## Learning Aims

- Describe the structure of two-dimensional data structures and give examples of their use
- Create and use 2D data structure
- Describe the record structure and explain what it is used for
- Design a record structure
- Process every record in 2D structure
- Search for an item in a 2D list
- Decompose tasks and use subprograms

## Two-Dimensional Data Structure

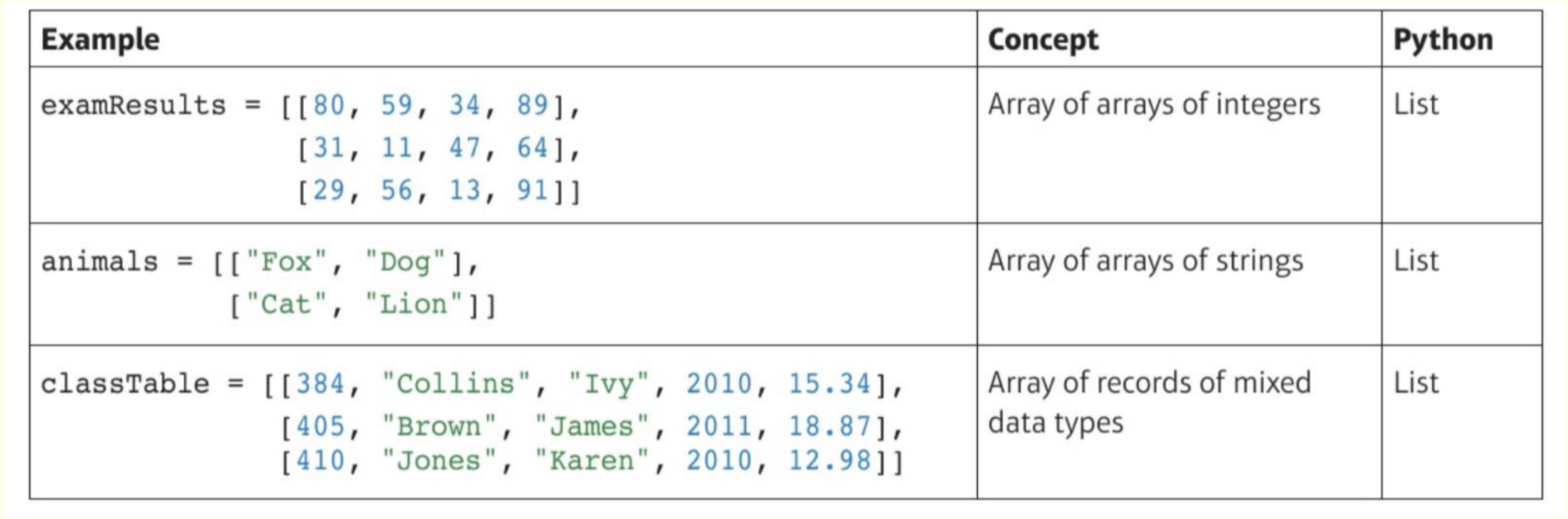
**Example of 1D in python:**
**studentRecord = [“Boris”, 3]**

**Example of a 2D in python:**
**classTable = [[“Boris”, 3], [“Marek”,46], [“Jane”, 75], [“Ryan”, 65]]**

- To store a table of data, we can use a two-dimensional data structure
- In python, it is stored as lists within a list

| | name | score1 |
|---|---|---|
| Index | 0 | 1 |
| 0 | Boris | 3 |
| 1 | Marek | 46 |
| 2 | Jane | 75 |
| 3 | Ryan | 65 |

## Examples of 2D data structures



| Example | Concept | Python |
|---|---|---|
| `examResults = [[80, 59, 34, 89],`<br>`[31, 11, 47, 64],`<br>`[29, 56, 13, 91]]` | Array of arrays of integers | List |
| `animals = [["Fox", "Dog"],`<br>`["Cat", "Lion"]]` | Array of arrays of strings | List |
| `classTable = [[384, "Collins", "Ivy", 2010, 15.34],`<br>`[405, "Brown", "James", 2011, 18.87],`<br>`[410, "Jones", "Karen", 2010, 12.98]]` | Array of records of mixed data types | List |

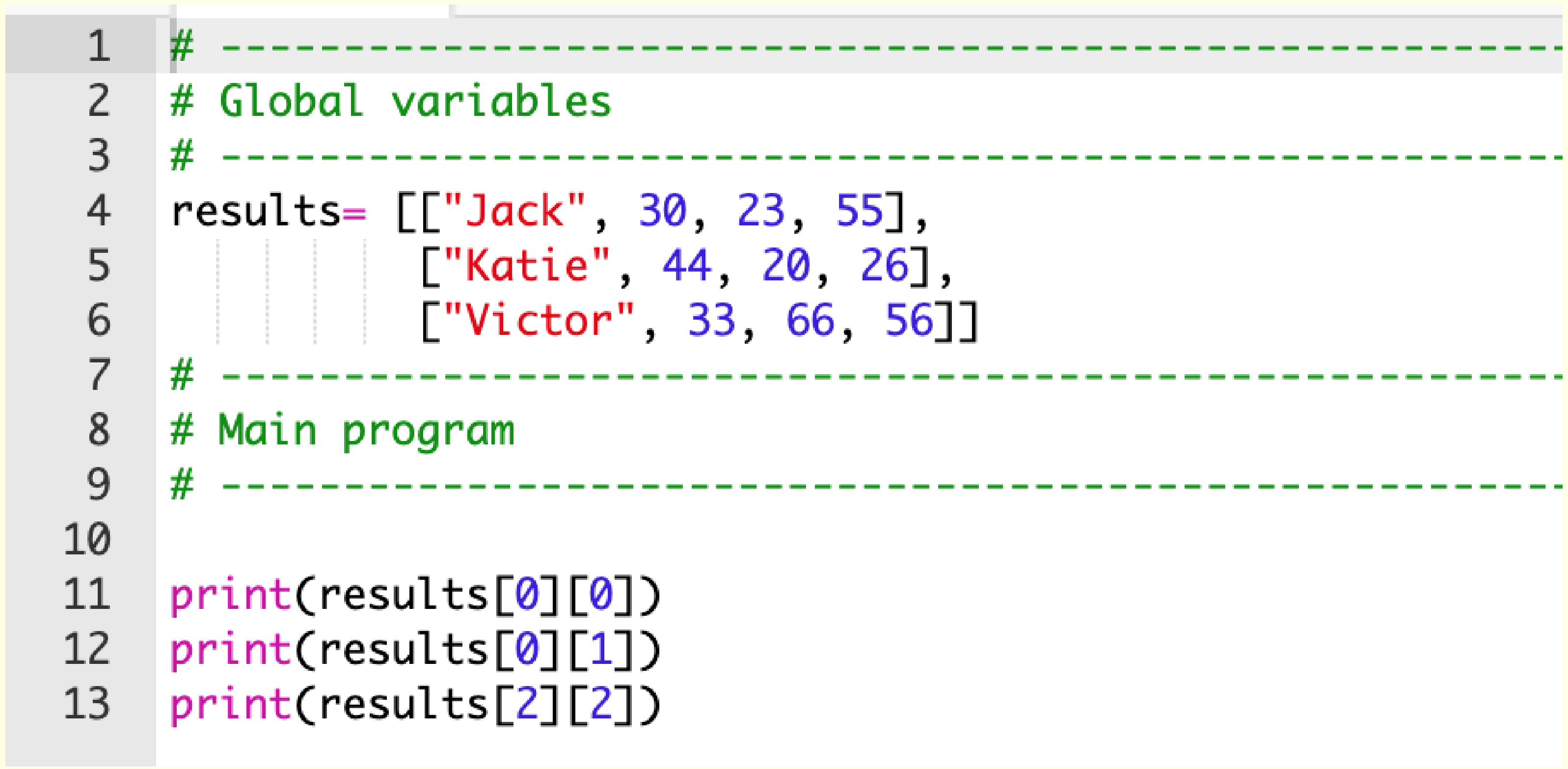
## Accessing date in 2D List

**classTable = [[“Boris”, 3], [“Marek”,46], [“Jane”, 75], [“Ryan”, 65]]**

classTable[row][column]
classTable[2][1] would be 75
classTable[0][1] would be 3
classTable[3][0] would be “Ryan”
classTable[4][0] would be Error!

| | name | score1 |
|---|---|---|
| Index | 0 | 1 |
| 0 | Boris | 3 |
| 1 | Marek | 46 |
| 2 | Jane | 75 |
| 3 | Ryan | 65 |

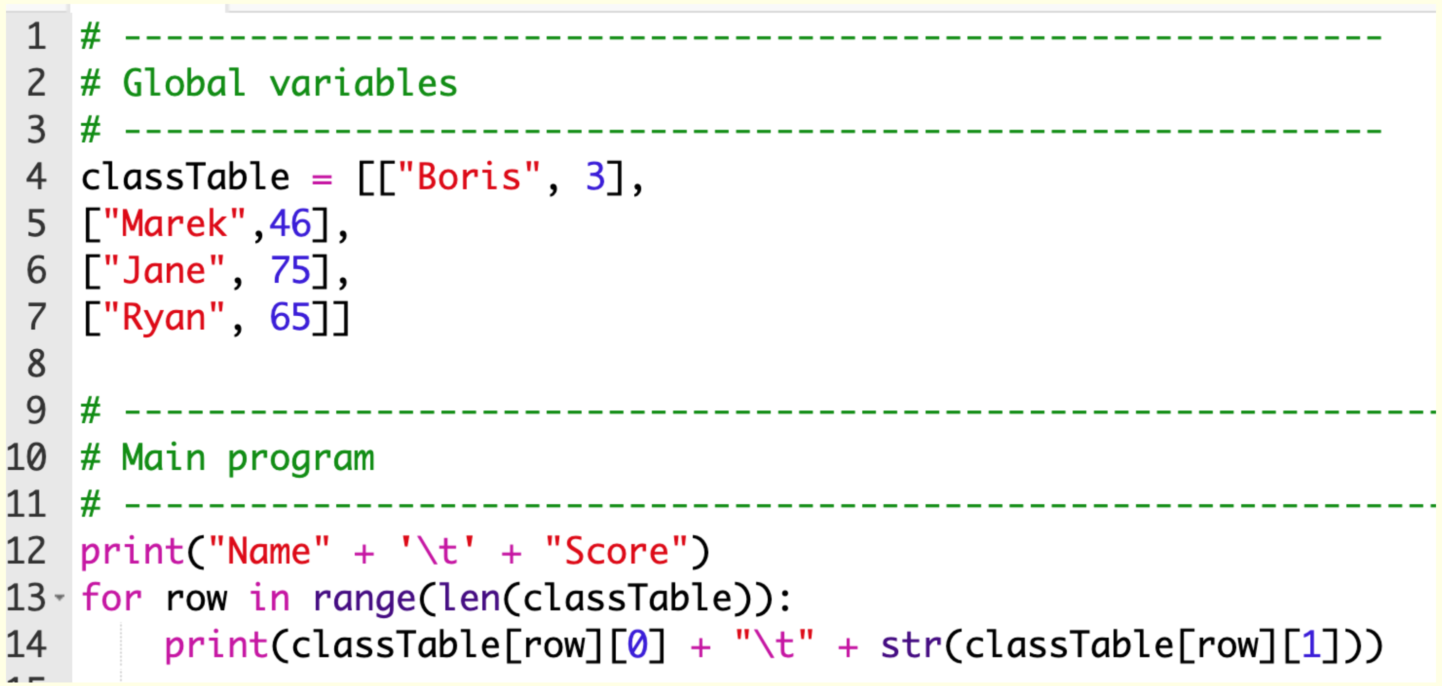
## Worked Example



```
# ------------------------------------------------------
# Global variables
# ------------------------------------------------------
results= [["Jack", 30, 23, 55],
          ["Katie", 44, 20, 26],
          ["Victor", 33, 66, 56]]
# ------------------------------------------------------
# Main program
# ------------------------------------------------------

print(results[0][0])
print(results[0][1])
print(results[2][2])
```

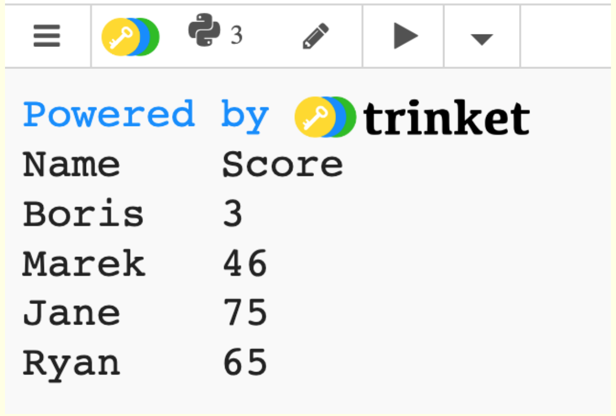
What will this program output?

## Worked Example: In this example a for loop in range() is used to print items in each row of the table.



```
# ----------------------------------------------------------------
# Global variables
# ----------------------------------------------------------------
classTable = [["Boris", 3],
["Marek",46],
["Jane", 75],
["Ryan", 65]]

# -----------------------------------------------------------------
# Main program
# -----------------------------------------------------------------
print("Name" + '\t' + "Score")
for row in range(len(classTable)):
    print(classTable[row][0] + "\t" + str(classTable[row][1]))
```

|  | name | score1 |
|---|---|---|
| Index | 0 | 1 |
| 0 | Boris | 3 |
| 1 | Marek | 46 |
| 2 | Jane | 75 |
| 3 | Ryan | 65 |



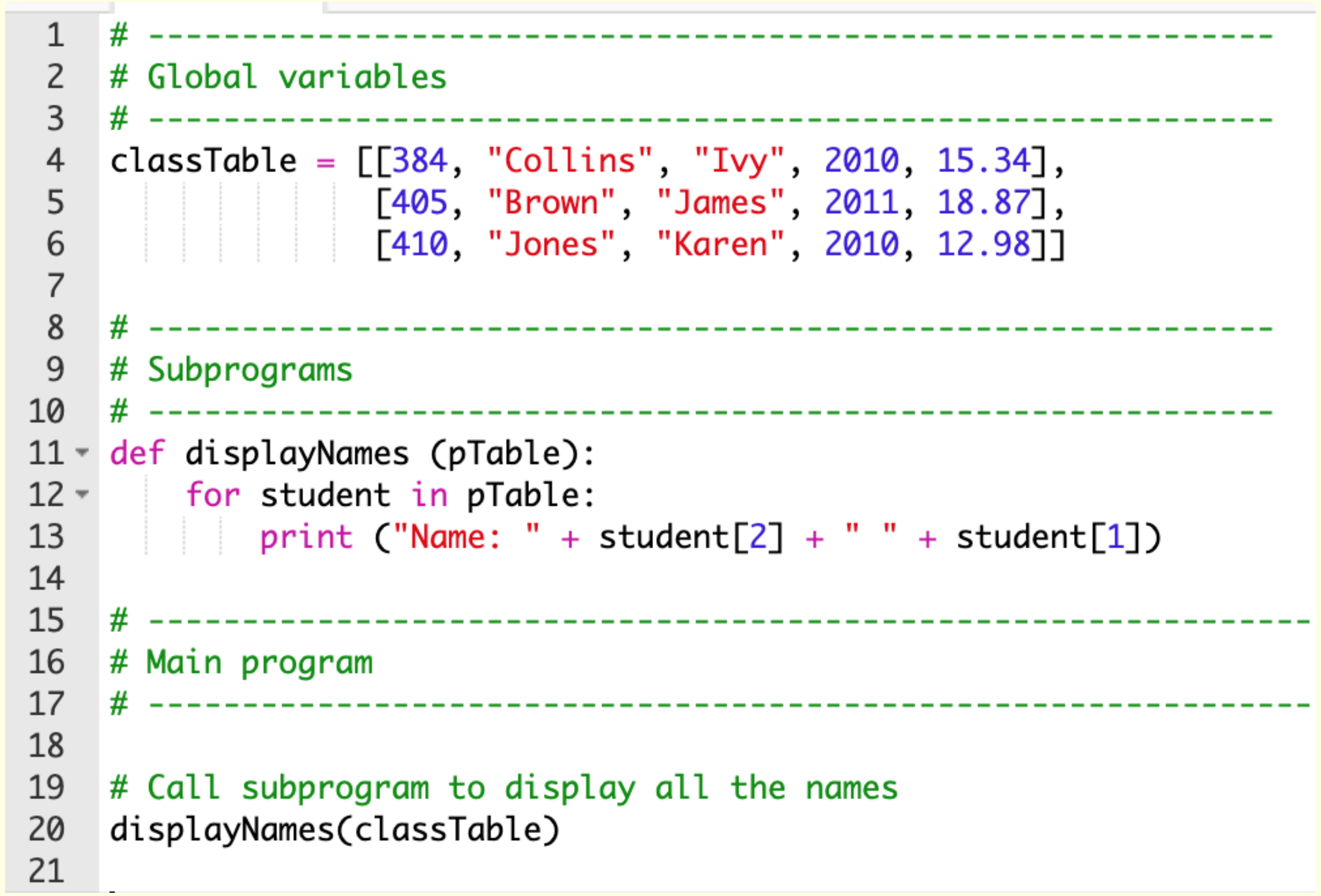
```
Name     Score
Boris    3
Marek    46
Jane     75
Ryan     65
```

## Worked Example

In this example, each student in the class is printed out in a first name, last name order.

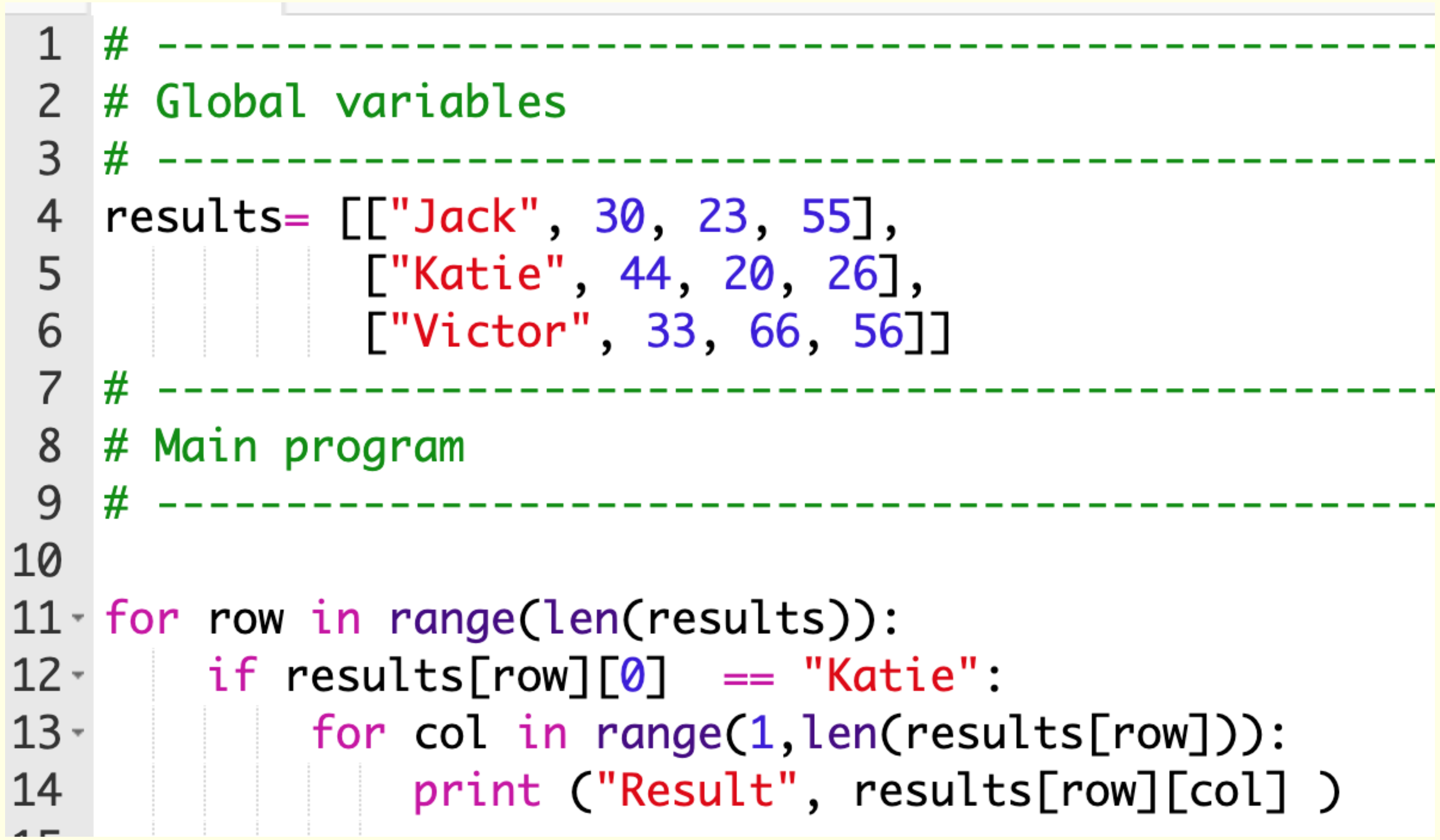
To do this, a subprogram displayNames(), has been created. It takes a single parameter, classTable.

The subprogram processes every record.

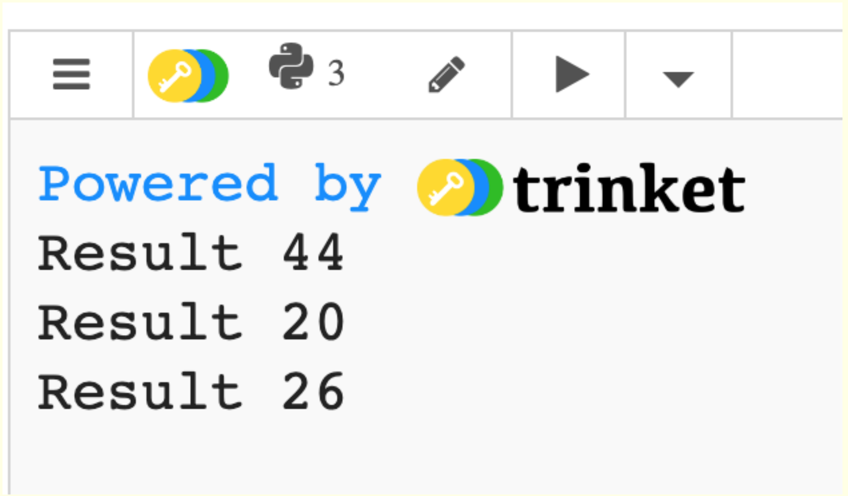


```
# ------------------------------------------------------------
# Global variables
# ------------------------------------------------------------
classTable = [[384, "Collins", "Ivy", 2010, 15.34],
              [405, "Brown", "James", 2011, 18.87],
              [410, "Jones", "Karen", 2010, 12.98]]

# ------------------------------------------------------------
# Subprograms
# ------------------------------------------------------------
def displayNames (pTable):
    for student in pTable:
        print ("Name: " + student[2] + " " + student[1])

# --------------------------------------------------------------
# Main program
# --------------------------------------------------------------

# Call subprogram to display all the names
displayNames(classTable)

```

## Worked Example: Searching for an item in 2D list using a for loop



```
# ---------------------------------------------------------
# Global variables
# ---------------------------------------------------------
results= [["Jack", 30, 23, 55],
          ["Katie", 44, 20, 26],
          ["Victor", 33, 66, 56]]
# ---------------------------------------------------------
# Main program
# ---------------------------------------------------------

for row in range(len(results)):
    if results[row][0]  == "Katie":
        for col in range(1,len(results[row])):
            print ("Result", results[row][col] )
```



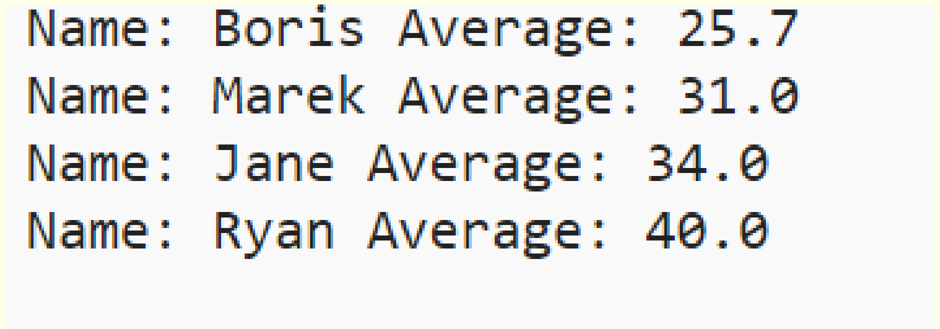
```
Result 44
Result 20
Result 26
```

## Worked Example Average scores in python

results = [[“Boris”, 34,46], [“Marek”,46,50], [“Jane”, 75,30], [“Ryan”, 65,58]]

|  | name | score1 | score2 |
|---|---|---|---|
| Index | 0 | 1 | 2 |
| 0 | Boris | 34 | 46 |
| 1 | Marek | 46 | 50 |
| 2 | Jane | 75 | 30 |
| 3 | Ryan | 65 | 58 |

```
for row in range(len(results)):
    total = 0
    print ("Name:” results[row][0],end=" ")
    #Looping through the columns from index 1
    for col in range (1, len(results[row])):
        total = total + results[row][col]

    average = round(total/ len(results[row])
    print("Average:", average)
```

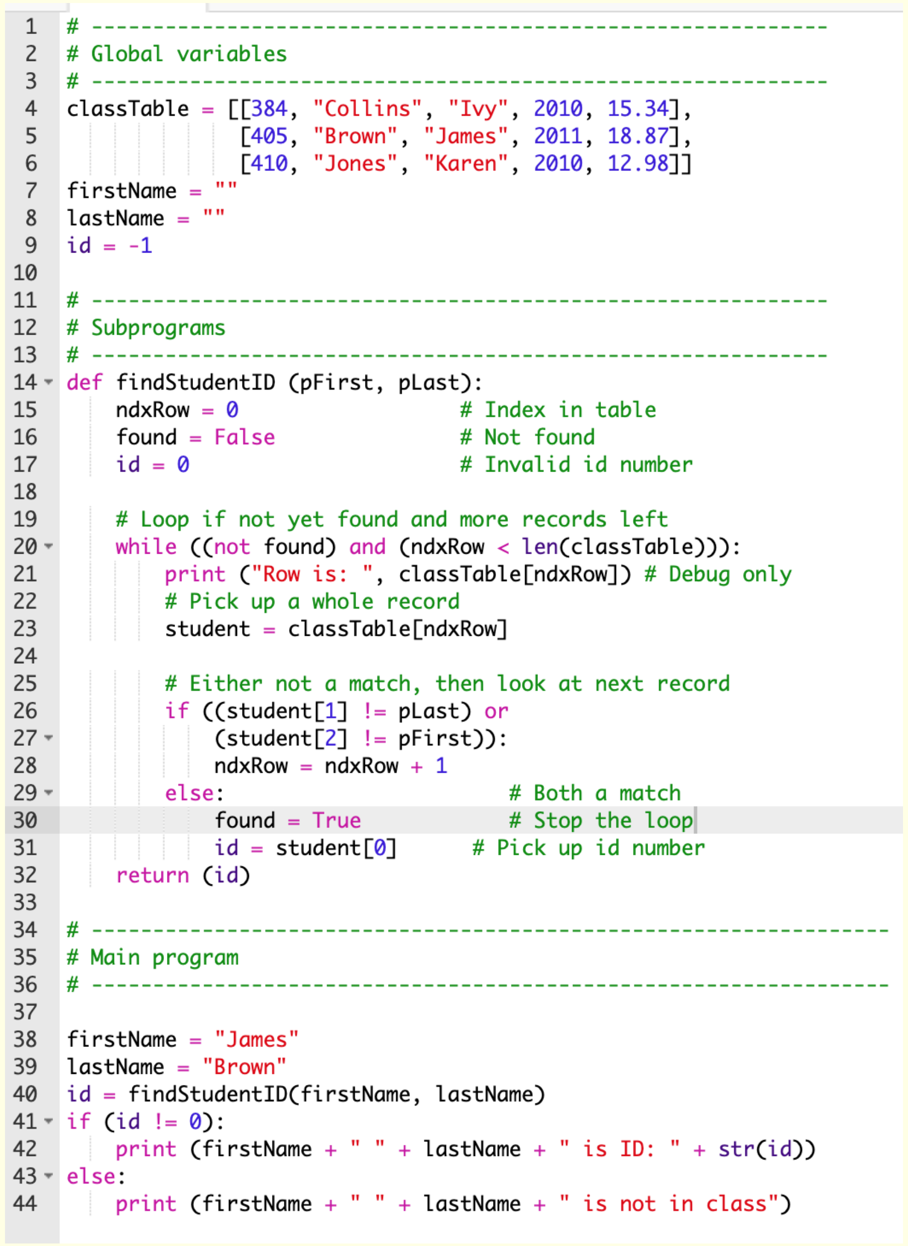


```
Name: Boris Average: 25.7
Name: Marek Average: 31.0
Name: Jane Average: 34.0
Name: Ryan Average: 40.0
```

# Worked example: searching a 2D list using a while loop

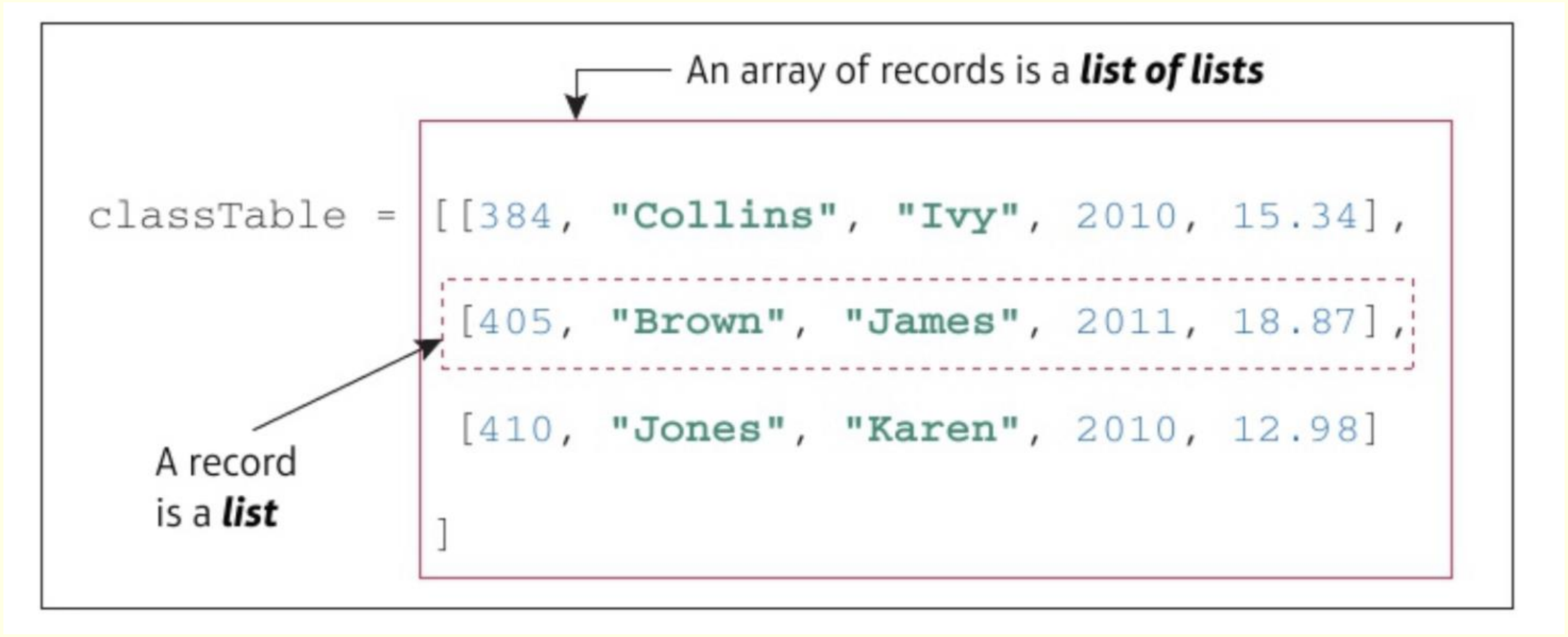
In this example, we are given the first name and last name of the student to find the ID number. A subprogram findStudentID() has been created.

It takes 2 parameters, pFirst and pLast.



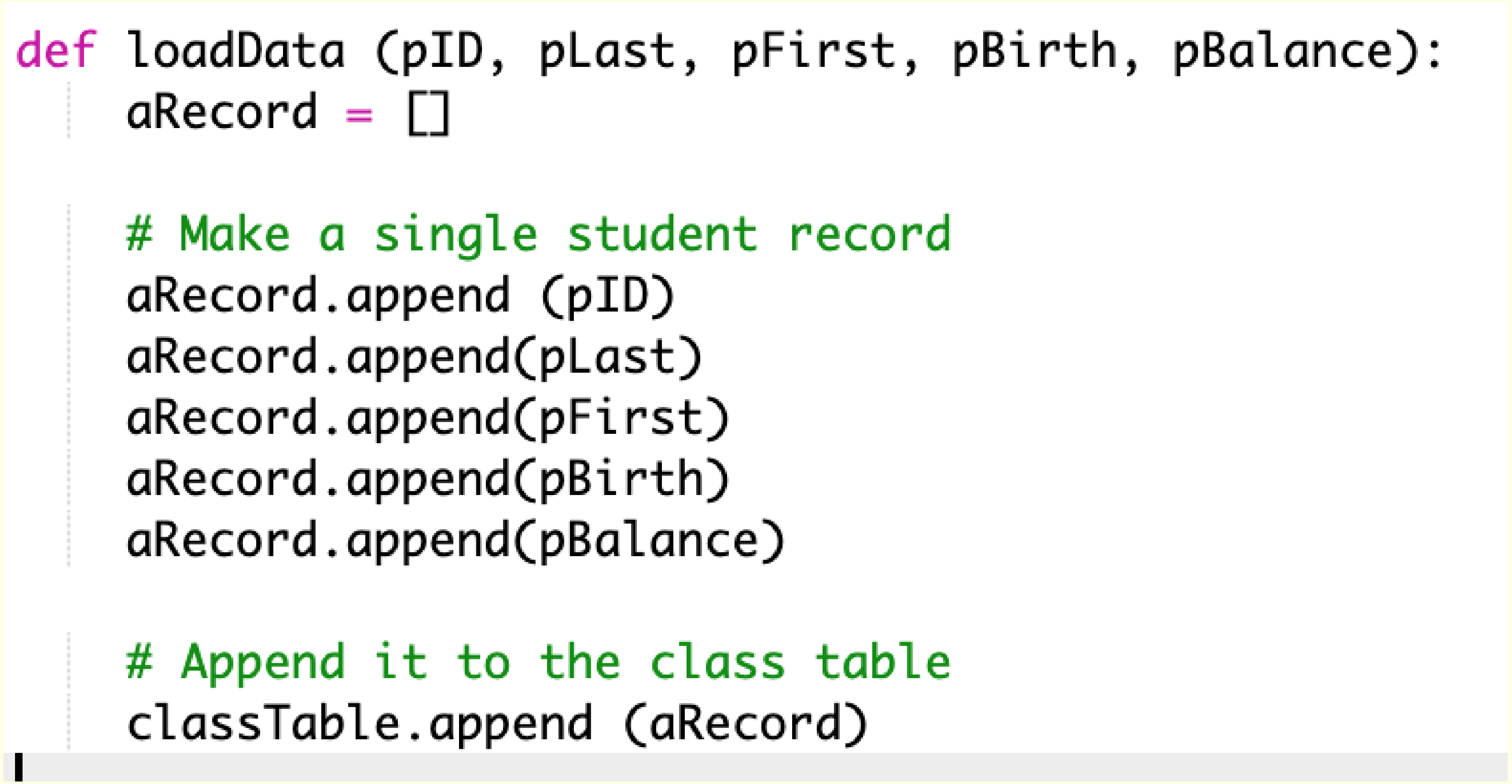
```python
# --------------------------------------------------------------
# Global variables
# --------------------------------------------------------------
classTable = [[384, "Collins", "Ivy", 2010, 15.34],
              [405, "Brown", "James", 2011, 18.87],
              [410, "Jones", "Karen", 2010, 12.98]]
firstName = ""
lastName = ""
id = -1

# --------------------------------------------------------------
# Subprograms
# --------------------------------------------------------------
def findStudentID (pFirst, pLast):
    ndxRow = 0                  # Index in table
    found = False               # Not found
    id = 0                      # Invalid id number

    # Loop if not yet found and more records left
    while ((not found) and (ndxRow < len(classTable))):
        print ("Row is: ", classTable[ndxRow]) # Debug only
        # Pick up a whole record
        student = classTable[ndxRow]

        # Either not a match, then look at next record
        if ((student[1] != pLast) or
            (student[2] != pFirst)):
            ndxRow = ndxRow + 1
        else:                       # Both a match
            found = True            # Stop the loop
            id = student[0]      # Pick up id number
    return (id)

# -------------------------------------------------------------------
# Main program
# -------------------------------------------------------------------

firstName = "James"
lastName = "Brown"
id = findStudentID(firstName, lastName)
if (id != 0):
    print (firstName + " " + lastName + " is ID: " + str(id))
else:
    print (firstName + " " + lastName + " is not in class")
```

## Appending, inserting and deleting records



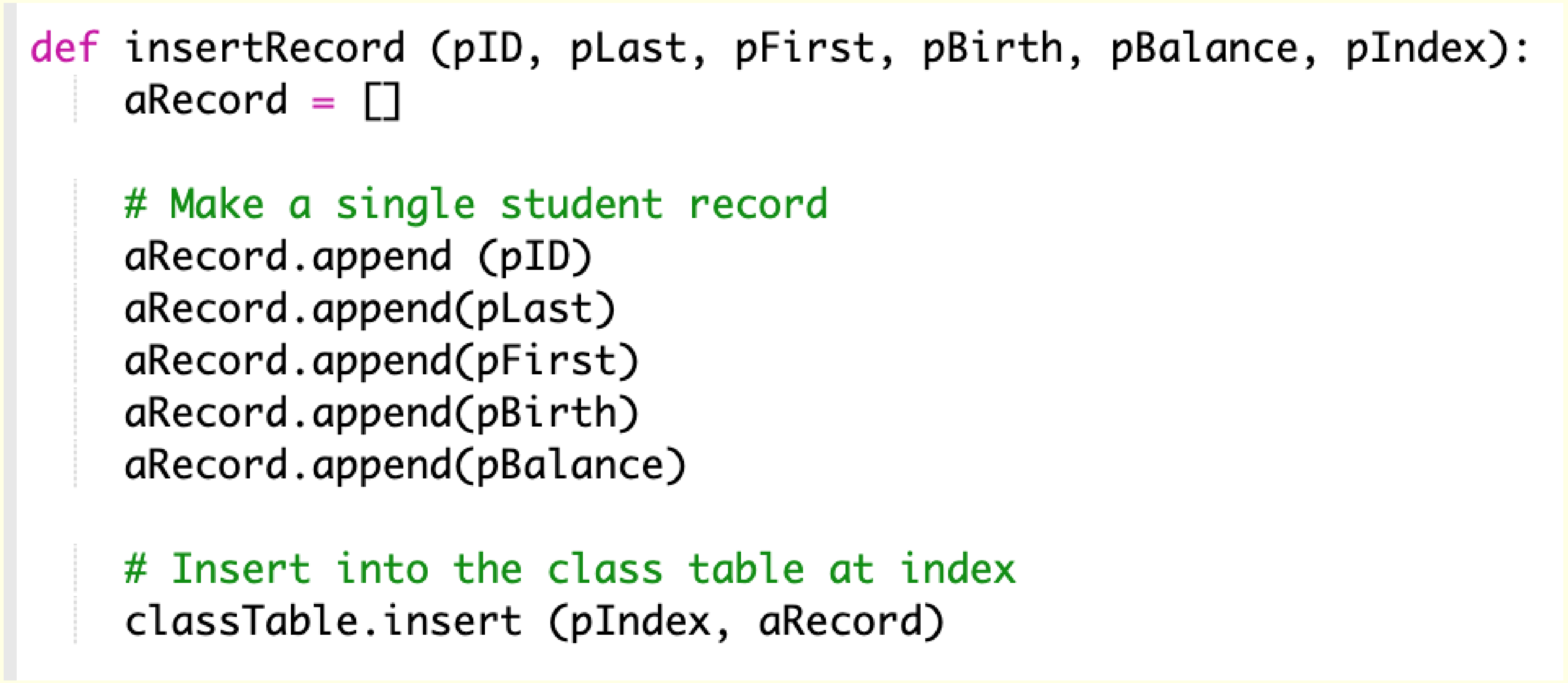
```
classTable = [[384, "Collins", "Ivy", 2010, 15.34],
              [405, "Brown", "James", 2011, 18.87],
              [410, "Jones", "Karen", 2010, 12.98]
             ]
```

## Append a new row to the 2D array



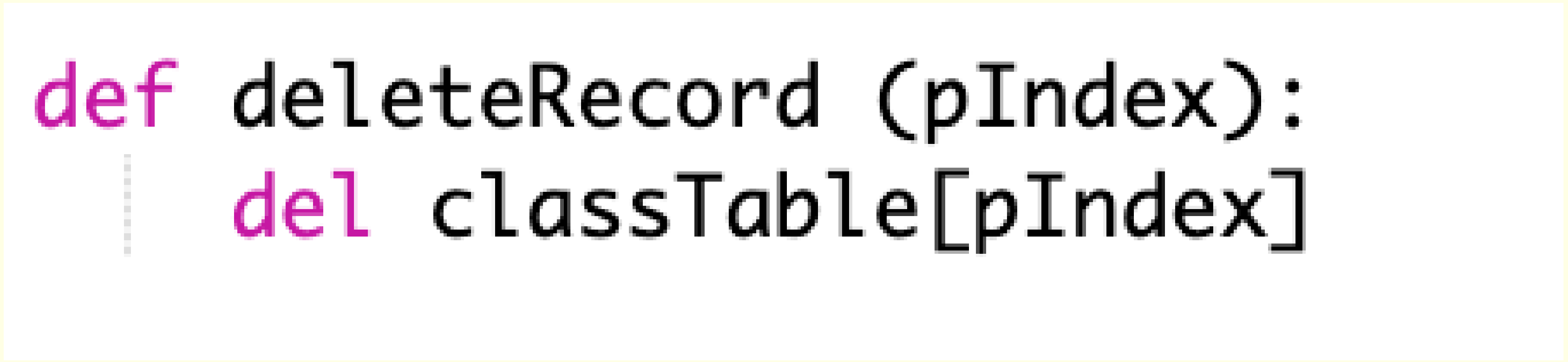
```python
def loadData (pID, pLast, pFirst, pBirth, pBalance):
    aRecord = []

    # Make a single student record
    aRecord.append (pID)
    aRecord.append(pLast)
    aRecord.append(pFirst)
    aRecord.append(pBirth)
    aRecord.append(pBalance)

    # Append it to the class table
    classTable.append (aRecord)
```

## Inserting a record



```
def insertRecord (pID, pLast, pFirst, pBirth, pBalance, pIndex):
    aRecord = []

    # Make a single student record
    aRecord.append (pID)
    aRecord.append(pLast)
    aRecord.append(pFirst)
    aRecord.append(pBirth)
    aRecord.append(pBalance)

    # Insert into the class table at index
    classTable.insert (pIndex, aRecord)
```

## Deleting a record



```
def deleteRecord (pIndex):
    del classTable[pIndex]
```