AWK Programming Language

A data-driven text processing tool

AWK is text-processing tool used for pattern scanning and analysis of text files.

Data Driven

You provide awk with rules for what to do with the data, rather than explicitly detailing how to process each line of data.

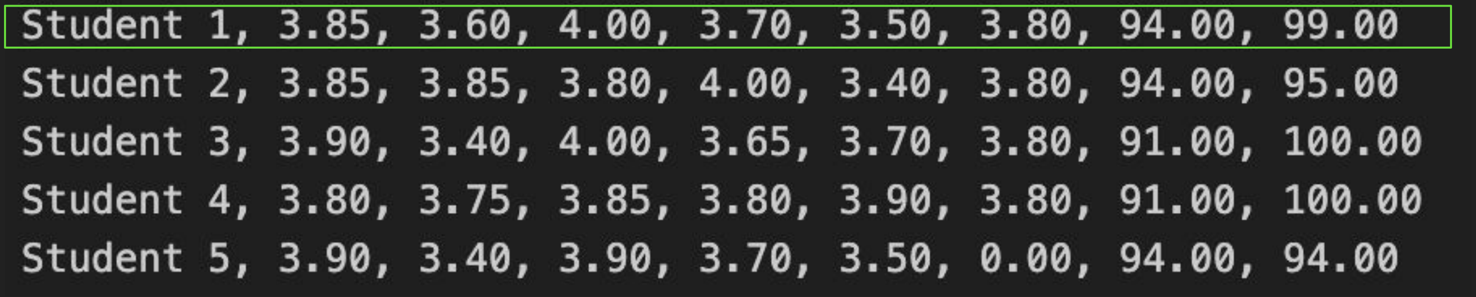
Field Based

Awk can easily work with structured data because it treats input data as a series of fields rather than just a continuous stream of characters.

Pattern Processing

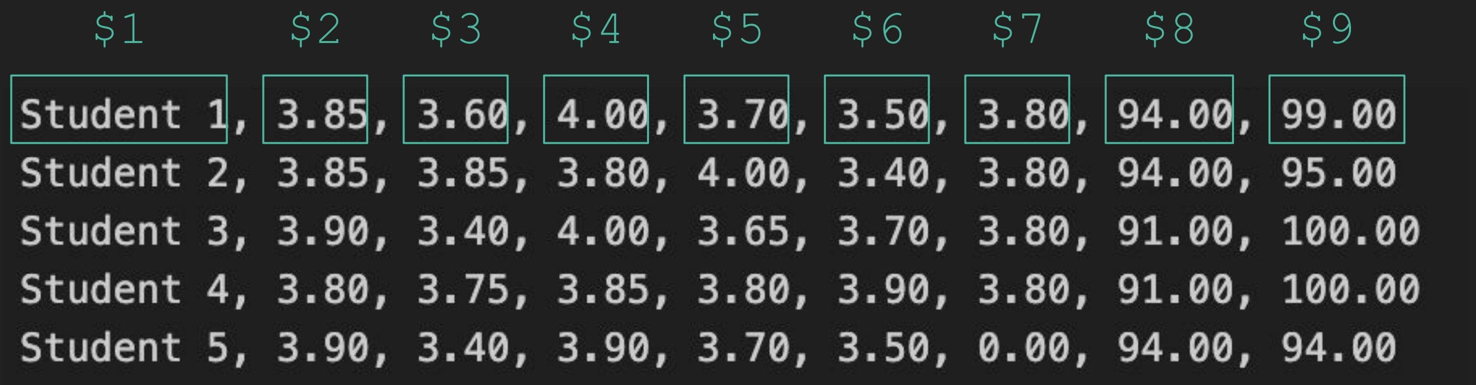
Awk operates by matching patterns in the input data and then executing actions based on those patterns.

Awk views its input data as a series of records. Generally each line in a text file is seen as a new record. Each record contains a series of fields. A field is a component of a record defined by a field separator.



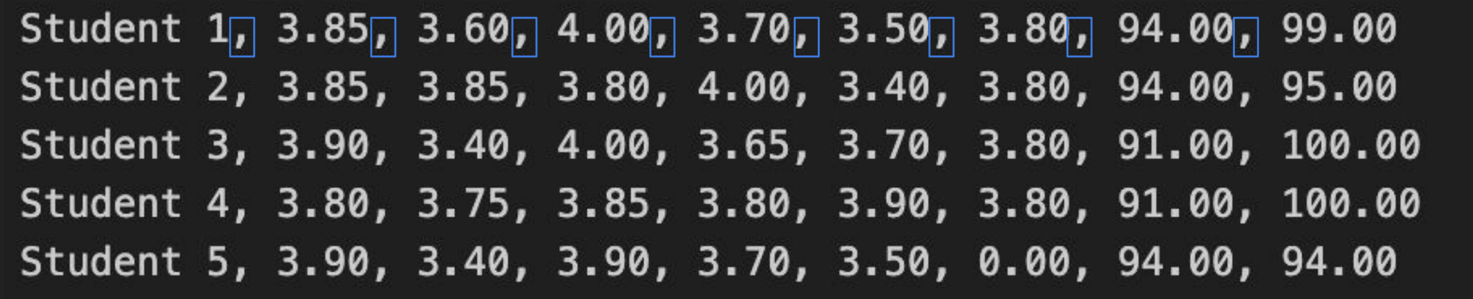
```
Student 1, 3.85, 3.60, 4.00, 3.70, 3.50, 3.80, 94.00, 99.00
Student 2, 3.85, 3.85, 3.80, 4.00, 3.40, 3.80, 94.00, 95.00
Student 3, 3.90, 3.40, 4.00, 3.65, 3.70, 3.80, 91.00, 100.00
Student 4, 3.80, 3.75, 3.85, 3.80, 3.90, 3.80, 91.00, 100.00
Student 5, 3.90, 3.40, 3.90, 3.70, 3.50, 0.00, 94.00, 94.00
```

Awk views its input data as a series of records. Generally each line in a text file is seen as a new record. Each record contains a series of fields. A field is a component of a record defined by a field separator.



```
Student 1, 3.85, 3.60, 4.00, 3.70, 3.50, 3.80, 94.00, 99.00
Student 2, 3.85, 3.85, 3.80, 4.00, 3.40, 3.80, 94.00, 95.00
Student 3, 3.90, 3.40, 4.00, 3.65, 3.70, 3.80, 91.00, 100.00
Student 4, 3.80, 3.75, 3.85, 3.80, 3.90, 3.80, 91.00, 100.00
Student 5, 3.90, 3.40, 3.90, 3.70, 3.50, 0.00, 94.00, 94.00
```

Awk views its input data as a series of records. Generally each line in a text file is seen as a new record. Each record contains a series of fields. A field is a component of a record defined by a field separator.

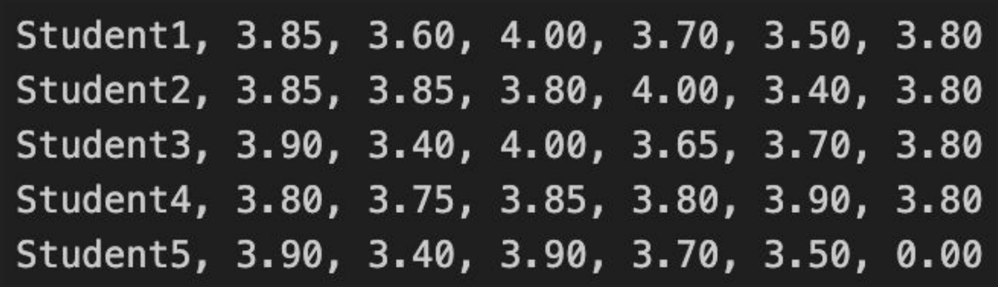


```
Student 1, 3.85, 3.60, 4.00, 3.70, 3.50, 3.80, 94.00, 99.00
Student 2, 3.85, 3.85, 3.80, 4.00, 3.40, 3.80, 94.00, 95.00
Student 3, 3.90, 3.40, 4.00, 3.65, 3.70, 3.80, 91.00, 100.00
Student 4, 3.80, 3.75, 3.85, 3.80, 3.90, 3.80, 91.00, 100.00
Student 5, 3.90, 3.40, 3.90, 3.70, 3.50, 0.00, 94.00, 94.00
```

Patterns and Actions

★ Pattern: Specifies conditions or rules for selecting lines of input data.
  ○ Can be regular expressions, string literals, numeric conditions, or combinations of these.

★ Action: Specifies what AWK should do when a line of input matches the pattern.
  ○ Can include printing the line, performing calculations, setting variables, or executing user-defined functions.

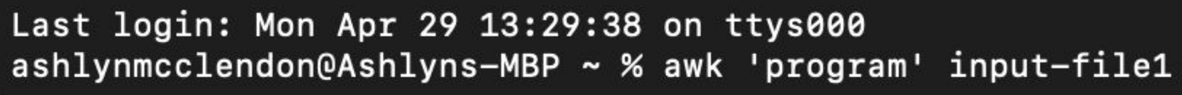


```
Student1, 3.85, 3.60, 4.00, 3.70, 3.50, 3.80
Student2, 3.85, 3.85, 3.80, 4.00, 3.40, 3.80
Student3, 3.90, 3.40, 4.00, 3.65, 3.70, 3.80
Student4, 3.80, 3.75, 3.85, 3.80, 3.90, 3.80
Student5, 3.90, 3.40, 3.90, 3.70, 3.50, 0.00
```

```
pattern { action }
/4.00/ { print $1 }
```

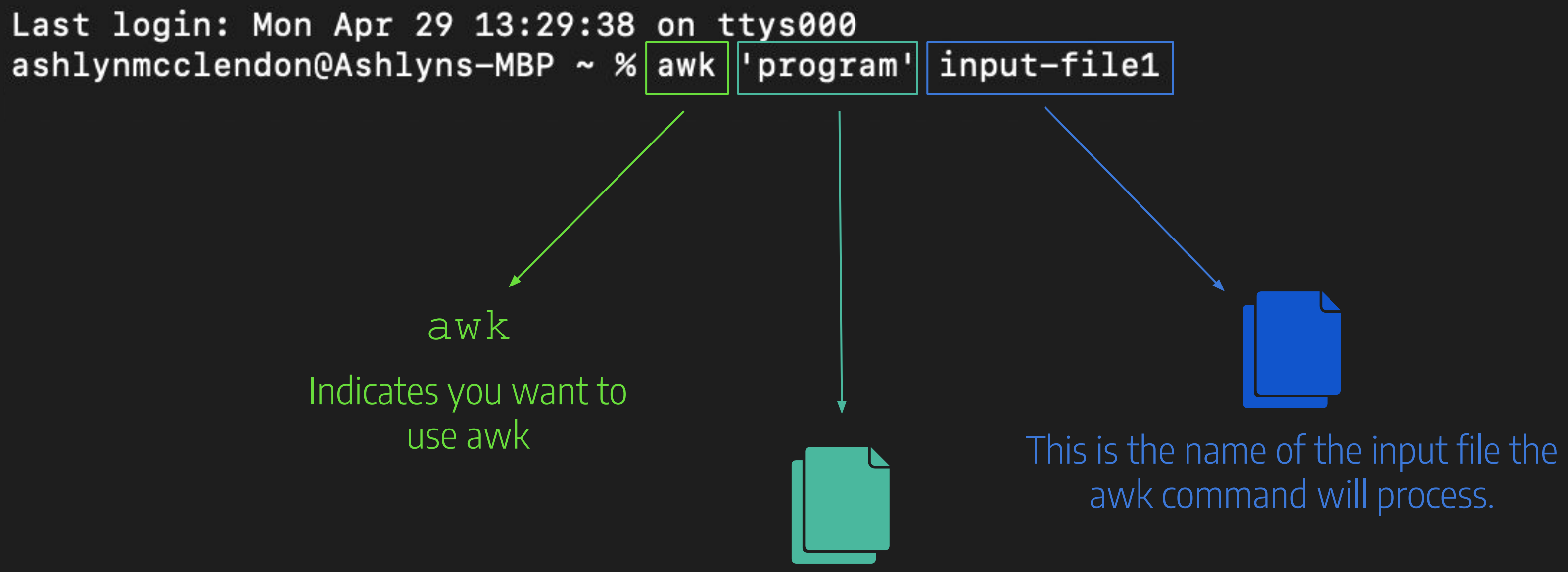
```
Student1
Student2
Student3
```

awk can be run from the command line:



```
Last login: Mon Apr 29 13:29:38 on ttys000
ashlynmcclendon@Ashlyns-MBP ~ % awk 'program' input-file1
```

awk can be run from the command line:



```
Last login: Mon Apr 29 13:29:38 on ttys000
ashlynmcclendon@Ashlyns-MBP ~ % awk 'program' input-file1
```

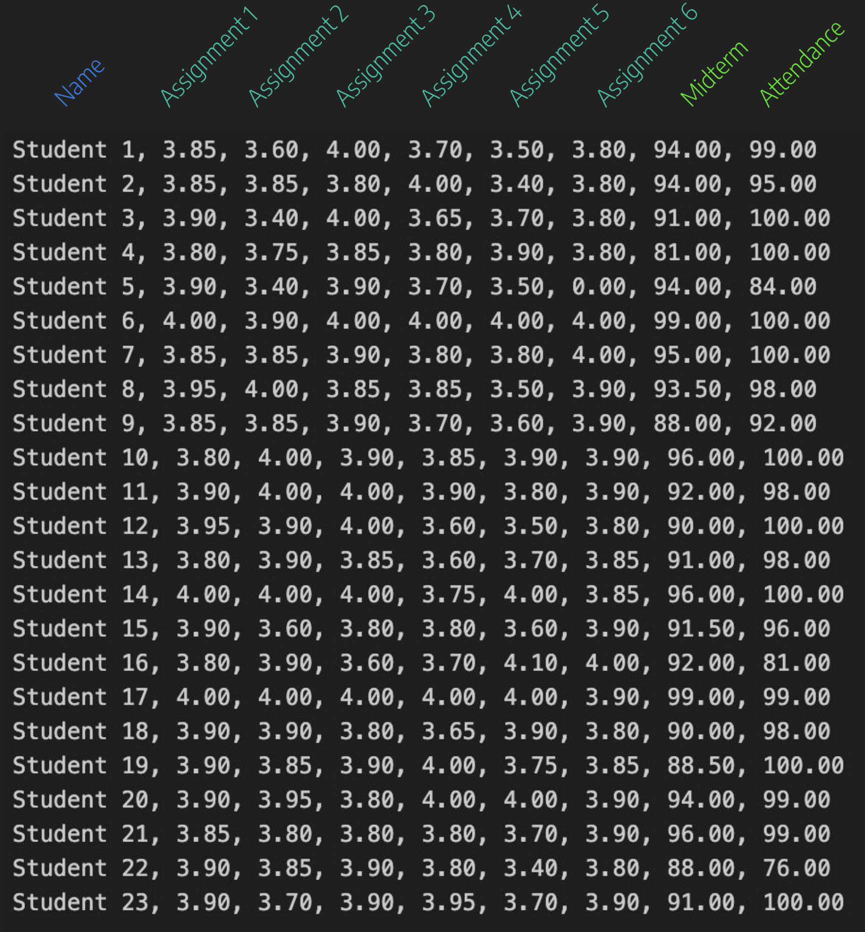
awk

Indicates you want to use awk

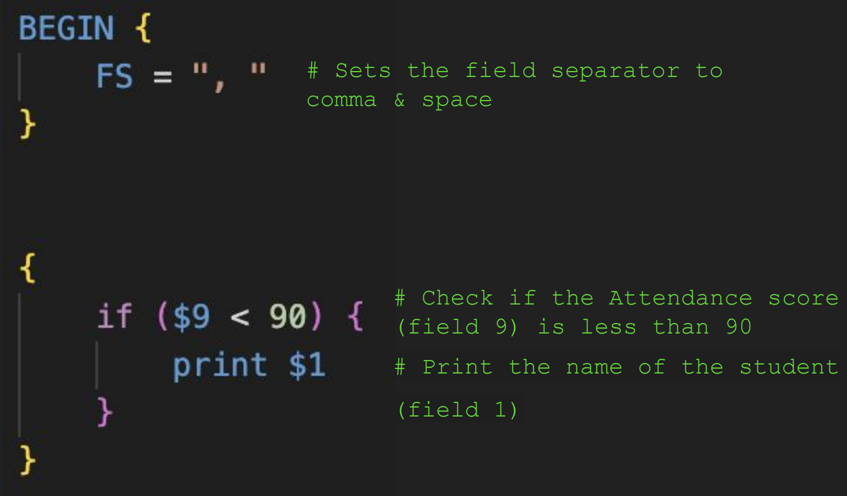
This is the name of the input file the awk command will process.

This is the .awk program that will be executed on the input file(s)

Who has attendance grades below a 90?



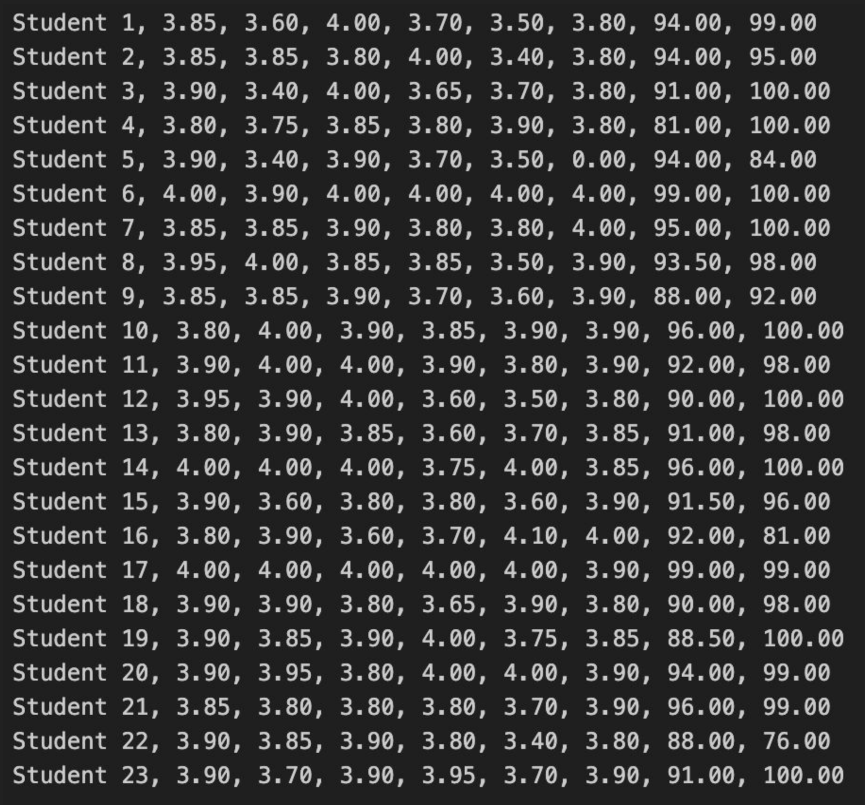
```
Student 1, 3.85, 3.60, 4.00, 3.70, 3.50, 3.80, 94.00, 99.00
Student 2, 3.85, 3.85, 3.80, 4.00, 3.40, 3.80, 94.00, 95.00
Student 3, 3.90, 3.40, 4.00, 3.65, 3.70, 3.80, 91.00, 100.00
Student 4, 3.80, 3.75, 3.85, 3.80, 3.90, 3.80, 81.00, 100.00
Student 5, 3.90, 3.40, 3.90, 3.70, 3.50, 0.00, 94.00, 84.00
Student 6, 4.00, 3.90, 4.00, 4.00, 4.00, 4.00, 99.00, 100.00
Student 7, 3.85, 3.85, 3.90, 3.80, 3.80, 4.00, 95.00, 100.00
Student 8, 3.95, 4.00, 3.85, 3.85, 3.50, 3.90, 93.50, 98.00
Student 9, 3.85, 3.85, 3.90, 3.70, 3.60, 3.90, 88.00, 92.00
Student 10, 3.80, 4.00, 3.90, 3.85, 3.90, 3.90, 96.00, 100.00
Student 11, 3.90, 4.00, 4.00, 3.90, 3.80, 3.90, 92.00, 98.00
Student 12, 3.95, 3.90, 4.00, 3.60, 3.50, 3.80, 90.00, 100.00
Student 13, 3.80, 3.90, 3.85, 3.60, 3.70, 3.85, 91.00, 98.00
Student 14, 4.00, 4.00, 4.00, 3.75, 4.00, 3.85, 96.00, 100.00
Student 15, 3.90, 3.60, 3.80, 3.80, 3.60, 3.90, 91.50, 96.00
Student 16, 3.80, 3.90, 3.60, 3.70, 4.10, 4.00, 92.00, 81.00
Student 17, 4.00, 4.00, 4.00, 4.00, 4.00, 3.90, 99.00, 99.00
Student 18, 3.90, 3.90, 3.80, 3.65, 3.90, 3.80, 90.00, 98.00
Student 19, 3.90, 3.85, 3.90, 4.00, 3.75, 3.85, 88.50, 100.00
Student 20, 3.90, 3.95, 3.80, 4.00, 4.00, 3.90, 94.00, 99.00
Student 21, 3.85, 3.80, 3.80, 3.80, 3.70, 3.90, 96.00, 99.00
Student 22, 3.90, 3.85, 3.90, 3.80, 3.40, 3.80, 88.00, 76.00
Student 23, 3.90, 3.70, 3.90, 3.95, 3.70, 3.90, 91.00, 100.00
```



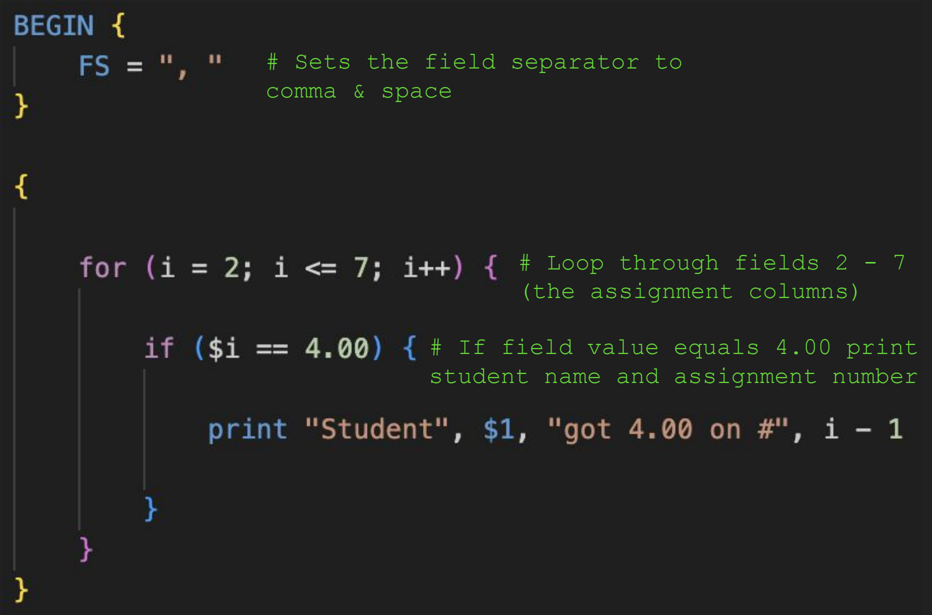
```
BEGIN {
    FS = ", "    # Sets the field separator to
                 comma & space
}


{
    if ($9 < 90) {    # Check if the Attendance score
                      (field 9) is less than 90
        print $1      # Print the name of the student

                      (field 1)
    }
}
```

Who has gotten a perfect score (4.00) on any assignment?

Name Assignment 1 Assignment 2 Assignment 3 Assignment 4 Assignment 5 Assignment 6 Midterm Attendance



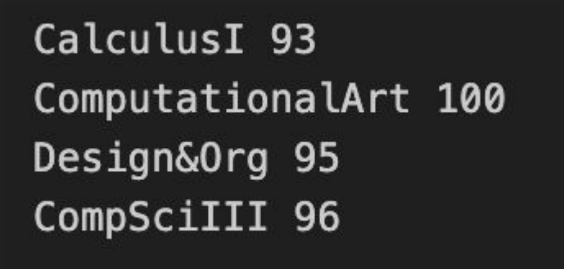
```
Student 1, 3.85, 3.60, 4.00, 3.70, 3.50, 3.80, 94.00, 99.00
Student 2, 3.85, 3.85, 3.80, 4.00, 3.40, 3.80, 94.00, 95.00
Student 3, 3.90, 3.40, 4.00, 3.65, 3.70, 3.80, 91.00, 100.00
Student 4, 3.80, 3.75, 3.85, 3.80, 3.90, 3.80, 81.00, 100.00
Student 5, 3.90, 3.40, 3.90, 3.70, 3.50, 0.00, 94.00, 84.00
Student 6, 4.00, 3.90, 4.00, 4.00, 4.00, 4.00, 99.00, 100.00
Student 7, 3.85, 3.85, 3.90, 3.80, 3.80, 4.00, 95.00, 100.00
Student 8, 3.95, 4.00, 3.85, 3.85, 3.50, 3.90, 93.50, 98.00
Student 9, 3.85, 3.85, 3.90, 3.70, 3.60, 3.90, 88.00, 92.00
Student 10, 3.80, 4.00, 3.90, 3.85, 3.90, 3.90, 96.00, 100.00
Student 11, 3.90, 4.00, 4.00, 3.90, 3.80, 3.90, 92.00, 98.00
Student 12, 3.95, 3.90, 4.00, 3.60, 3.50, 3.80, 90.00, 100.00
Student 13, 3.80, 3.90, 3.85, 3.60, 3.70, 3.85, 91.00, 98.00
Student 14, 4.00, 4.00, 4.00, 3.75, 4.00, 3.85, 96.00, 100.00
Student 15, 3.90, 3.60, 3.80, 3.80, 3.60, 3.90, 91.50, 96.00
Student 16, 3.80, 3.90, 3.60, 3.70, 4.10, 4.00, 92.00, 81.00
Student 17, 4.00, 4.00, 4.00, 4.00, 4.00, 3.90, 99.00, 99.00
Student 18, 3.90, 3.90, 3.80, 3.65, 3.90, 3.80, 90.00, 98.00
Student 19, 3.90, 3.85, 3.90, 4.00, 3.75, 3.85, 88.50, 100.00
Student 20, 3.90, 3.95, 3.80, 4.00, 4.00, 3.90, 94.00, 99.00
Student 21, 3.85, 3.80, 3.80, 3.80, 3.70, 3.90, 96.00, 99.00
Student 22, 3.90, 3.85, 3.90, 3.80, 3.40, 3.80, 88.00, 76.00
Student 23, 3.90, 3.70, 3.90, 3.95, 3.70, 3.90, 91.00, 100.00
```



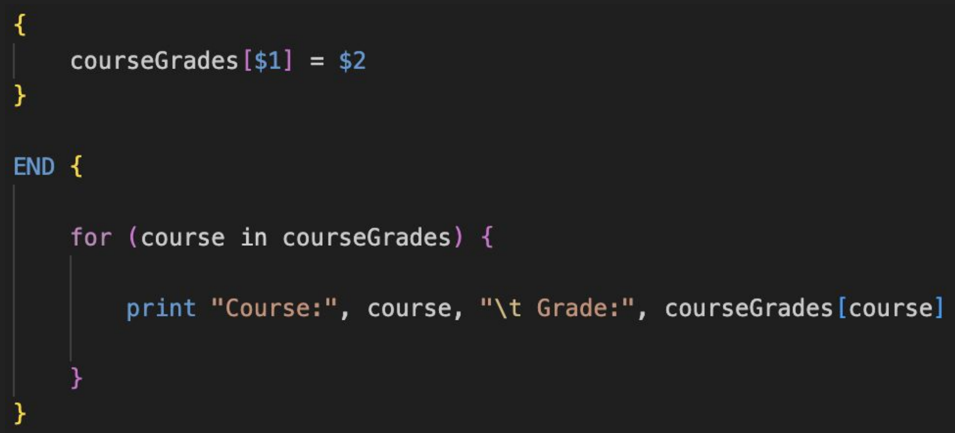
```
BEGIN {
    FS = ", "    # Sets the field separator to
                 comma & space
}

{

    for (i = 2; i <= 7; i++) { # Loop through fields 2 - 7
                                 (the assignment columns)

        if ($i == 4.00) { # If field value equals 4.00 print
                            student name and assignment number

            print "Student", $1, "got 4.00 on #", i - 1

        }
    }
}
```

Associative Arrays

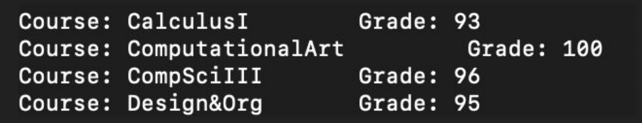
- ★ Keys
  - ○ Unique identifiers used to access values in the array (can be strings or numbers).
- ★ Values
  - ○ Data associated with each key.



```
CalculusI 93
ComputationalArt 100
Design&Org 95
CompSciIII 96
```



```
{
    courseGrades[$1] = $2
}

END {
    for (course in courseGrades) {
        print "Course:", course, "\t Grade:", courseGrades[course]
    }
}
```



```
Course: CalculusI        Grade: 93
Course: ComputationalArt         Grade: 100
Course: CompSciIII       Grade: 96
Course: Design&Org       Grade: 95
```

Thank you! Any Questions?