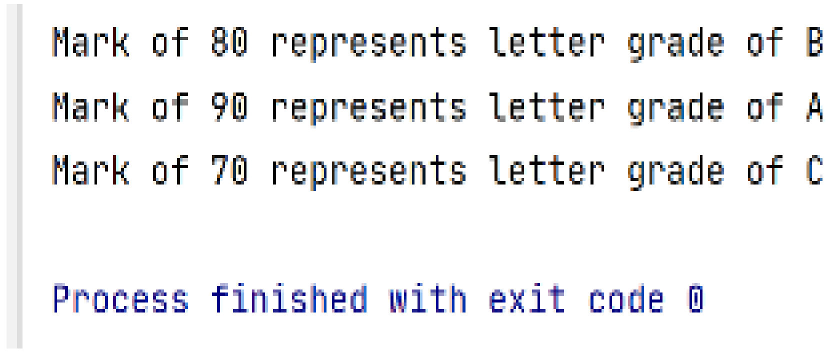


```
After checking then printing the result:

if 'job' in info:
    print(info.['job']
else:
    print ('Job is not key-association in the dictionary')


Removing key:

info.pop('job', None)   # pop removes the 'job' and its value, if it exists. If it does not, then None is displayed

info.pop('occupation', None) # pop tries to remove 'occupation' and its value if it exists.
```

Traverse Dictionary Items (keys and associations)

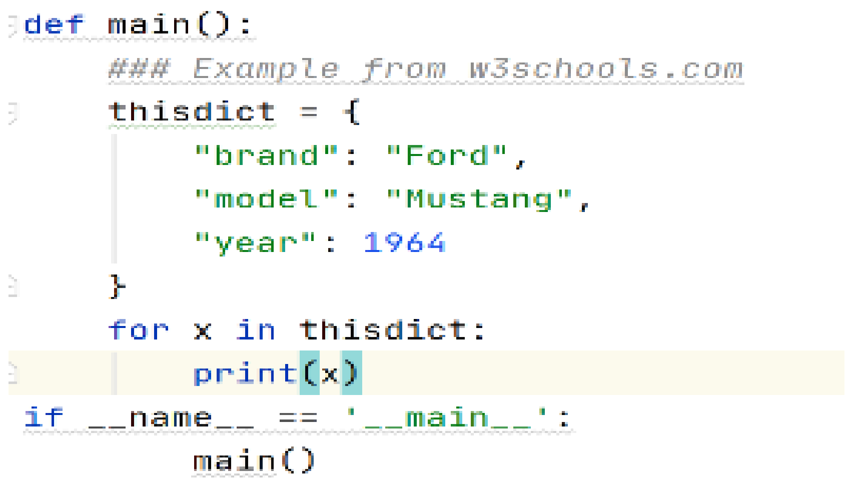


```
def main():
    grade_book={90:'A', 80:'B', 70:'C'}

    for (key, value) in grade_book.items():
        print ('Mark of {} represents letter grade of {}'.format (key,value))
if __name__=='__main__':
    main()
```

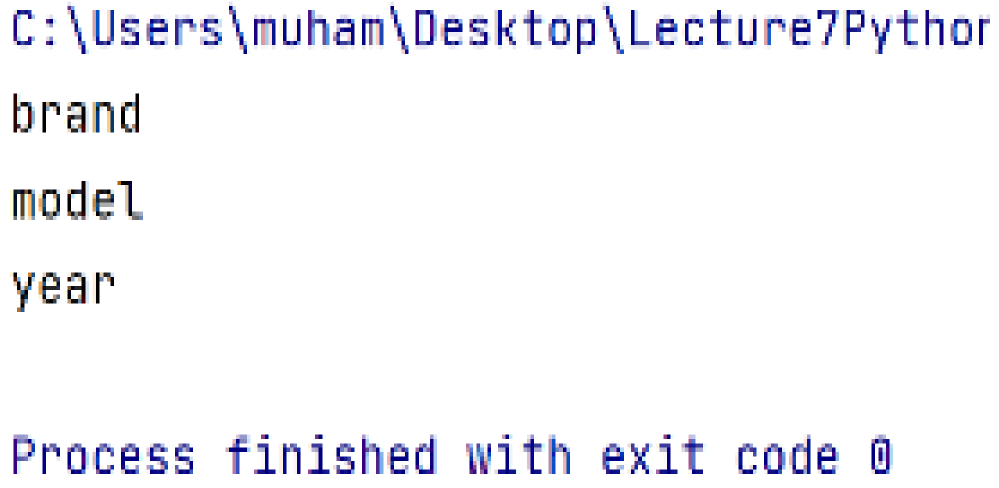


```
Mark of 80 represents letter grade of B
Mark of 90 represents letter grade of A
Mark of 70 represents letter grade of C

Process finished with exit code 0
```

Display keys from Dictionary (examples from w3schools.com)

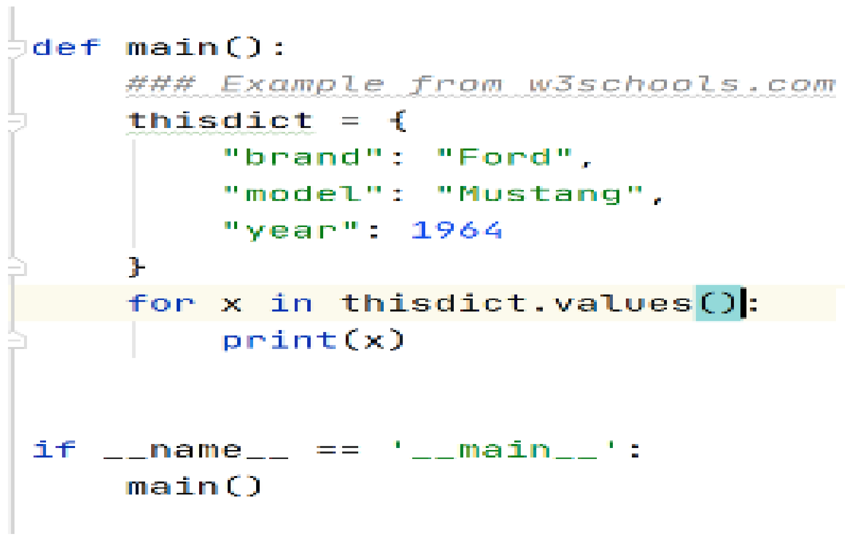


```
def main():
    ### Example from w3schools.com
    thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
    }
    for x in thisdict:
        print(x)
if __name__ == '__main__':
        main()
```

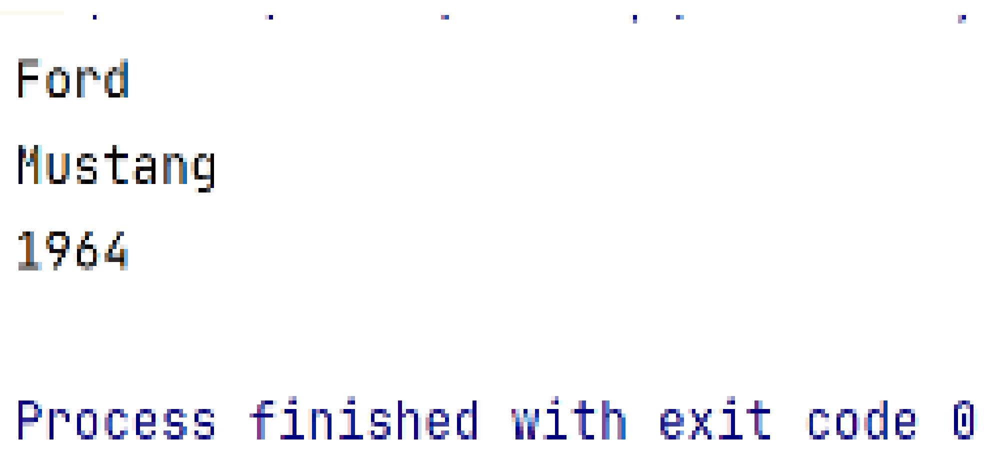


```
C:\Users\muham\Desktop\Lecture7Pythor
brand
model
year

Process finished with exit code 0
```

Display only item values (NOT keys) from Dictionary

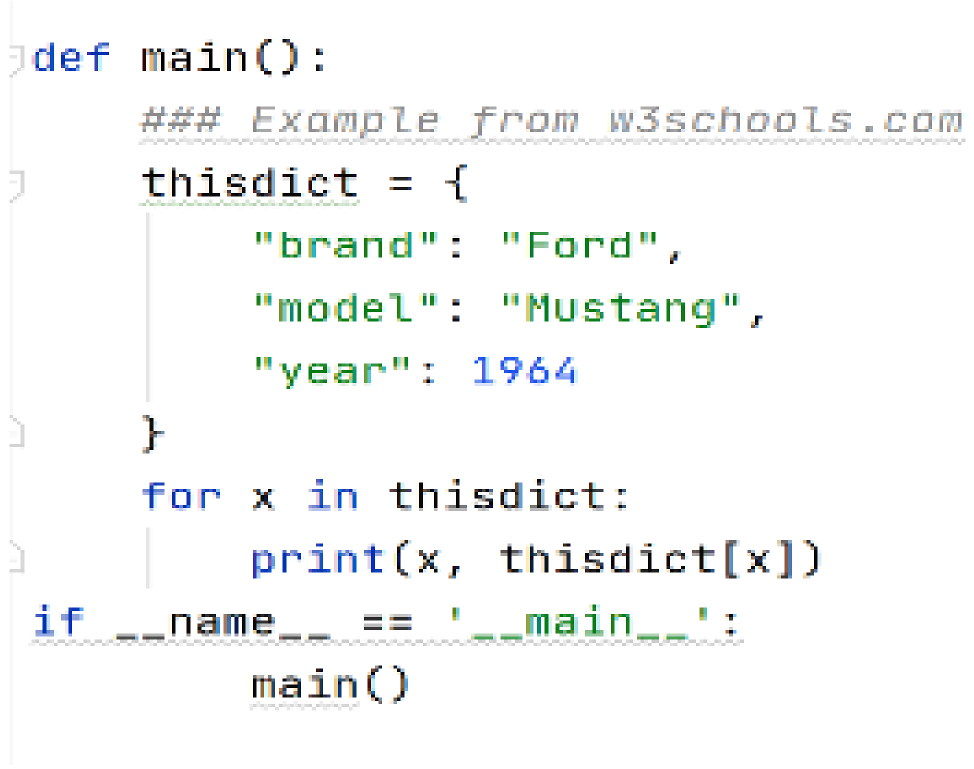


```
def main():
    ### Example from w3schools.com
    thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
    }
    for x in thisdict.values():
        print(x)


if __name__ == '__main__':
    main()
```

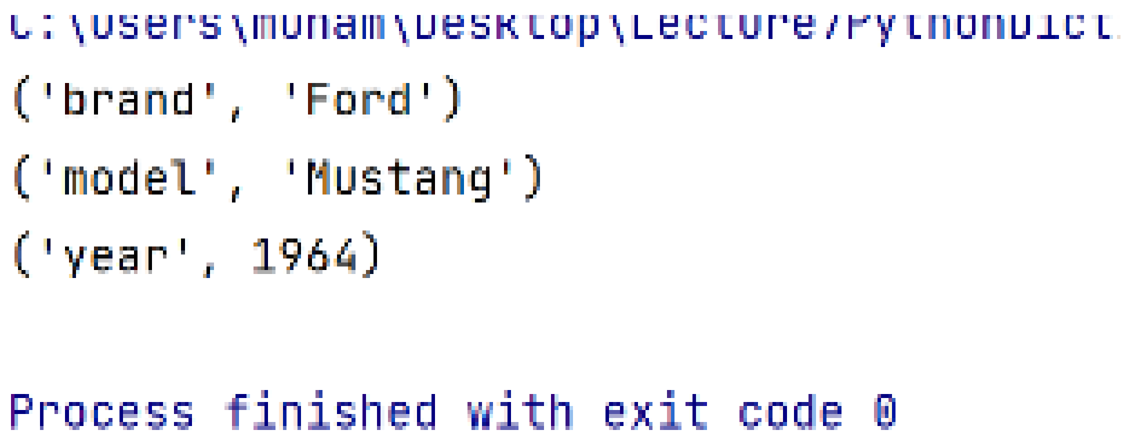


```
Ford
Mustang
1964

Process finished with exit code 0
```

Display Key and Value pair

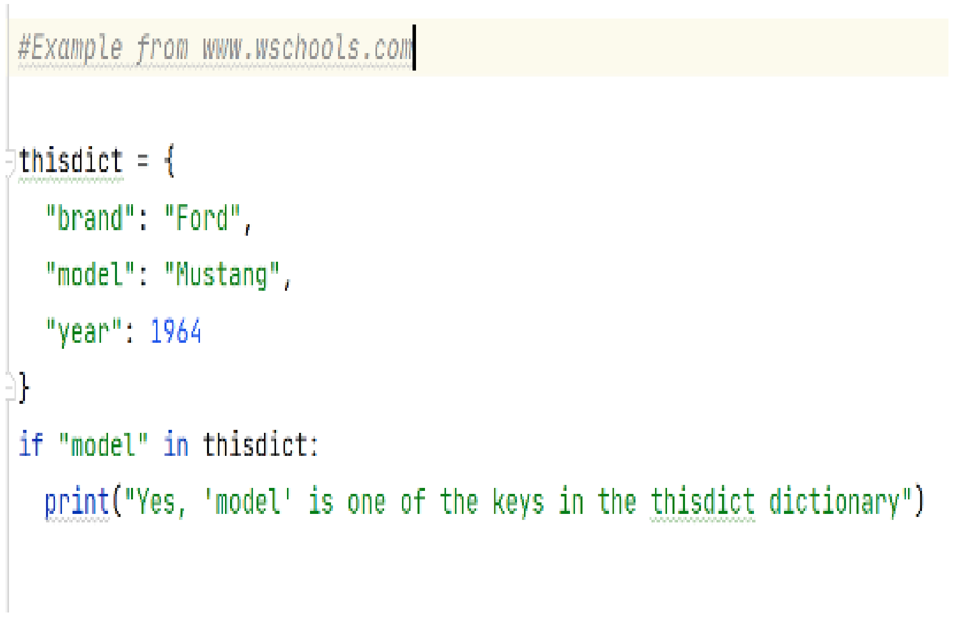


```
def main():
    ### Example from w3schools.com
    thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
    }
    for x in thisdict:
        print(x, thisdict[x])
if __name__ == '__main__':
        main()
```

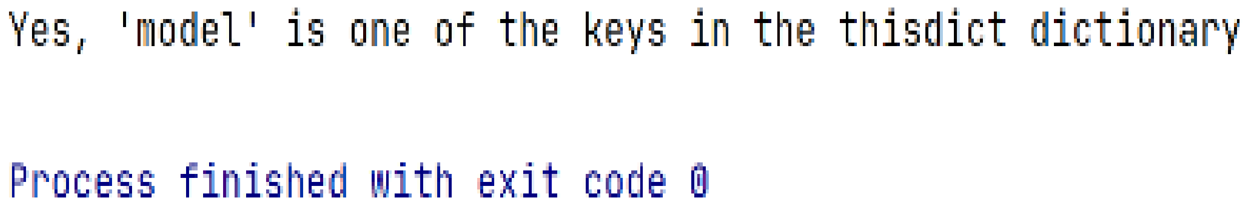


```
C:\Users\munam\Desktop\Lecture/PythonDict
('brand', 'Ford')
('model', 'Mustang')
('year', 1964)

Process finished with exit code 0
```

Check if key is in Dictionary

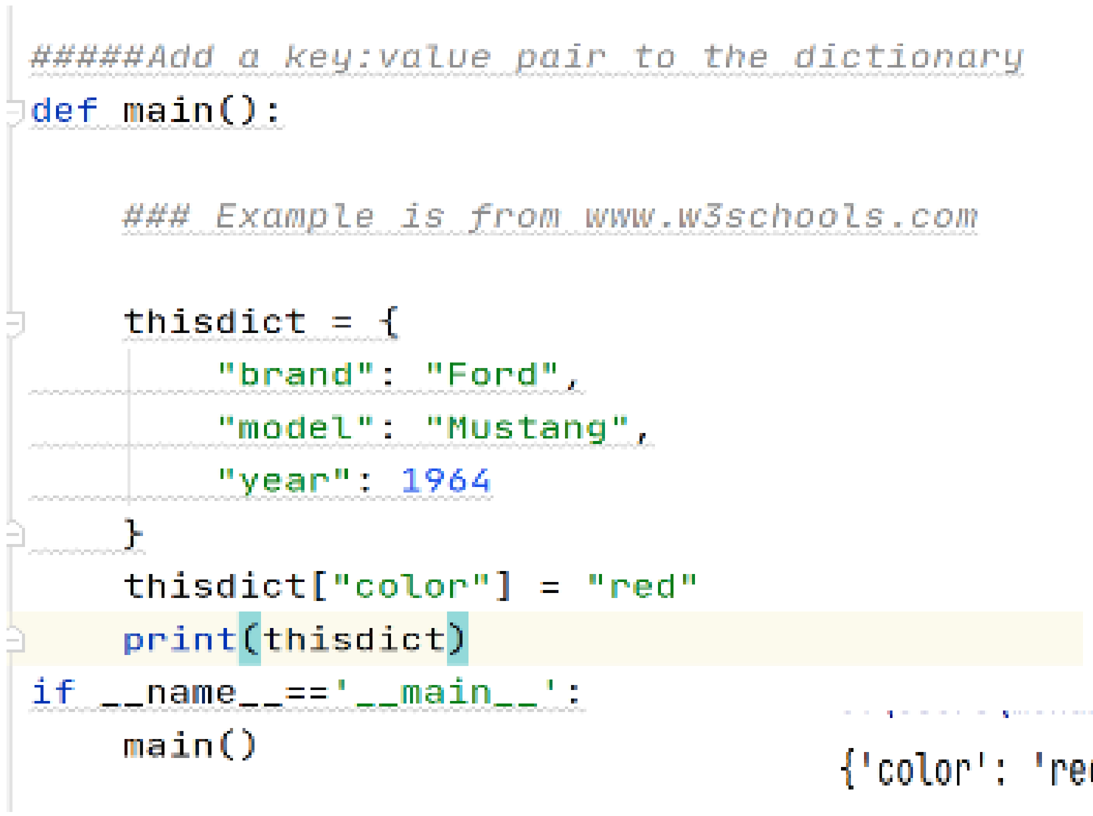


```
#Example from www.wschools.com

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

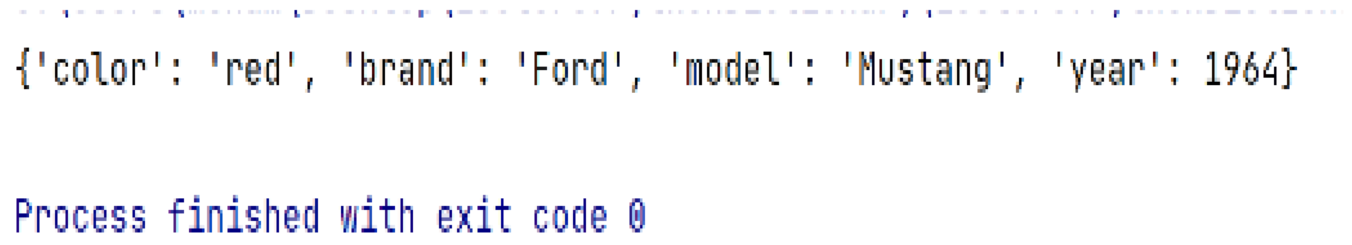


```
Yes, 'model' is one of the keys in the thisdict dictionary

Process finished with exit code 0
```

Add new Item (key:value pair) in Dictionary

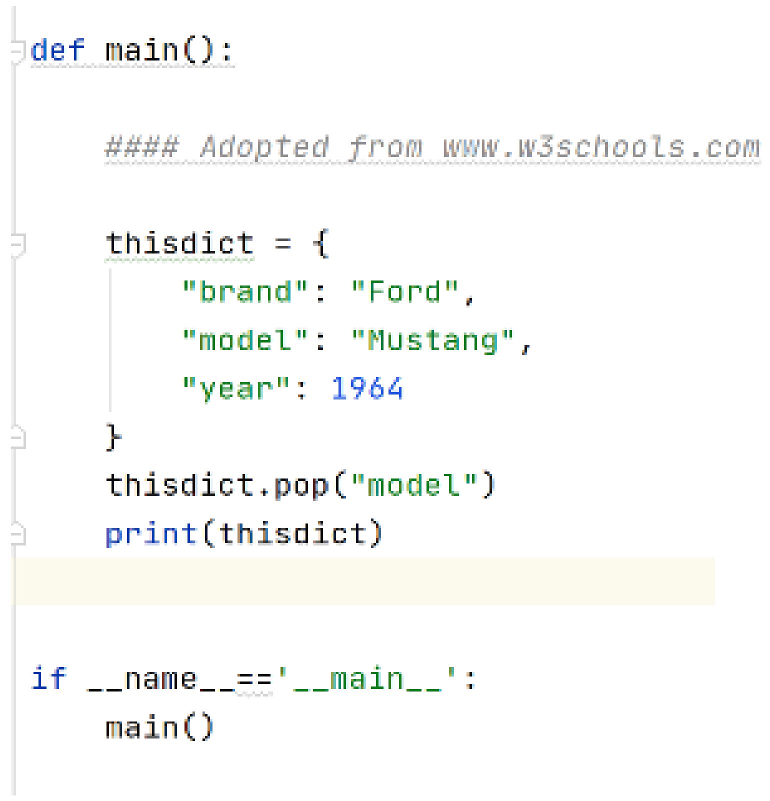


```
#####Add a key:value pair to the dictionary
def main():

    ### Example is from www.w3schools.com

    thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
    }
    thisdict["color"] = "red"
    print(thisdict)
if __name__=='__main__':
    main()
```

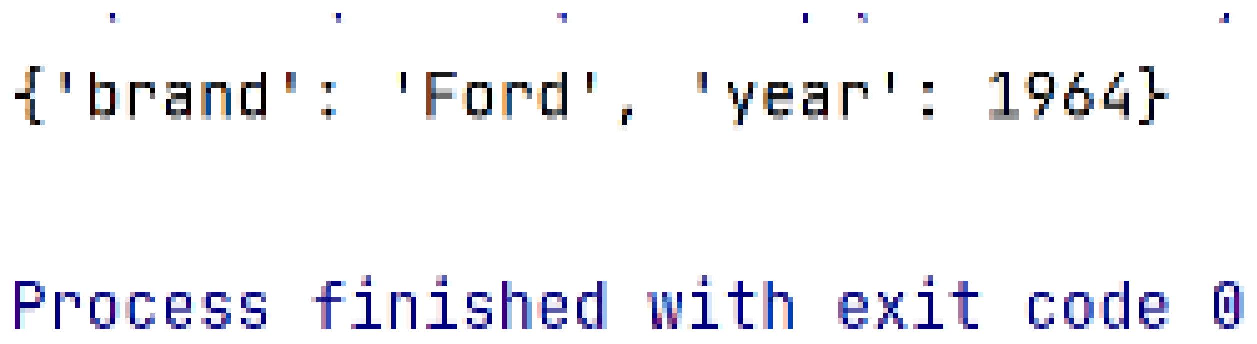


```
{'color': 'red', 'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

Process finished with exit code 0
```

Removing an Item (key:value pair) from Dictionary using pop()

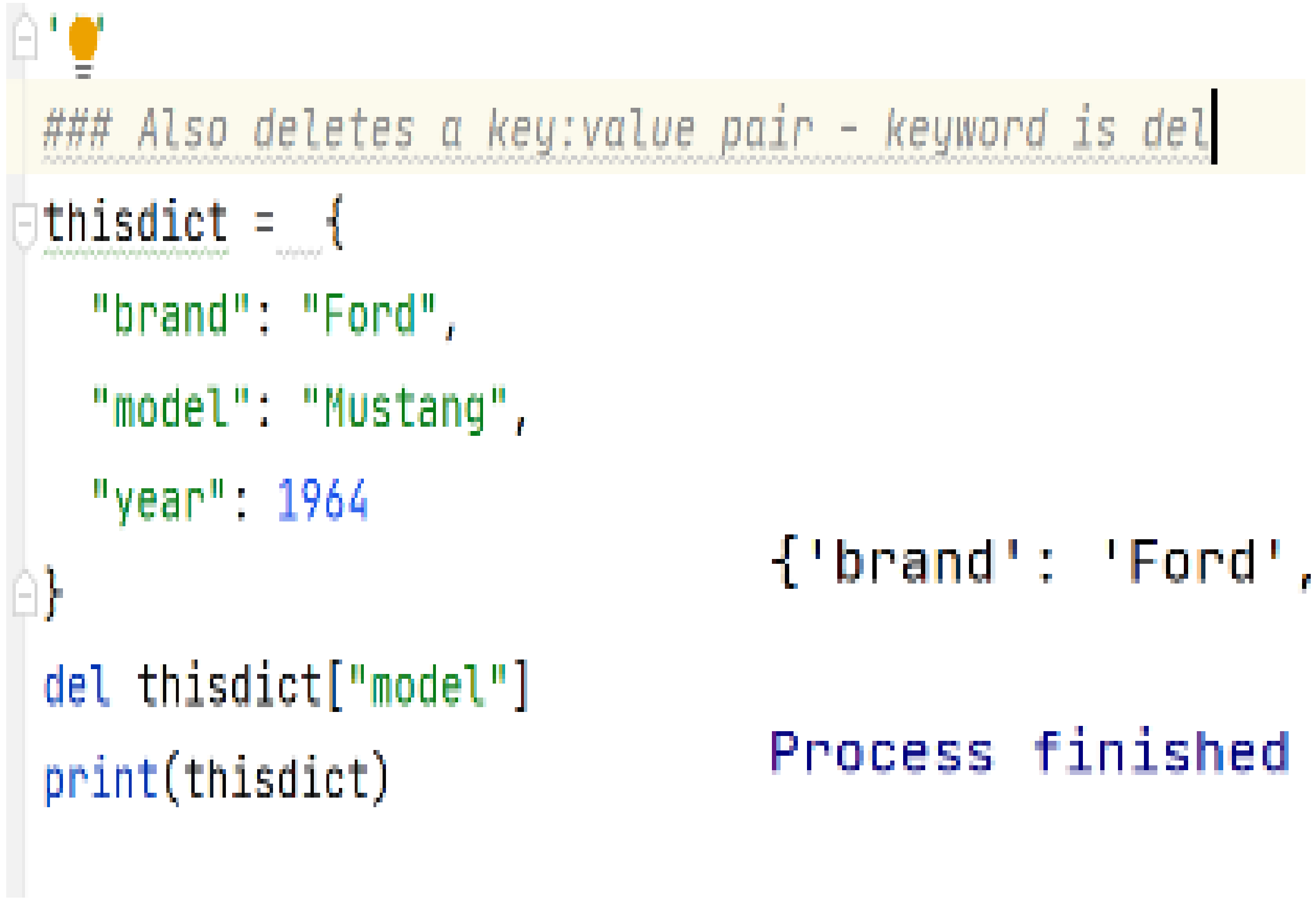


```
def main():

    #### Adopted from www.w3schools.com

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


if __name__=='__main__':
    main()
```

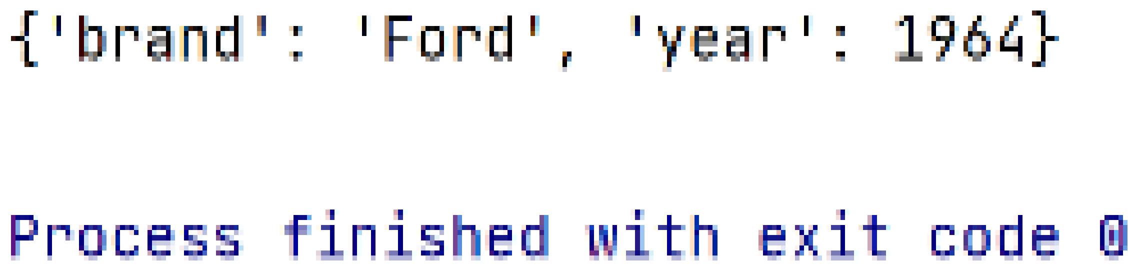


```
{'brand': 'Ford', 'year': 1964}


Process finished with exit code 0
```

Removing item (key:value pair) from Dictionary using del keyword (command)

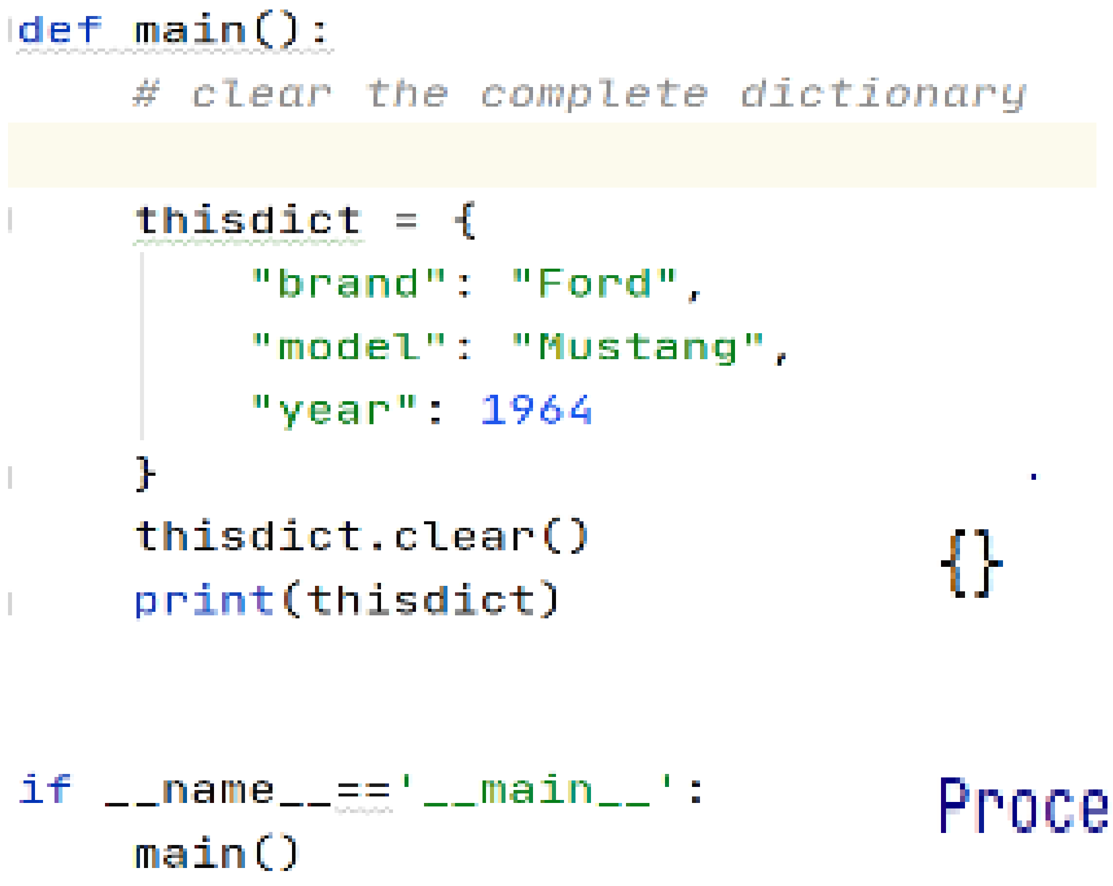


```
'
### Also deletes a key:value pair - keyword is del
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

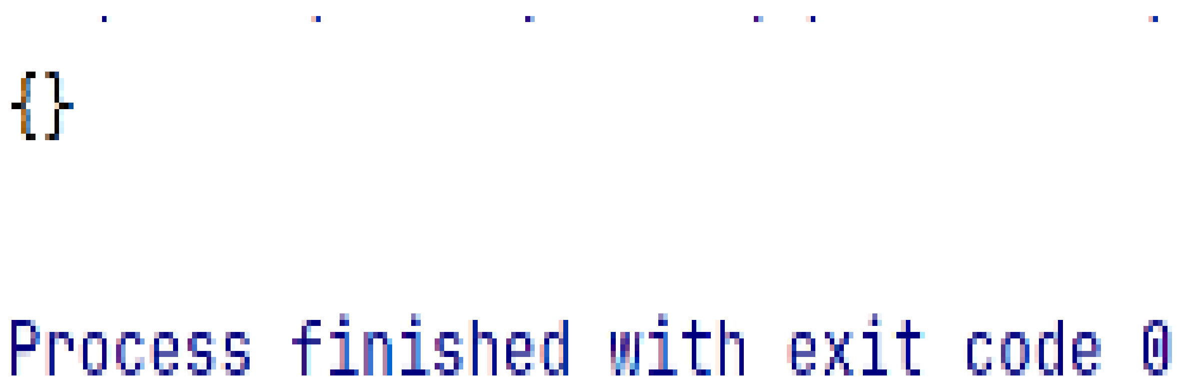


```
{'brand': 'Ford', 'year': 1964}

Process finished with exit code 0
```

Deleting all items in Dictionary using clear() Dictionary API

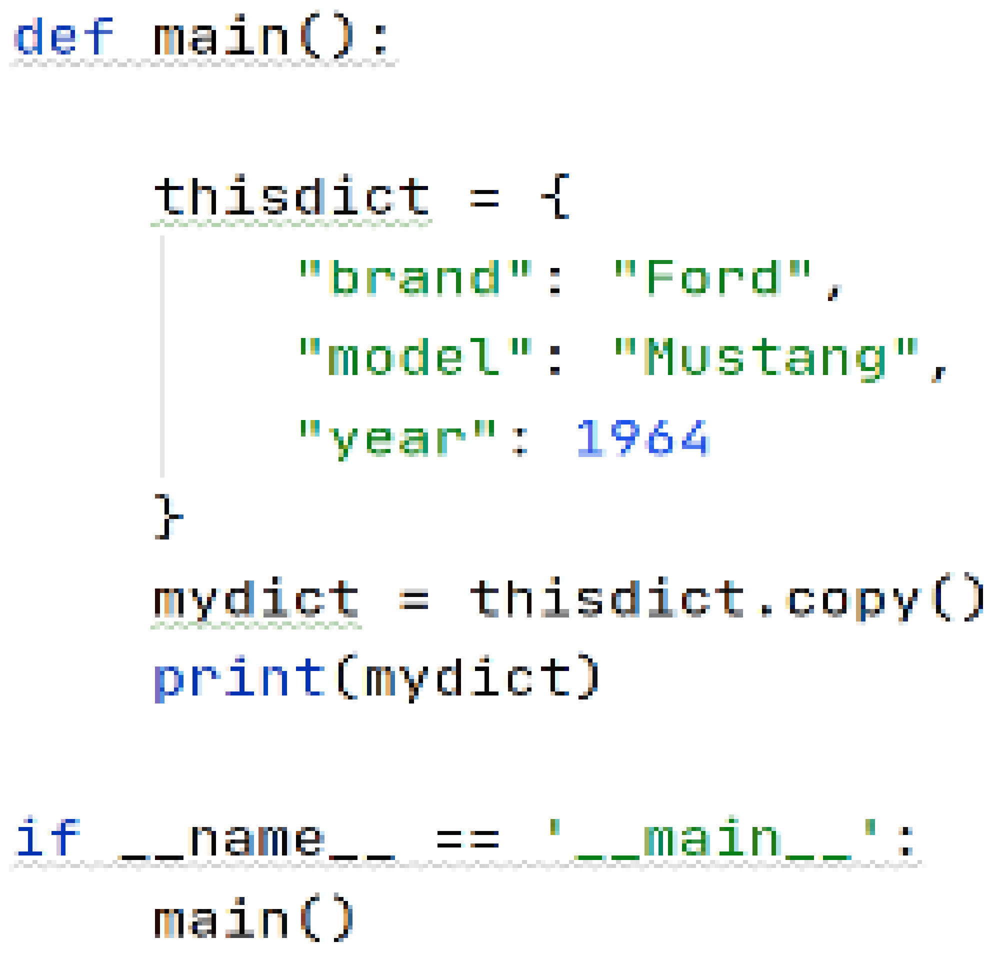


```python
def main():
    # clear the complete dictionary

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


if __name__=='__main__':
    main()
```

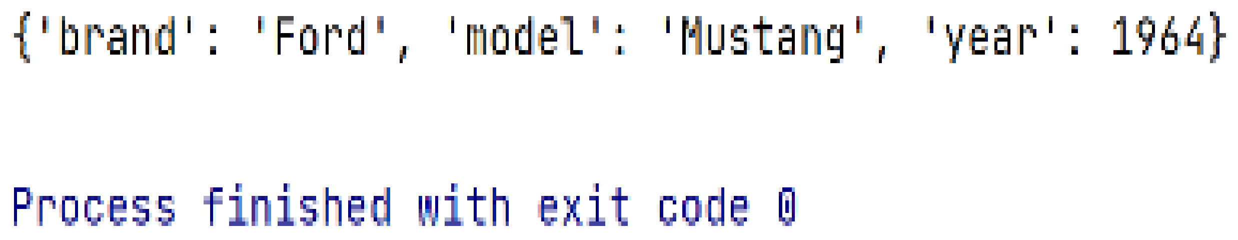


```
{}

Process finished with exit code 0
```

Create copy of Dictionary using copy() API

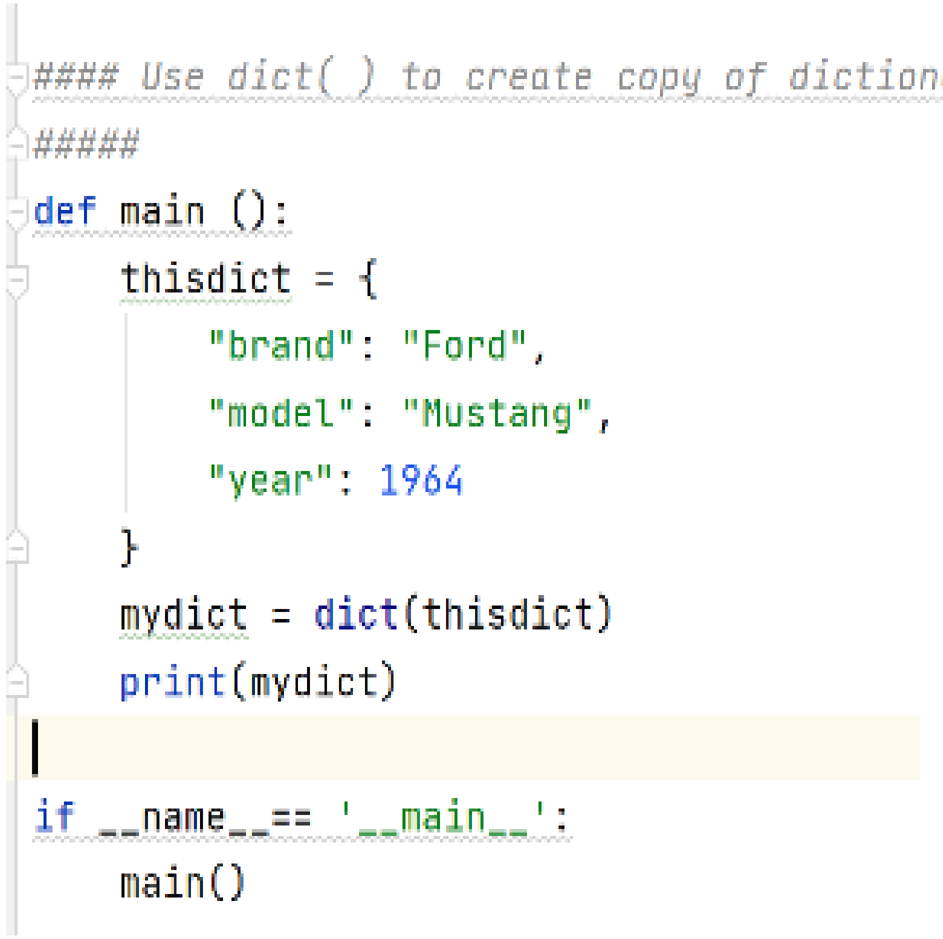


```python
def main():

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

if __name__ == '__main__':
    main()
```

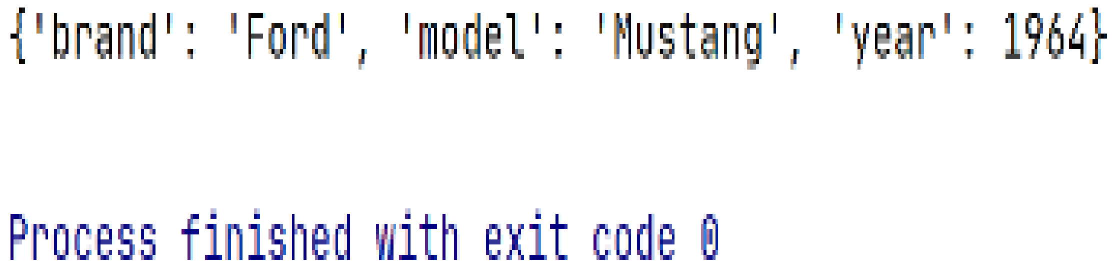


```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

Process finished with exit code 0
```

Create copy of Dictionary using dict() constructor

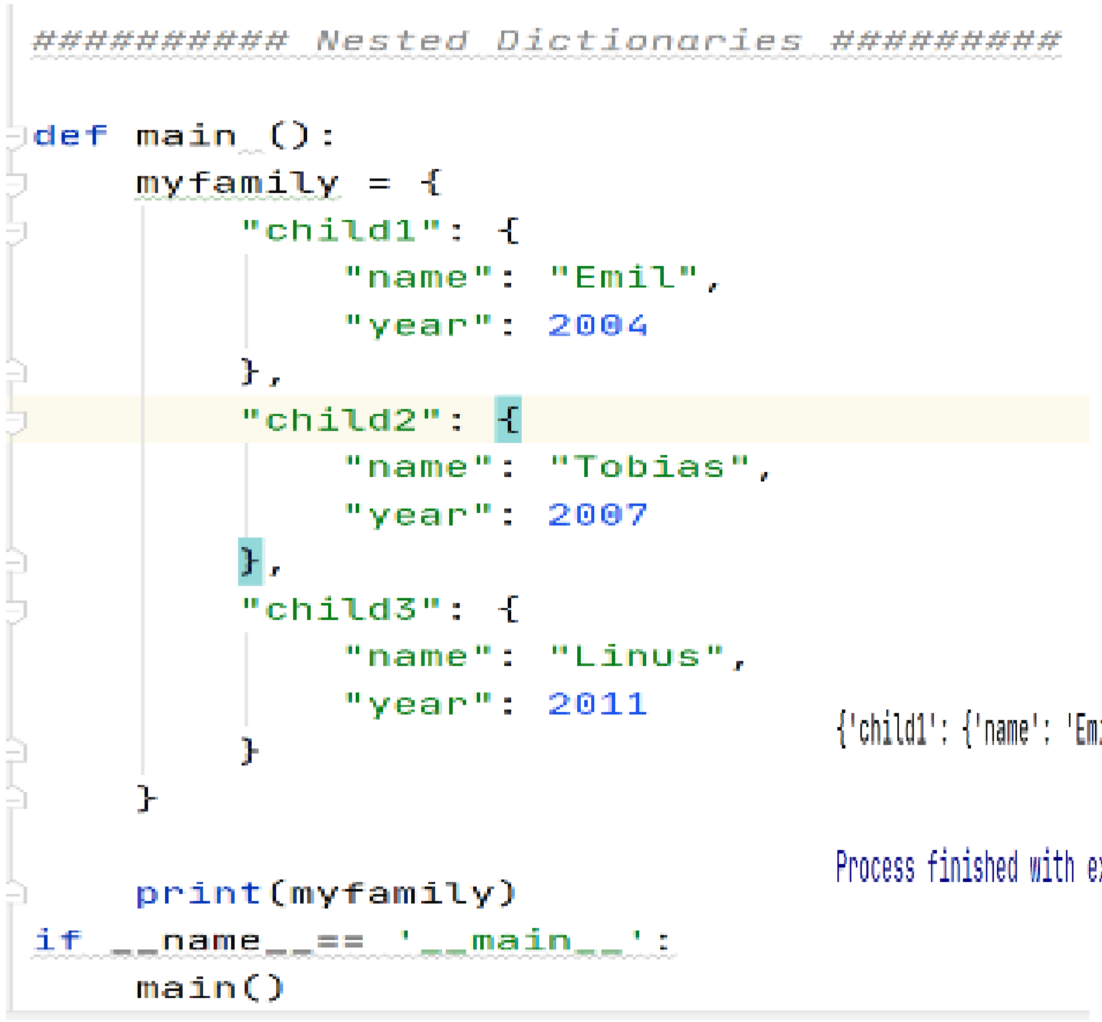


```python
#### Use dict( ) to create copy of dictionary
#####
def main ():
    thisdict = {
        "brand": "Ford",
        "model": "Mustang",
        "year": 1964
    }
    mydict = dict(thisdict)
    print(mydict)

if __name__== '__main__':
    main()
```



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


Process finished with exit code 0
```

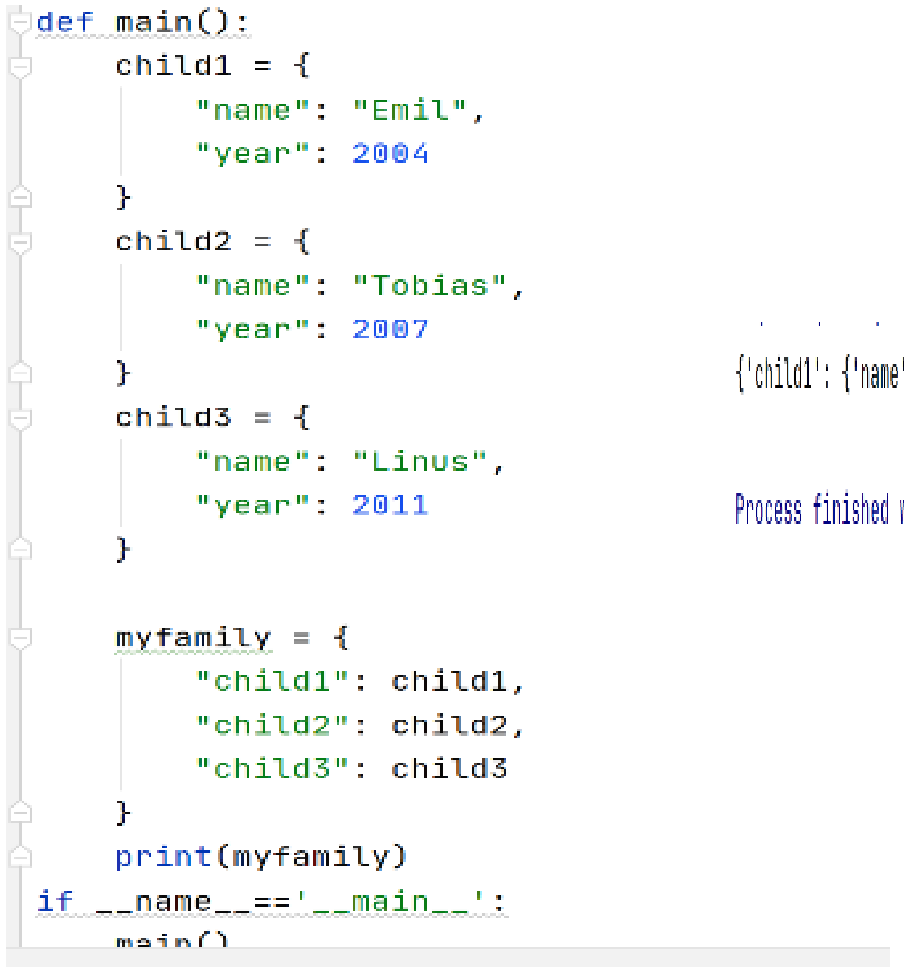
Nested Dictionaries



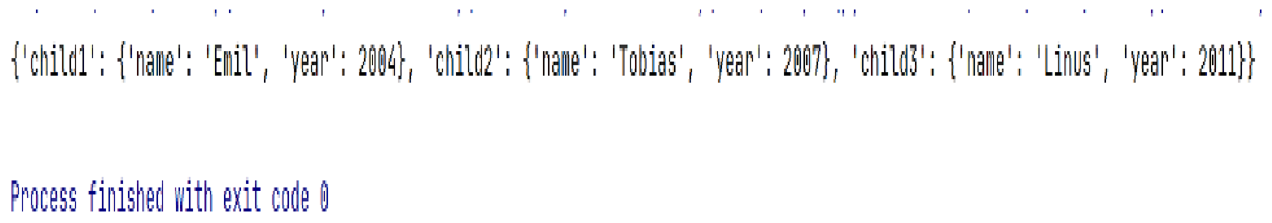
```
########## Nested Dictionaries #########

def main ():
    myfamily = {
        "child1": {
            "name": "Emil",
            "year": 2004
        },
        "child2": {
            "name": "Tobias",
            "year": 2007
        },
        "child3": {
            "name": "Linus",
            "year": 2011
        }
    }

    print(myfamily)
if __name__== '__main__':
    main()
```

```
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name': 'Linus', 'year': 2011}}


Process finished with exit code 0
```

Creating Nested Dictionary (different method)

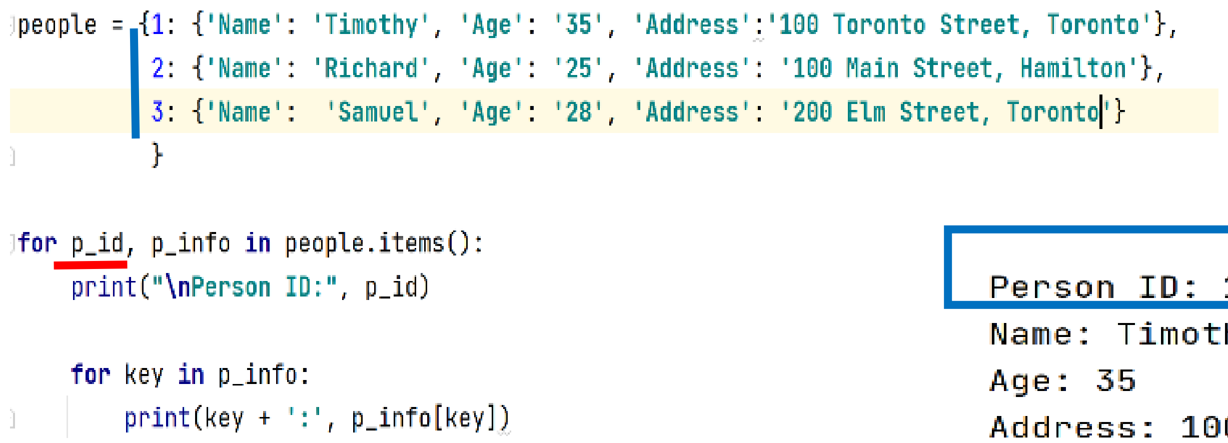


```
def main():
    child1 = {
        "name": "Emil",
        "year": 2004
    }
    child2 = {
        "name": "Tobias",
        "year": 2007
    }
    child3 = {
        "name": "Linus",
        "year": 2011
    }

    myfamily = {
        "child1": child1,
        "child2": child2,
        "child3": child3
    }
    print(myfamily)
if __name__=='__main__':
    main()
```

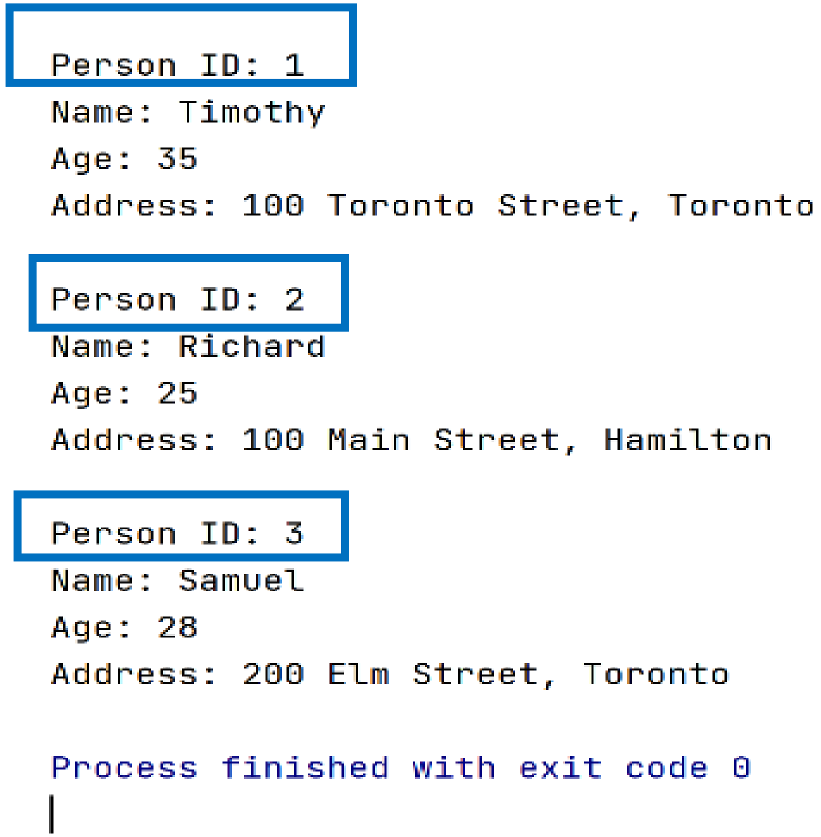


```
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name': 'Linus', 'year': 2011}}


Process finished with exit code 0
```

Traversing nested Dictionary

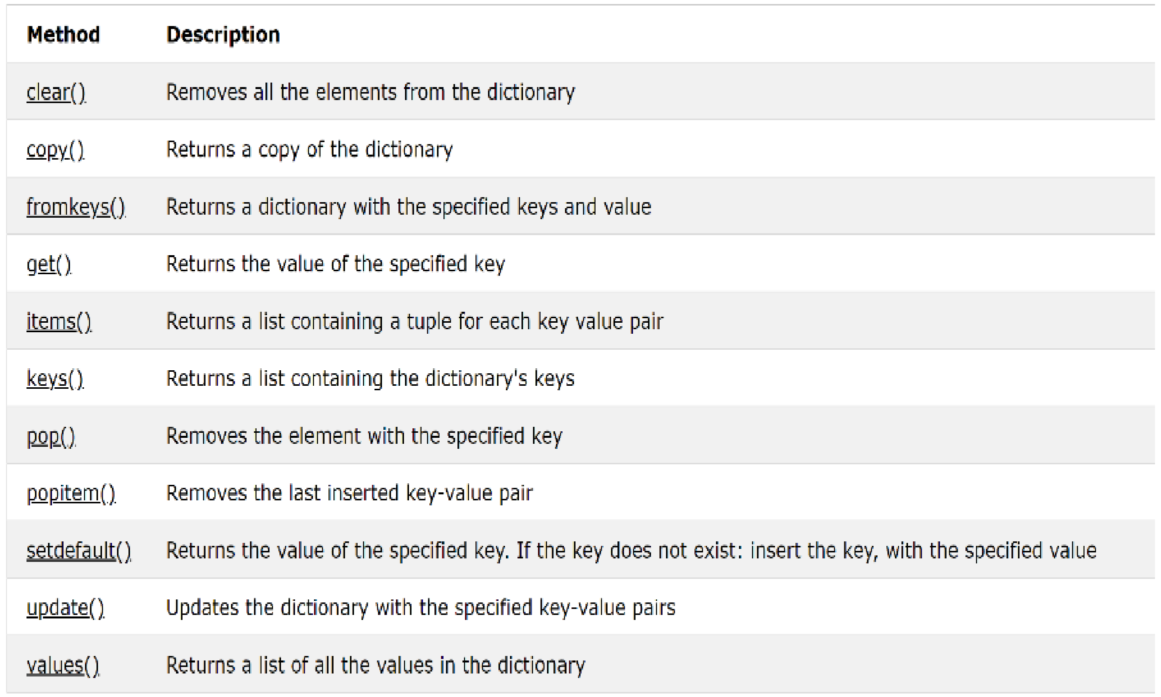


```
people = {1: {'Name': 'Timothy', 'Age': '35', 'Address':'100 Toronto Street, Toronto'},
          2: {'Name': 'Richard', 'Age': '25', 'Address': '100 Main Street, Hamilton'},
          3: {'Name':  'Samuel', 'Age': '28', 'Address': '200 Elm Street, Toronto'}
          }

for p_id, p_info in people.items():
    print("\nPerson ID:", p_id)

    for key in p_info:
        print(key + ':', p_info[key])
```



```

Person ID: 1
Name: Timothy
Age: 35
Address: 100 Toronto Street, Toronto

Person ID: 2
Name: Richard
Age: 25
Address: 100 Main Street, Hamilton

Person ID: 3
Name: Samuel
Age: 28
Address: 200 Elm Street, Toronto

Process finished with exit code 0
```

List of important Dictionary functions (w3schools.com)



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 value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
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

Files

- Stores data in non-volatile environment
- Need to provide name of the file (or absolute path including the filename)
- By default file is created in current working directory
- Files may be text files or they may be binary file
- By default, file is created as text file

Modes to open file

- There are four modes to open file:
  - Write mode – ‘w’ – File is created in write mode. If file exists, its data is lost and new data is over-written. If file does not already exists, it is created
  - Read mode –’r’ – File is opened in read mode with pointer pointing to the first character in the file
  - Append mode –’a’ – File is opened in append (write) mode, but internal pointer points to the next line where the new data is written in file
  - Create mode – ‘x’ – Creates specified file. If file already exists, Exception is generated

  - File can be opened in binary mode (‘b’) for storing and reading binary data (e.g. images etc.)
  - File may be opened in text mode (‘t’ – which is default mode of opening file) for storing text data and reading text data

Files – open options

- data_file = open (file_name, ‘w’)
- data_file = open (file_name, ‘wb’) - open/create binary file to write binary data in the file
- data_file = open (file_name, ‘rb’) – open binary file for reading binary contents from file
- data_file = open (file_name, ‘a’) – open text file in append mode to add new data to the file

File – read functions

read() – reads all contents from file

e.g., file_data.read()

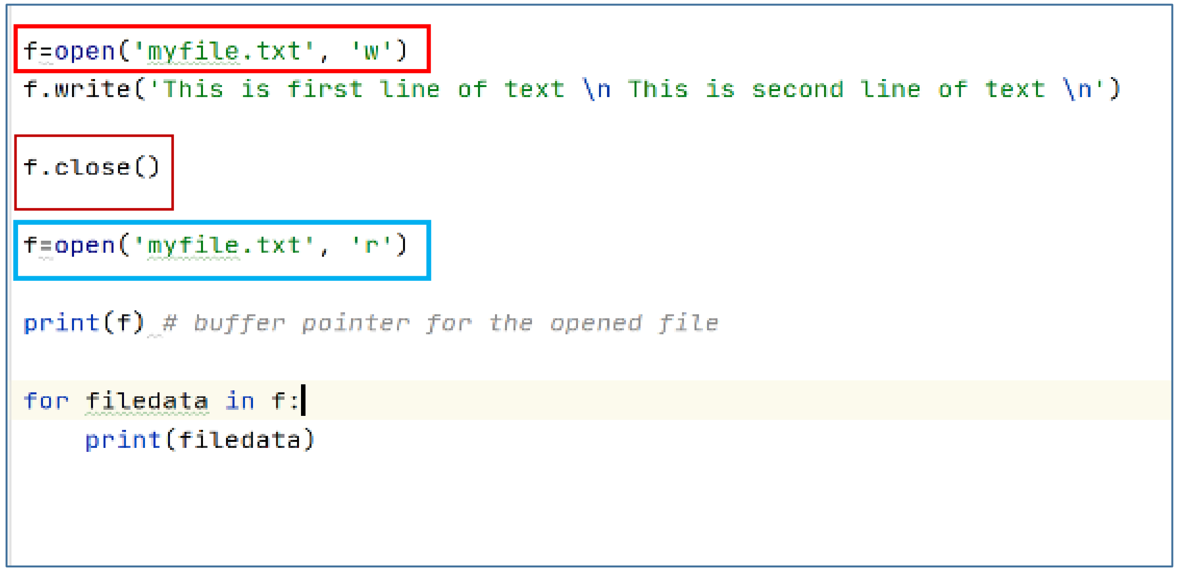
readline() – reads one line at a time (line separated by newline character ‘\n’)

readlines() – read all contents from file

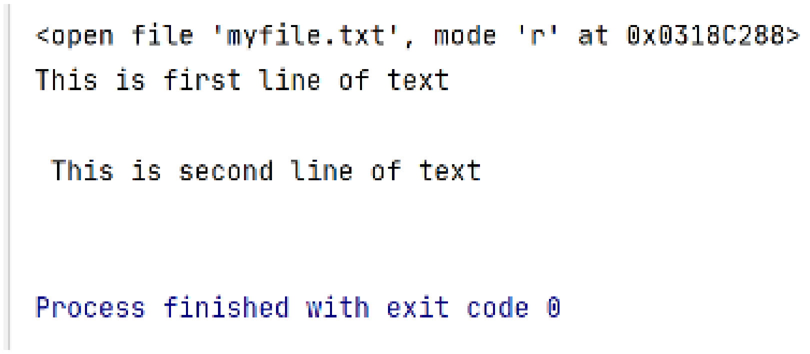
write() – write data in file

e.g., file_data.write()

Text Files – Open a text file (for write and read)

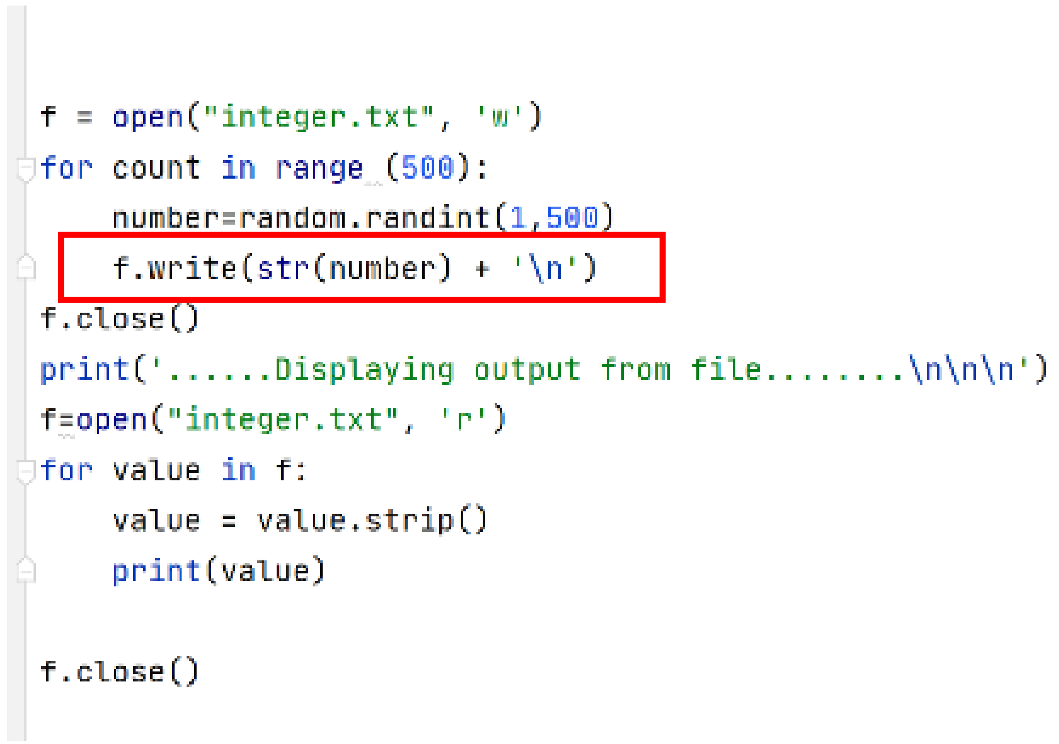


```
f=open('myfile.txt', 'w')
f.write('This is first line of text \n This is second line of text \n')

f.close()

f=open('myfile.txt', 'r')

print(f) # buffer pointer for the opened file

for filedata in f:
    print(filedata)
```



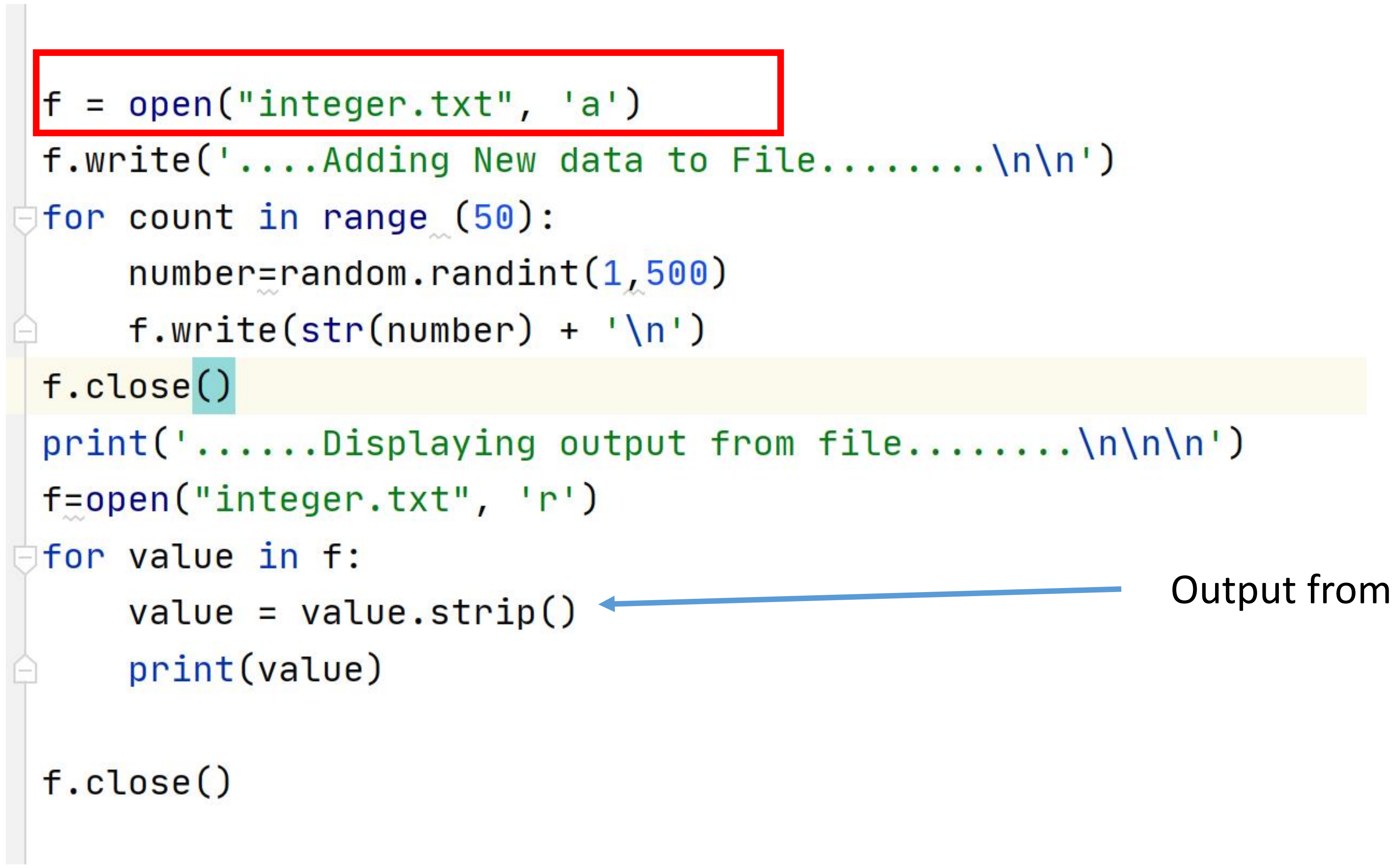
```
<open file 'myfile.txt', mode 'r' at 0x0318C288>
This is first line of text

 This is second line of text



Process finished with exit code 0
```

Converting integers to String and write in file



```
f = open("integer.txt", 'w')
for count in range (500):
    number=random.randint(1,500)
    f.write(str(number) + '\n')
f.close()
print('......Displaying output from file........\n\n\n')
f=open("integer.txt", 'r')
for value in f:
    value = value.strip()
    print(value)

f.close()
```

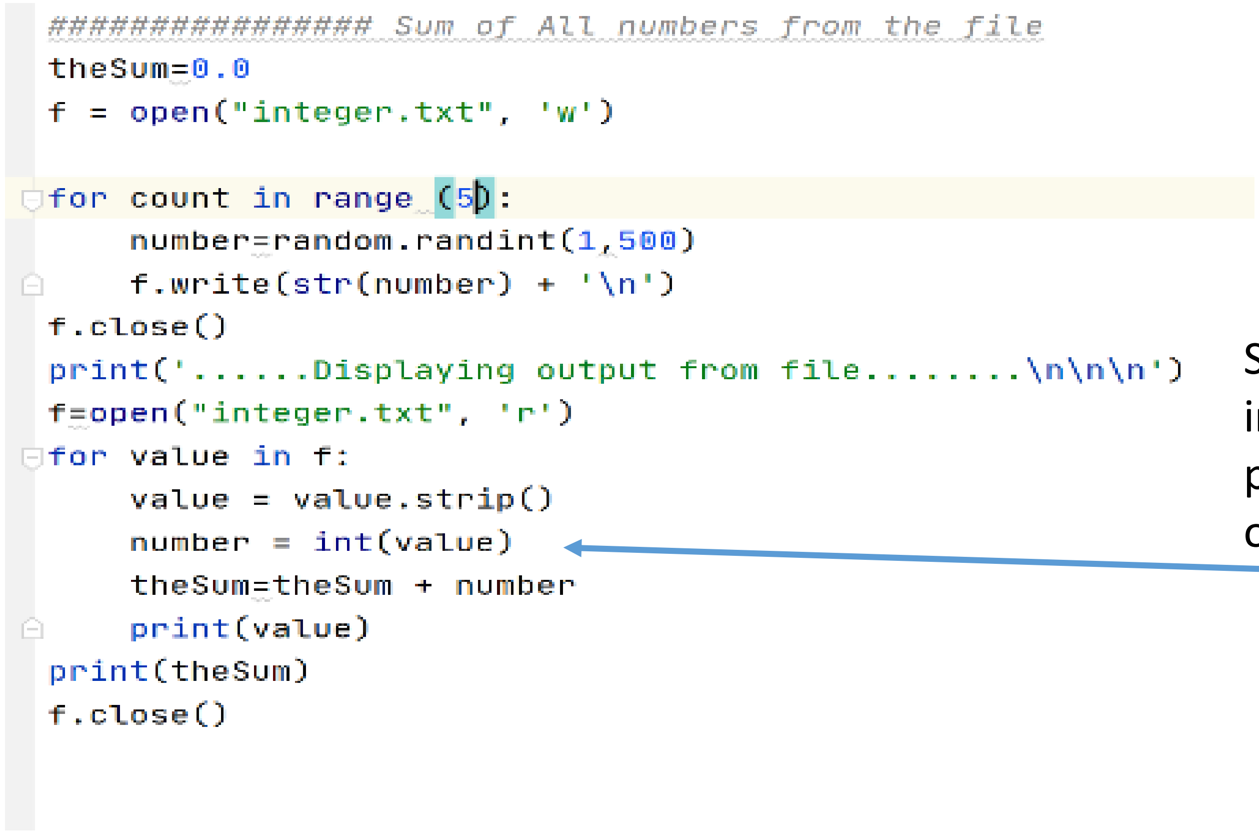
Appending data to contents in file



```
f = open("integer.txt", 'a')
f.write('....Adding New data to File........\n\n')
for count in range (50):
    number=random.randint(1,500)
    f.write(str(number) + '\n')
f.close()
print('......Displaying output from file........\n\n\n')
f=open("integer.txt", 'r')
for value in f:
    value = value.strip()
    print(value)

f.close()
```

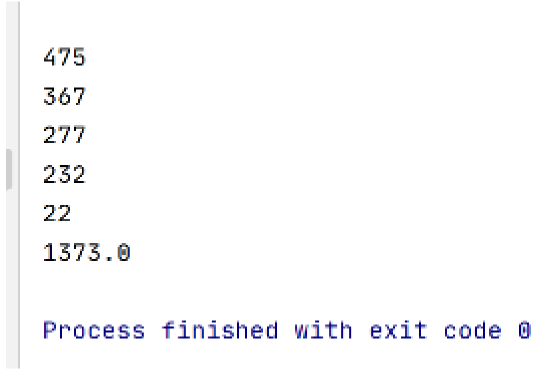
Output from file is STRING

Sum of data stored in file



```
################ Sum of All numbers from the file
theSum=0.0
f = open("integer.txt", 'w')

for count in range (5):
    number=random.randint(1,500)
    f.write(str(number) + '\n')
f.close()
print('......Displaying output from file........\n\n\n')
f=open("integer.txt", 'r')
for value in f:
    value = value.strip()
    number = int(value)
    theSum=theSum + number
    print(value)
print(theSum)
f.close()
```

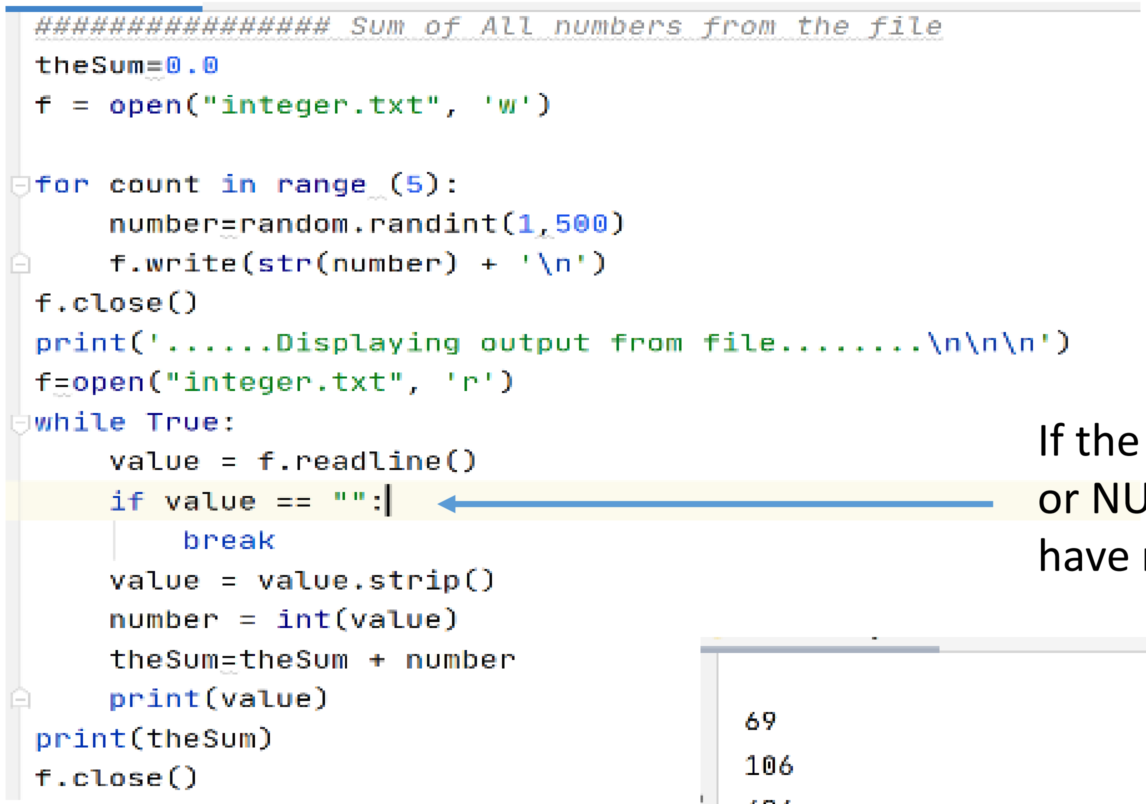
String numeric is converted into integer value before performing arithmetic operation



```
475
367
277
232
22
1373.0

Process finished with exit code 0
```

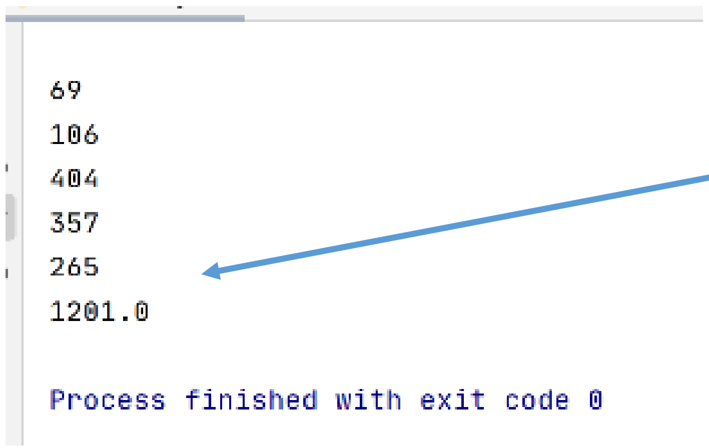
Output from file displayed with Sum at the end

Sum of stored numbers using ‘while-loop’



```
################ Sum of All numbers from the file
theSum=0.0
f = open("integer.txt", 'w')

for count in range (5):
    number=random.randint(1,500)
    f.write(str(number) + '\n')
f.close()
print('......Displaying output from file........\n\n\n')
f=open("integer.txt", 'r')
while True:
    value = f.readline()
    if value == "":
        break
    value = value.strip()
    number = int(value)
    theSum=theSum + number
    print(value)
print(theSum)
f.close()
```

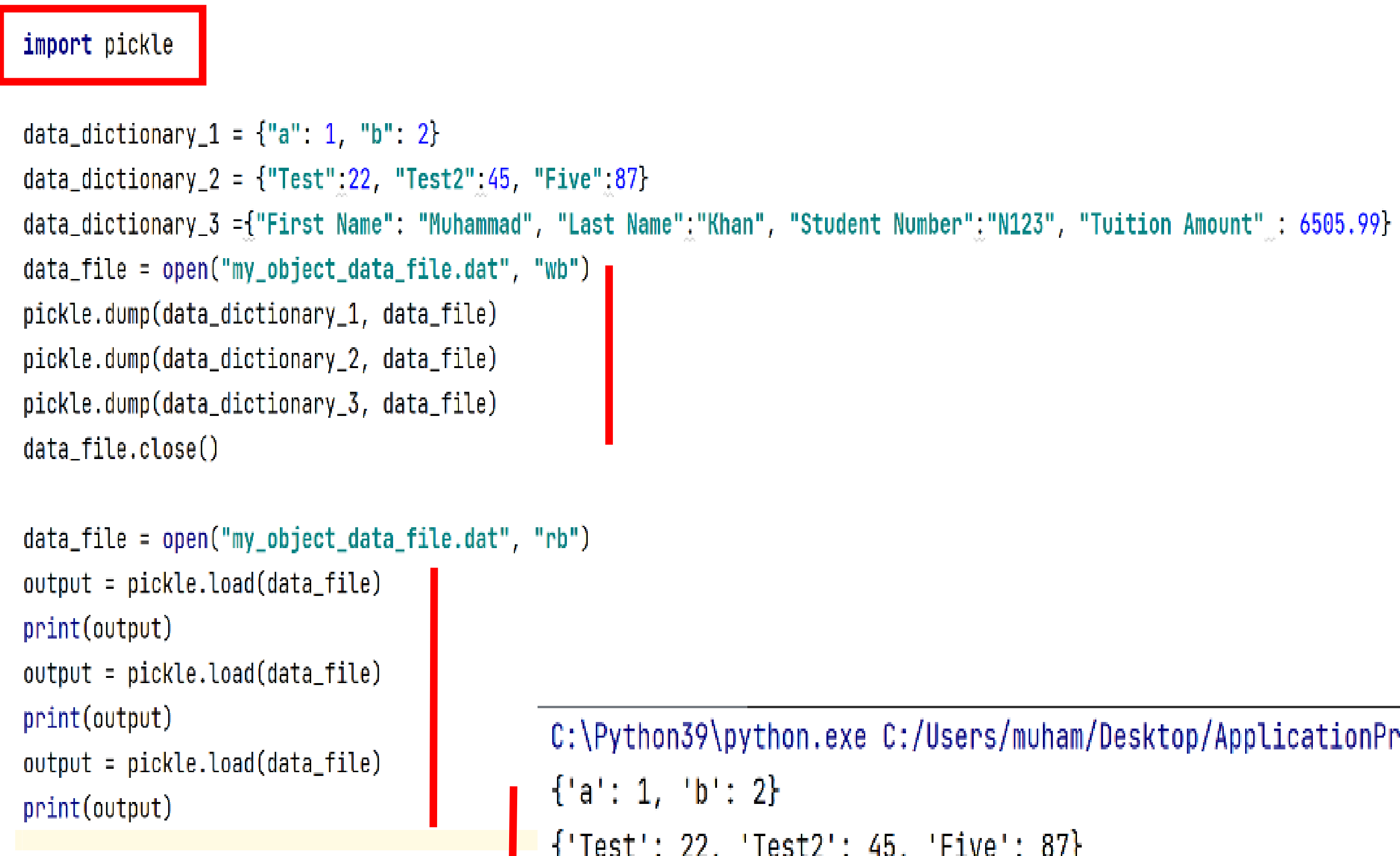
If the line read from file is ‘EMPTY’ or NULL, then break the loop, we have reached the END OF FILE



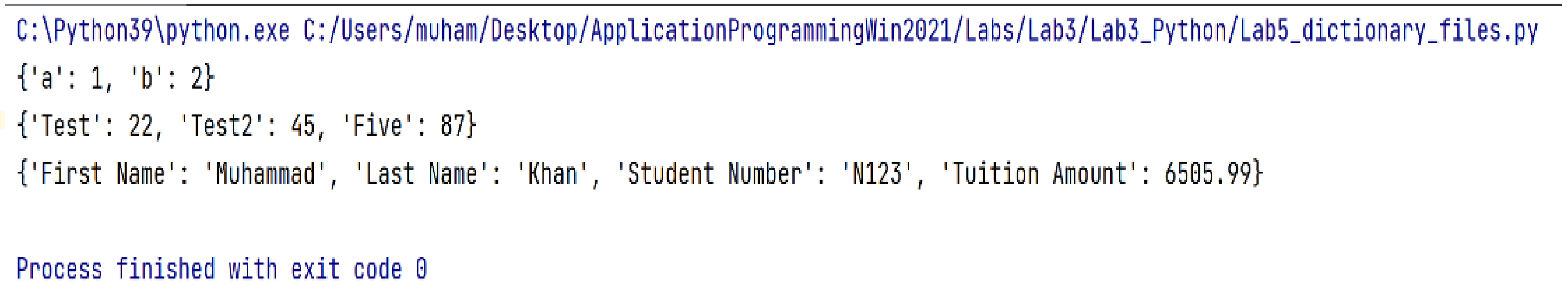
```
69
106
404
357
265
1201.0

Process finished with exit code 0
```

Output from file (numeric vales) and sum of the five numbers read from the file

Saving multiple objects in a binary file:

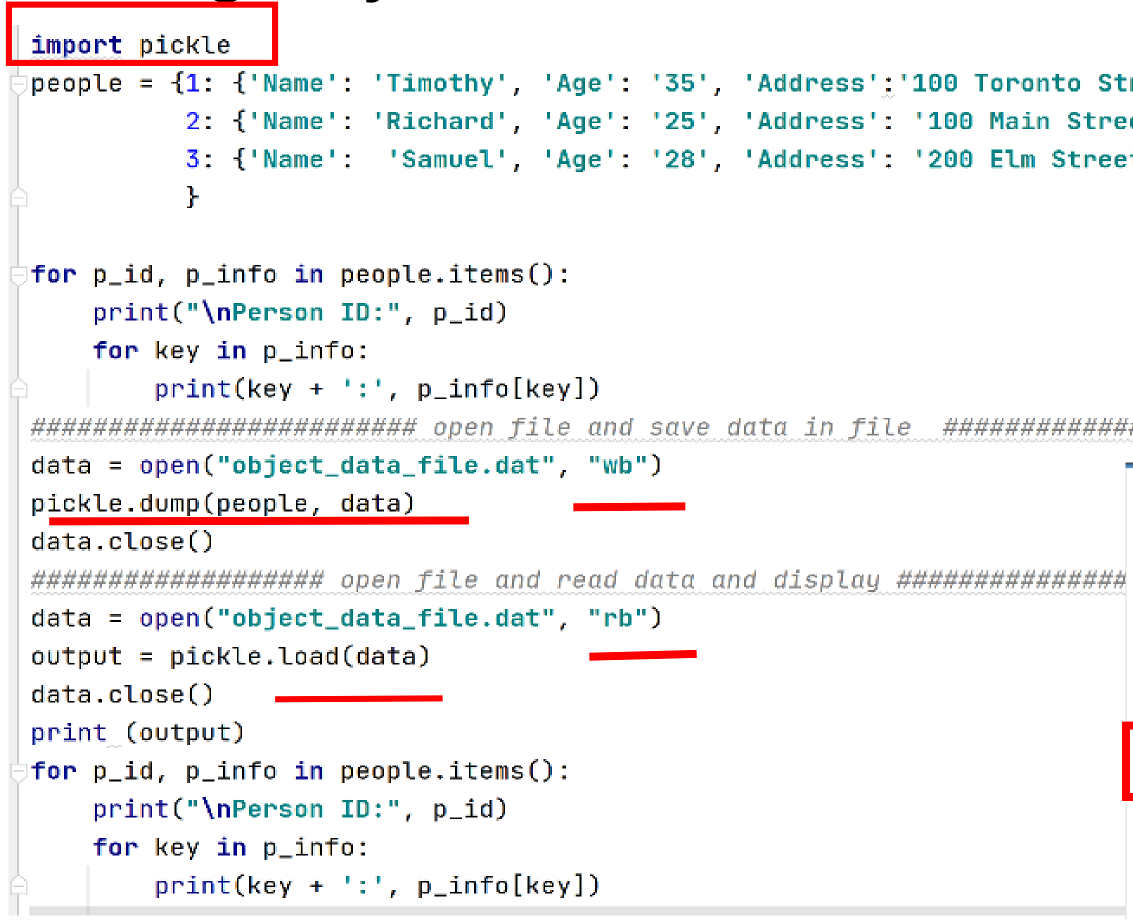


```
import pickle

data_dictionary_1 = {"a": 1, "b": 2}
data_dictionary_2 = {"Test":22, "Test2":45, "Five":87}
data_dictionary_3 ={"First Name": "Muhammad", "Last Name":"Khan", "Student Number":"N123", "Tuition Amount" : 6505.99}
data_file = open("my_object_data_file.dat", "wb")
pickle.dump(data_dictionary_1, data_file)
pickle.dump(data_dictionary_2, data_file)
pickle.dump(data_dictionary_3, data_file)
data_file.close()

data_file = open("my_object_data_file.dat", "rb")
output = pickle.load(data_file)
print(output)
output = pickle.load(data_file)
print(output)
output = pickle.load(data_file)
print(output)
```

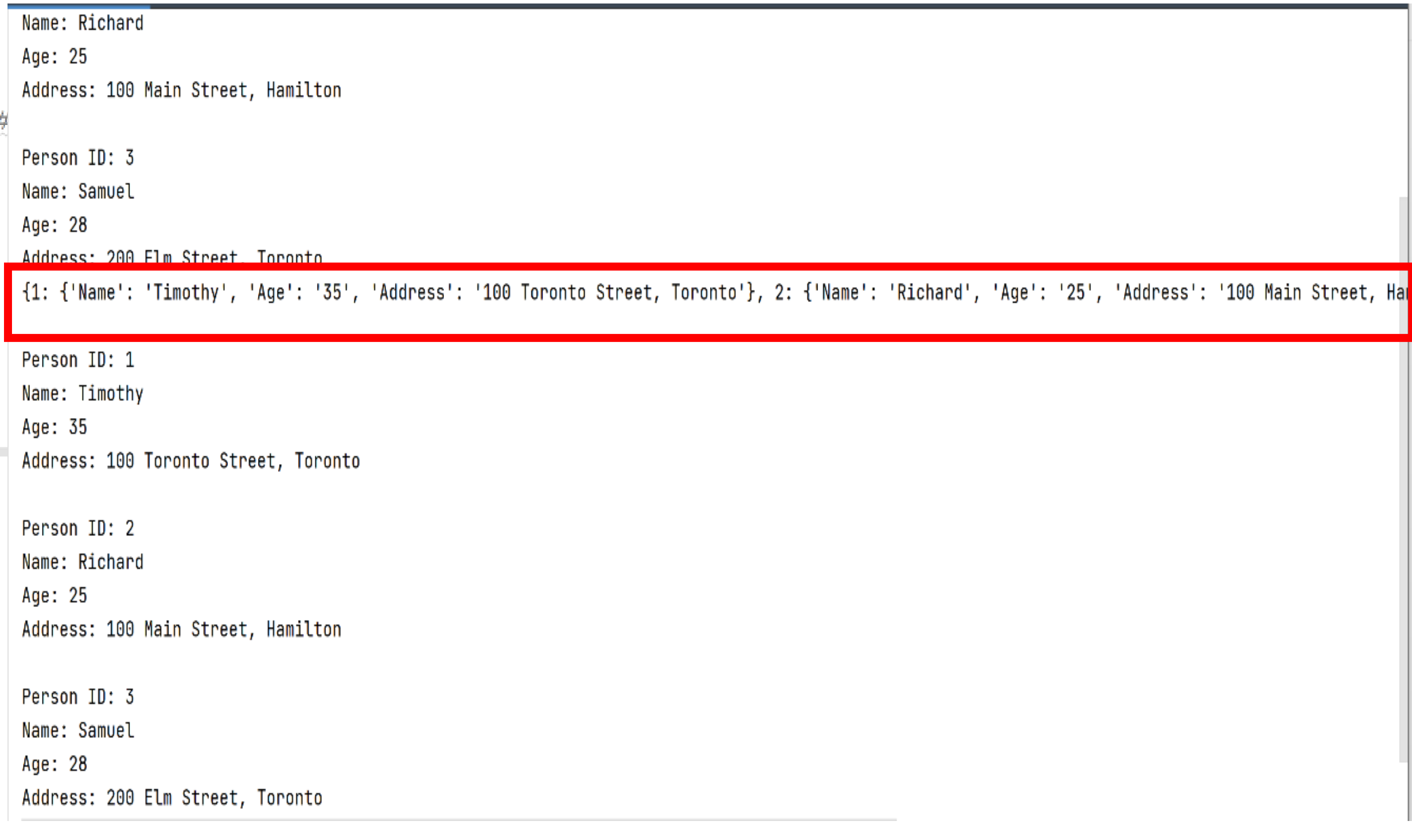


```
C:\Python39\python.exe C:/Users/muham/Desktop/ApplicationProgrammingWin2021/Labs/Lab3/Lab3_Python/Lab5_dictionary_files.py
{'a': 1, 'b': 2}
{'Test': 22, 'Test2': 45, 'Five': 87}
{'First Name': 'Muhammad', 'Last Name': 'Khan', 'Student Number': 'N123', 'Tuition Amount': 6505.99}

Process finished with exit code 0
```

Saving object in Files



```
import pickle
people = {1: {'Name': 'Timothy', 'Age': '35', 'Address':'100 Toronto Street, Toronto'},
          2: {'Name': 'Richard', 'Age': '25', 'Address': '100 Main Street, Hamilton'},
          3: {'Name':  'Samuel', 'Age': '28', 'Address': '200 Elm Street, Toronto'}
          }


for p_id, p_info in people.items():
    print("\nPerson ID:", p_id)
    for key in p_info:
        print(key + ':', p_info[key])
######################### open file and save data in file  ##########################################
data = open("object_data_file.dat", "wb")
pickle.dump(people, data)
data.close()
################### open file and read data and display ###############
data = open("object_data_file.dat", "rb")
output = pickle.load(data)
data.close()
print (output)
for p_id, p_info in people.items():
    print("\nPerson ID:", p_id)
    for key in p_info:
        print(key + ':', p_info[key])
```



```
Name: Richard
Age: 25
Address: 100 Main Street, Hamilton

Person ID: 3
Name: Samuel
Age: 28
Address: 200 Elm Street, Toronto
{1: {'Name': 'Timothy', 'Age': '35', 'Address': '100 Toronto Street, Toronto'}, 2: {'Name': 'Richard', 'Age': '25', 'Address': '100 Main Street, Ha

Person ID: 1
Name: Timothy
Age: 35
Address: 100 Toronto Street, Toronto

Person ID: 2
Name: Richard
Age: 25
Address: 100 Main Street, Hamilton

Person ID: 3
Name: Samuel
Age: 28
Address: 200 Elm Street, Toronto
```

Summary of Topics covered

- Creating and traversing dictionary items
- Adding items (key:value) in dictionary
- Removing items (key:value) in dictionary (using pop() and del)
- Clear entire Dictionary using clear() Dictionary API
- Copy Dictionary to another Dictionary (using copy() and dict() constructor)
- Nested Dictionary
- Files
- Modes of opening file
- File write and read operation
- File functions