```python
# A "dictionary" is a "data structure" -- a way of organizing many values into
# one variable. A list is another type of data structure. Data structures
# typically allow us to add, remove, and find values in them. Whereas the values
# of a list are associated with a position (an integer index), values of a
# dictionary are associated with a "key".

# A Python dictionary is denoted with { }, similar to a list denoted with [ ].
# Instead of individual values separated by commas, we give a list of
# "key-value pairs". The "key" of each pair is something easy to remember,
# through which the dictionary can provide access to the associated value.

            # This is a key : this is a value
contacts = {"Neal Terrell" : "562-123-4567",
            "Anthony Giacalone" : "714-987-6543",
            "Mehrdad Aliasgari" : "310-678-9012"}

# To retrieve a value, we "index" the dictionary with the associated key.
print("Neal's phone number is " + contacts["Neal Terrell"])
print("Mehrdad's phone number is " + contacts["Mehrdad Aliasgari"])

# Keys have to match exactly; it is a run-time error (semantic error) to index
# a dictionary with a key that does not exist.

# This line causes an error:
#print(contacts["Neal terrell"])

# We can add new items to the dictionary
contacts["Joshua Hayter"] = "213-555-5555"
contacts["Joshua Hayter"] = "213-555-1111"
print(contacts["Joshua Hayter"])
print()

# We can loop through a dictionary using a for loop, just like a list... but
# what we get out are the keys of the dictionary.
for name in contacts:
    print(name + ": " + contacts[name])


# We can also loop through the dictionary's values only, using the
# .values() method.
for phone_number in contacts.values():
    # Do something with the phone numbers.
    print(phone_number)


```

```python
# Rework the Baseball example to use a dictinoary as the main data structure,
# instead of a list. Each player's tuple (a value) will be associated with
# the player's name (a key).

def read_players(file_name):
    first_line = True
    results = {} # create an empty dictionary
    for line in open(file_name):
        if not first_line:
            # work here
            split = line.split(",")
            name = split[0].strip('"')
            player = (name, split[1].strip('"'), int(split[9].strip('"')),\
                      int(split[11].strip('"')), int(split[19].strip('"')), \
                      float(split[21].strip('"')))

            # insert an association between the player's name
            # and their statistics in the dictionary
            results[name] = player
        else:
            first_line = False
    return results

def main():
    all_players = read_players("baseball_players.csv")
    # Reminder: all_players is now a dictionary, with keys that are player names.
    choice = 0
    while choice != 4:
        print("1. Search for player")
        print("2. Search for team")
        print("3. Find max homeruns")
        print("4. Quit")

        choice = int(input("Enter a choice: "))
        if choice == 1:
            search_for_player(all_players)
        elif choice == 2:
            search_for_team(all_players)
        elif choice == 3:
            find_max_hrs(all_players)

def print_player(player):
    (player_name, team, hr, rbi, sb, avg) = player
    print("{0} ({1}). {2} HR, {3} RBI, {4} SB, {5:0.3f} AVG"\
          .format(player_name, team, hr, rbi, sb, avg))

def find_max_hrs(all_players):
    max_hr = 0
    max_name = ''
    # Since we don't know the names of the players we are examining, we use
    # .values() to loop over all players in the dictionary.
    for player in all_players.values():
        (player_name, team, hr, rbi, sb, avg) = player
        if hr > max_hr:
            max_hr = hr
            max_name = player

    print_player(max_name)

def search_for_team(all_players):
    team_name = input("Enter a team name: ")
    for player in all_players.values():
        (player_name, team, hr, rbi, sb, avg) = player
        if team_name == team:
            print_player(player)


def search_for_player(all_players):
    name = input("Enter a player's name: ")

    # This is the way to search a list for the given player name. It
    # is now unnecessary.
    #for player in all_players:
    #    (player_name, team, hr, rbi, sb, avg) = player
    #    if name == player_name:
    #        print_player(player)
    #        break

    # Instead, we just check to see if the dictionary contains the player
    # name as a key. The "in" keyword serves this purpose.
    if name in all_players:
        player = all_players[name]
        print_player(player)



main()

```

```python
# Rework the Baseball example AGAIN, so that each PLAYER is also a dictionary
# instead of a tuple.

def read_players(file_name):
    first_line = True
    results = {} # create an empty dictionary
    for line in open(file_name):
        if not first_line:
            # work here
            split = line.split(",")

            # Instead of packing the line into a tuple, we create a dictionary
            # for the player's data. Later, instead of having to remember the
            # ORDER of each stat in the player tuple, we just remember the name
            # of the stat in the dictionary. Much easier!
            player = {
                "name" : split[0].strip('"'),
                "team" : split[1].strip('"'),
                "hr" : int(split[9].strip('"')),
                "rbi" : int(split[11].strip('"')),
                "sb" : int(split[19].strip('"')),
                "avg" : float(split[21].strip('"'))
                }

            results[player["name"]] = player
        else:
            first_line = False
    return results

def main():
    all_players = read_players("baseball_players.csv")
    # Reminder: all_players is now a dictionary, with keys that are player names.
    choice = 0
    while choice != 4:
        print("1. Search for player")
        print("2. Search for team")
        print("3. Find max homeruns")
        print("4. Quit")

        choice = int(input("Enter a choice: "))
        if choice == 1:
            search_for_player(all_players)
        elif choice == 2:
            search_for_team(all_players)
        elif choice == 3:
            find_max_hrs(all_players)

def print_player(player):
    print("{0} ({1}). {2} HR, {3} RBI, {4} SB, {5:0.3f} AVG"\
          .format(player["name"], player["team"], player["hr"],\
                  player["rbi"], player["sb"], player["avg"]))

```

```python
def find_max_hrs(all_players):
    max_hr = -1

    # Since we don't know the names of the players we are examining, we use
    # .values() to loop over all players in the dictionary.
    for player in all_players.values():
        if player["hr"] > max_hr:
            max_hr = player["hr"]
            max_name = player["name"]


    print("The most homeruns was {0} by {1}".format(max_hr, max_name))

def search_for_team(all_players):
    team_name = input("Enter a team name: ")
    for player in all_players.values():
        if player["team"] == team_name:
            print_player(player)


def search_for_player(all_players):
    name = input("Enter a player's name: ")
    # We just check to see if the dictionary contains the player
    # name as a key. The "in" keyword serves this purpose.
    if name in all_players:
        player = all_players[name]
        print_player(player)


main()

```