- ✓ Patterns & Actions
- ✓ Fields · NF · NR
- ✓ Variables & Operators
- ✓ if · loops
- ✓ Associative Arrays
- ✓ printf & Reports
- ✓ Functions
- ✓ CSV · Logs · Pipelines

AWK

FULL COURSE

12 Lessons · Beginner to Advanced

Free · English & Dari

Learn it all @MyWebUniversity

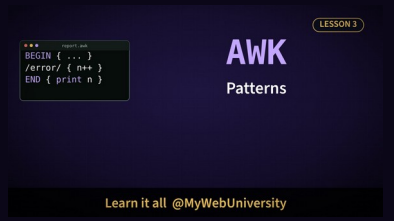
The 12 Lessons



1. Welcome to AWK



2. Fields and Records



3. Patterns, BEGIN and END



4. Variables and Operators



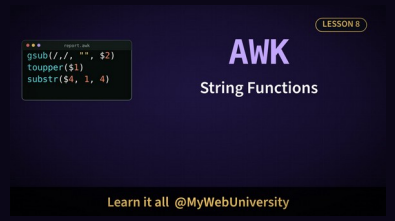
5. Control Flow



6. printf and Reports



7. Associative Arrays



8. String Functions



9. Functions and Script Files



10. Multiple Files and I/O



11. Real-World AWK



12. gawk Power Features and Next Steps



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 1

Welcome to AWK

What AWK is, pattern and action, and your first one-liners

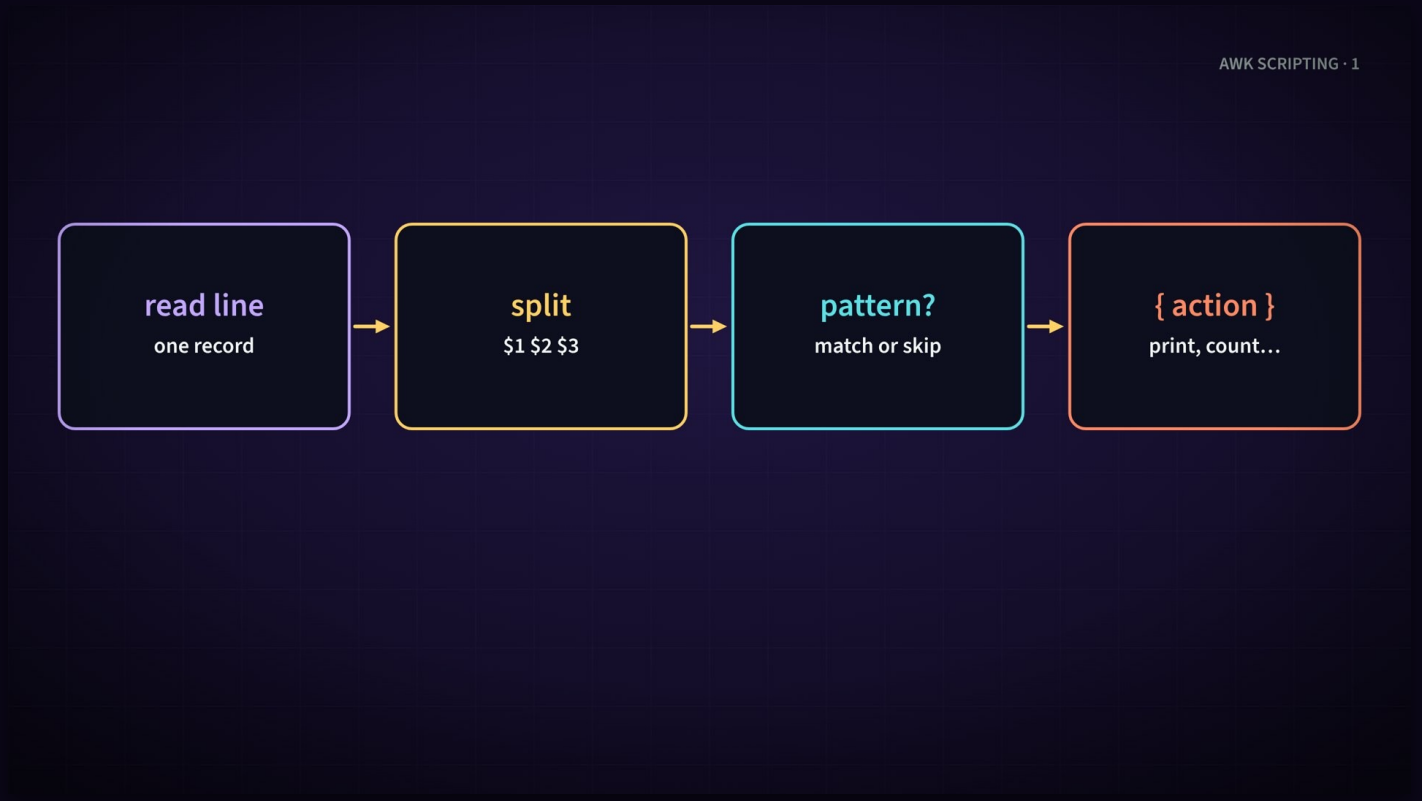
What Is AWK?



- Text in — logs, CSV, reports
- Fields — $1, $2, $3 …
- Answers out — totals, counts, reports

1.01

Pattern { Action }



- read line — one record
- split — $1 $2 $3
- pattern? — match or skip

1.02

Your First One-Liners

terminal

```
$ cat scores.txt
Mariam 95 Kabul
Ahmad 88 Herat
Zahra 72 Mazar

$ awk '{ print $1 }' scores.txt
Mariam
Ahmad
Zahra

$ awk '$2 > 80' scores.txt
Mariam 95 Kabul
Ahmad 88 Herat
```

- The simplest program prints everything: awk, then print in braces, then the file name. …
- A pattern alone works too: awk slash error slash prints only the lines that contain the …

1.03

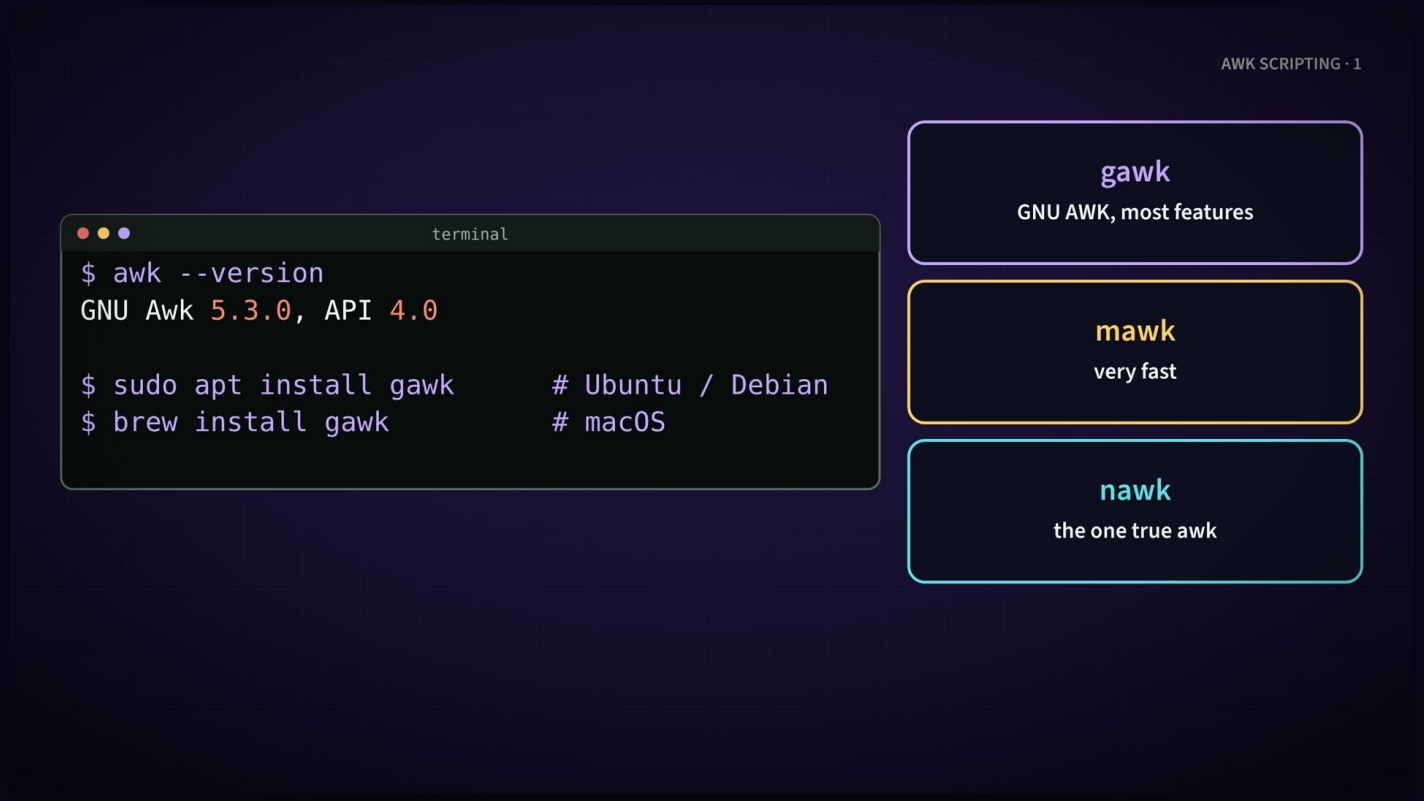
Where AWK Shines



- System administrators use AWK to read logs, data people use it to clean CSV files, and …
- A task that needs twenty lines in another language is often one line of AWK. That is why it …

1.04

Versions and Setup



- gawk — GNU AWK, most features
- mawk — very fast
- nawk — the one true awk

1.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 2

Fields and Records

$0, $1, NF, NR, the -F option, FS and OFS

Records and Fields



- Each line is a record. AWK splits the record into fields at spaces and tabs, and numbers …
- Dollar zero is the whole line. So print dollar three, dollar one prints the third field, …

2.01

NF and NR

terminal

```
$ awk '{ print NR, $0 }' scores.txt
1 Mariam 95 Kabul
2 Ahmad 88 Herat

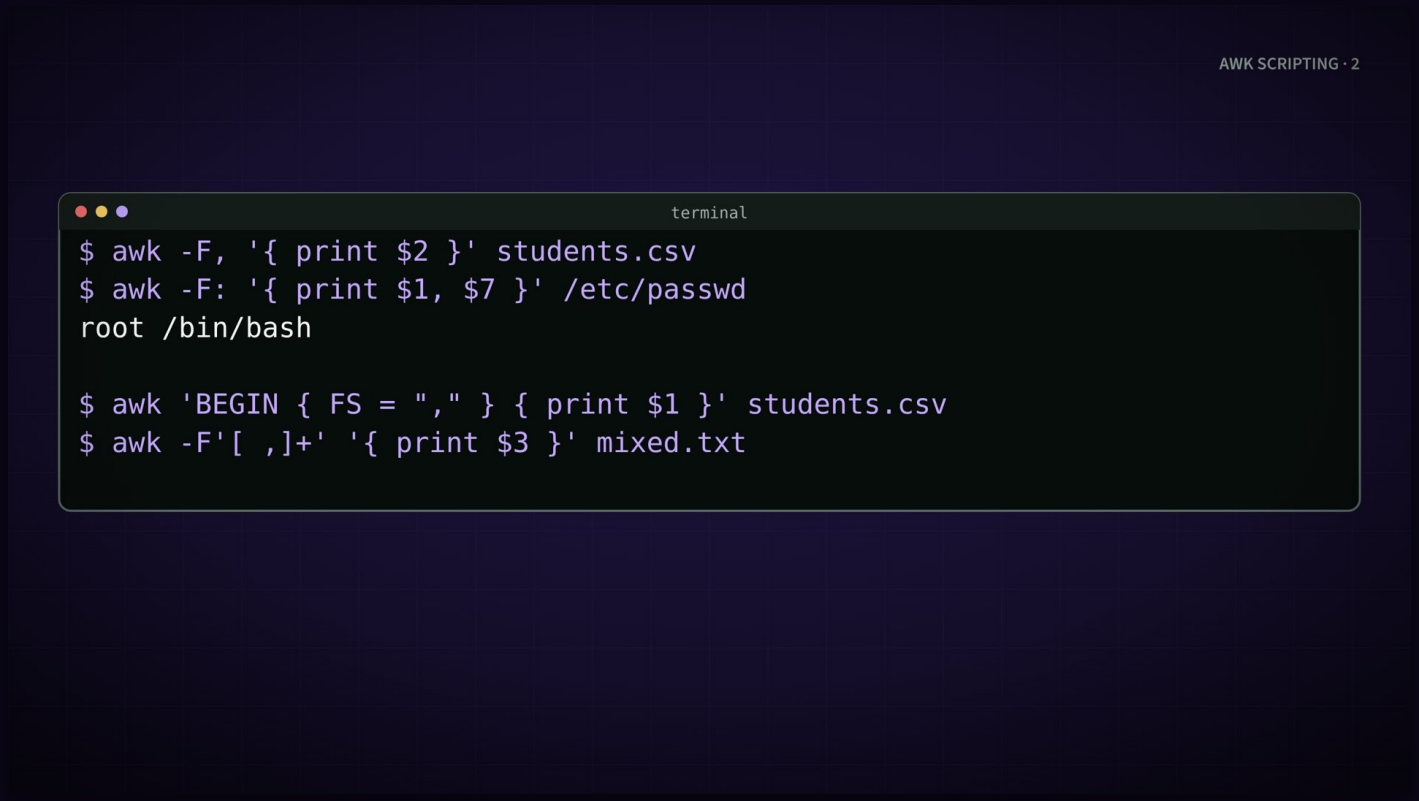
$ awk '{ print $NF }' scores.txt     # last field
Kabul
Herat

$ awk 'NR > 1' data.csv              # skip header
```

- NF is the number of fields in the current line, so dollar NF is always the last field. NR …
- Print NR, dollar zero numbers every line. And NR greater than one skips a header line in a …

2.02

Choosing the Separator: -F



- Many files are not separated by spaces. Use dash F to set the field separator: dash F comma …
- Inside a program, set FS in a BEGIN block. FS can even be a regular expression, like one or …

2.03

Output Separator: OFS

to_csv.awk

```
BEGIN { OFS = "," }
{ $1 = $1; print }        # rebuild $0 with commas

# input : Mariam 95 Kabul
# output: Mariam,95,Kabul
```

- The comma in a print statement inserts OFS, the output field separator, which is a space by …
- Set OFS to a comma to turn a space-separated file into CSV. One trick: assign dollar one to …

2.04

Changing Fields

grade.awk

```
{
    $2 = $2 * 1.1                    # 10% bonus
    $4 = ($2 >= 60) ? "pass" : "fail"  # new field
    print
}
# Mariam 104.5 Kabul pass
```

- You can assign to any field. Dollar two equals dollar two times 1.1 gives everyone a ten …
- You can even create new fields: dollar four equals pass or fail. AWK adds the field and …

2.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 3

Patterns, BEGIN and END

Regular expressions, comparisons, ranges and special blocks

Regular Expression Patterns

regex.awk

```
/Kabul/            { print }          # anywhere in the line
$3 ~ /^K/          { print $1 }       # field 3 starts with K
$1 !~ /^#/         { print }          # skip comment lines
/^[0-9]+$/         { count++ }        # lines that are only digits
```

- A pattern between slashes is a regular expression. It matches anywhere in the line, so …
- To test a single field, use the tilde operator: dollar three tilde slash caret K slash …

3.01

Comparison Patterns

Pattern	Selects
`$2 >= 90`	scores of 90 or more
`NF == 0`	empty lines
`$3 == "Herat" && $2 > 80`	Herat students over 80
`NR % 2 == 0`	every even line
`!/^#/`	lines that are not comments

- Any expression can be a pattern. Dollar two greater than or equal to ninety selects top …
- Combine patterns with and, or and not: double ampersand, double bar and the exclamation mark.

3.02

Range Patterns

terminal

```
$ awk '/^\[database\]/, /^$/' app.ini
[database]
host = localhost
port = 5432

$ awk 'NR==10, NR==20' big.log     # lines 10 to 20
```

- Two patterns separated by a comma make a range. AWK starts printing at the first match and …
- This is perfect for pulling one section out of a config file or a log, from a START marker …

3.03

BEGIN and END

average.awk

```
BEGIN { print "Name   Score"; print "-----  -----" }
      { print $1, $2; total += $2 }
END   { print "Average:", total / NR }

# Average: 85
```

- BEGIN runs once before AWK reads any input, so it is the place to set FS, OFS and print …
- END runs once after the last line, so it is the place to print totals and averages. …

3.04

Missing Pattern or Action



- If you leave out the pattern, the action runs on every line. If you leave out the action, …
- So a bare pattern is a filter, and a bare action is a transformer. Most AWK programs mix both.

3.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 4

Variables and Operators

Numbers, strings, arithmetic, concatenation and built-in variables

Variables Need No Declaration

vars.awk

```
{ total += $2; count++ }          # created on first use
END {
    avg = total / count
    name = "Average"
    print name ": " avg            # concatenation
}
```

- AWK variables appear the moment you use them. They start as empty string and zero at the …
- Each value is both a number and a string. AWK decides from how you use it: plus means a …

4.01

Arithmetic

Expression	Result	Meaning
`7 / 2`	`3.5`	always floating point
`7 % 2`	`1`	remainder
`2 ^ 10`	`1024`	power
`int(7 / 2)`	`3`	drop the fraction
`x += 5`	`x + 5`	shortcut

- AWK has plus, minus, times, divide, percent for remainder and caret for power. All numbers …
- Use int to drop the fraction, and the shortcuts plus equals, plus plus and minus minus just …

4.02

Strings and Concatenation

strings.awk

```
{
    full = $1 " " $2              # join with a space
    id   = "STU-" NR               # STU-1, STU-2 ...
    if ($1 == "Mariam") print "found:", full
    if ($1 < "M") print $1, "comes before M"
}
```

- Strings go in double quotes. To join strings, simply write them next to each other: first …
- Comparison works on strings too. Dollar one equals equals Mariam compares text, and the …

4.03

Built-in Variables

Variable	Meaning
NR · FNR	record number · per file
NF	number of fields
FS · OFS	input · output field separator
RS · ORS	input · output record separator
FILENAME · ENVIRON	current file · environment

- You already know NR, NF, FS and OFS. RS is the record separator, so setting RS to an empty …
- FILENAME holds the current file name, FNR counts lines within each file, and ENVIRON gives …

4.04

Passing Values with -v

terminal

```
$ min=80
$ awk -v min="$min" '$2 >= min { print $1 }' scores.txt
Mariam
Ahmad

$ awk -v city=Herat '$3 == city' scores.txt
```

- Never paste shell variables inside the AWK program. Use dash v name equals value instead, …
- This keeps quoting safe and makes your one-liners reusable, for example a minimum score …

4.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 5

Control Flow

if, else, while, for, next and exit

if and else

grade.awk

```
{
    if ($2 >= 90)      grade = "A"
    else if ($2 >= 75) grade = "B"
    else if ($2 >= 60) grade = "C"
    else               grade = "F"
    print $1, grade, ($2 >= 60 ? "pass" : "fail")
}
```

- Inside an action, AWK has the same if and else as C. Put the condition in parentheses and …
- For short choices, the ternary operator is handy: condition, question mark, value if true, …

5.01

while and for

rowsum.awk

```
{
    sum = 0
    for (i = 1; i <= NF; i++)
        sum += $i
    print "line", NR, "sum =", sum
}

# 10 20 30  ->  line 1 sum = 60
```

- Most of the time AWK loops over lines for you, but sometimes you loop over fields. A for …
- while and do-while work too. Here we add up all the numbers on each line, however many …

5.02

next and exit

next_exit.awk

```
/^#/ { next }                    # skip comments
NR == 1 { next }                 # skip header
$1 == "STOP" { exit }           # stop reading
{ count++ }
END { print count, "data lines" }
```

- next stops processing the current line and jumps to the next one. Use it to skip comments …
- exit stops reading input completely, but the END block still runs. It is useful when you …

5.03

Multi-Line Programs

terminal

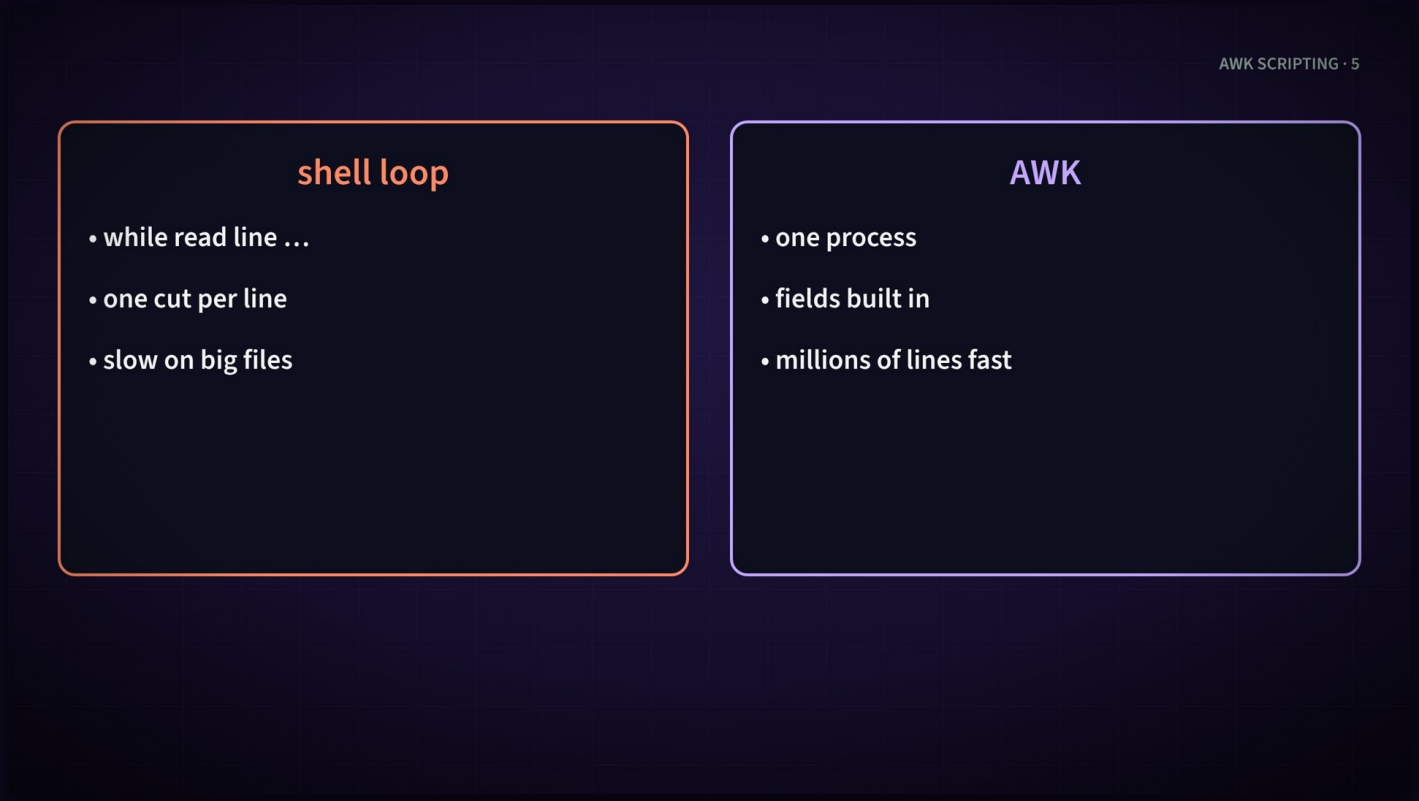
```
$ cat report.awk
# report.awk — totals per line
NR == 1 { next }
{
    total += $2
}
END { printf "Total: %d\n", total }

$ awk -f report.awk scores.txt
```

- One-liners grow. When they do, put the program in a file and run it with awk dash f. …
- Indent like any other language. A readable AWK file is easy to keep and share with your team.

5.04

AWK Logic vs Shell Logic



- A shell while-read loop runs a separate program for every line and is slow. One AWK process …
- So when your shell script loops over lines and cuts columns, it is usually time to switch …

5.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 6

printf and Reports

Formatted output, alignment, decimals and redirecting output

printf Basics

printf.awk

```
{ printf "%s scored %d\n", $1, $2 }

# Mariam scored 95
# Ahmad scored 88
```

- print is quick, but printf gives you control. It takes a format string with placeholders, …
- Percent s is a string, percent d an integer, and percent f a decimal number. Unlike print, …

6.01

Width and Alignment

table.awk

```
BEGIN { printf "%-10s %6s %8s\n", "Name", "Score", "Bonus" }
      { printf "%-10s %6d %8.2f\n", $1, $2, $2 * 0.1 }

# Name        Score    Bonus
# Mariam         95     9.50
# Ahmad          88     8.80
```

- A number after the percent sign sets the column width. Percent minus ten s means …
- Percent eight point two f means eight characters wide with two decimals, perfect for money …

6.02

Format Cheat Sheet

Format	Example	Output
%s · %-10s	"Kabul"	Kabul · Kabul
%d · %05d	42	42 · 00042
%.2f	3.14159	3.14
%x	255	ff
%5.1f%%	87.25	87.3%

- Here are the formats you will use most. Percent x shows hexadecimal, percent e scientific …
- sprintf works like printf but returns the string instead of printing it, so you can store a …

6.03

Writing to Files

split_by_city.awk

```
{ print > ($3 ".txt") }           # Kabul.txt, Herat.txt ...

$2 < 60 { print $1 >> "retake.txt" }

END { close("retake.txt") }
```

- print and printf can send output to a file with the greater-than sign, or append with two …
- The file name can be a variable, so one AWK run can split a big file into one file per …

6.04

A Complete Report

report.awk

```
BEGIN { printf "%-10s %6s\n", "Student", "Score"; print "-----------------" }
{ printf "%-10s %6d\n", $1, $2; n++; sum += $2 }
END {
    print "-----------------"
    printf "%-10s %6d\n%-10s %6.1f\n", "Total", sum, "Average", sum / n
}
```

- Here is everything together: a header in BEGIN, aligned rows in the main rule, and a …
- This is the shape of most AWK reports you will write in real work.

6.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 7

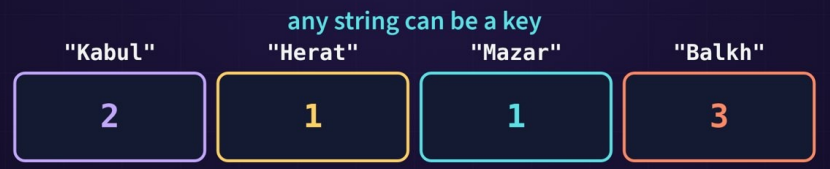
Associative Arrays

Counting, grouping, for-in loops, delete and split

Arrays with Any Key

report.awk

```
count[$3]++
```



- AWK arrays are associative: the index can be any string, not just a number. count of Kabul, ...
- Like variables, arrays need no declaration and grow automatically. Think of them as a ...

7.01

Counting and Summing by Group

by_city.awk

```
{ count[$3]++; sum[$3] += $2 }
END {
    for (city in count)
        printf "%-8s %3d students  avg %.1f\n", city, count[city], sum[city] / count[city]
}
```

- The classic AWK idiom is count of dollar three plus plus. It counts how many lines belong …
- The same idea sums by group: sum of dollar three plus equals dollar two gives the total …

7.02

Looping with for-in

terminal

```
$ awk '{ c[$3]++ } END { for (k in c) print c[k], k }' scores.txt | sort -rn
3 Balkh
2 Kabul
1 Herat

$ awk '!seen[$0]++' list.txt        # remove duplicate lines
```

- for key in array visits every key. The order is not guaranteed, so pipe the output into …
- Use in as a test, too: if key in array checks whether a key exists without creating it.

7.03

delete and split

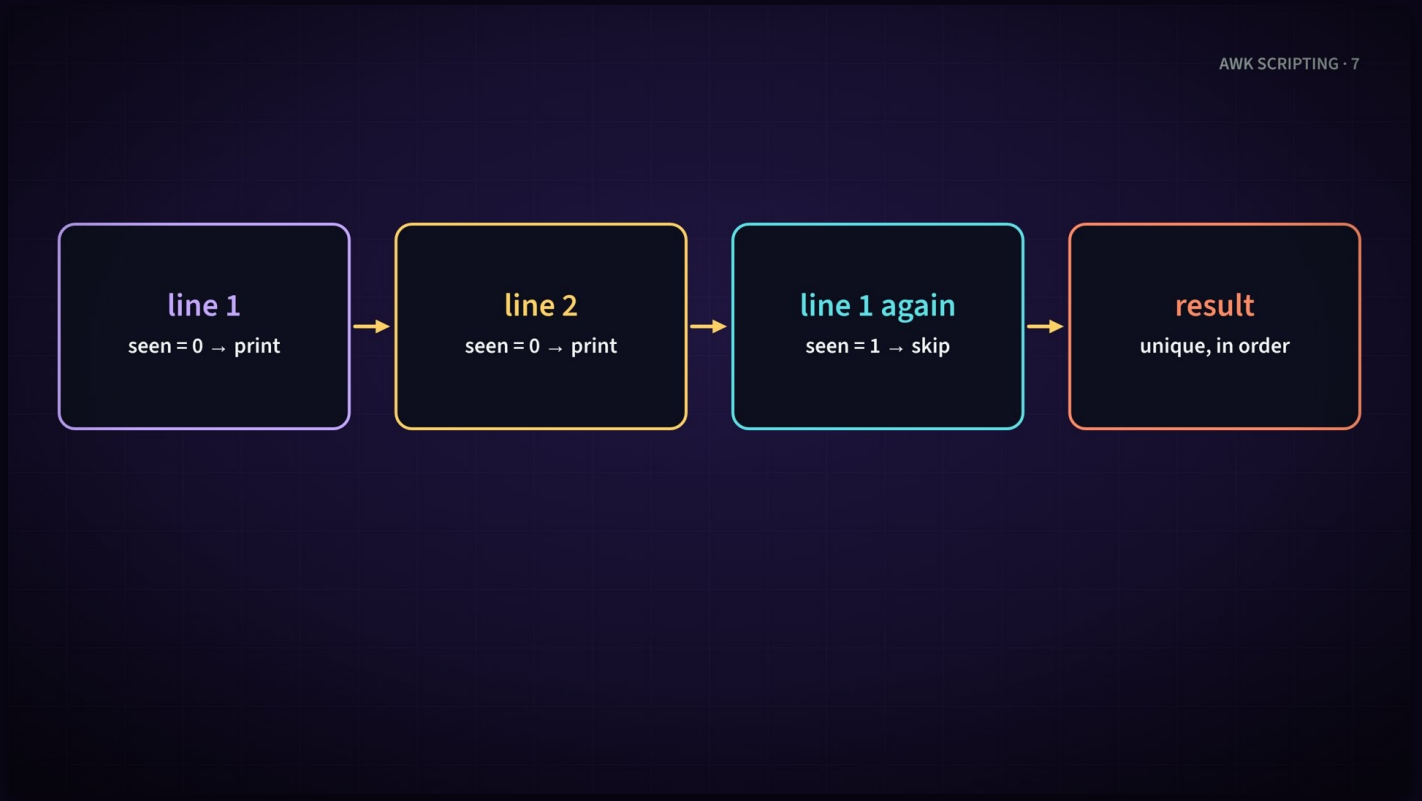
split.awk

```
{
    n = split($4, d, "-")          # 2026-10-24
    print "year", d[1], "month", d[2], "day", d[3]
    delete d                         # clear the array
}
```

- delete array of key removes one element, and delete array removes them all. split breaks a …
- split of a date with a dash as separator gives you year, month and day in elements one, two …

7.04

The Famous Dedup One-Liner



- line 1 — seen = 0 → print
- line 2 — seen = 0 → print
- line 1 again — seen = 1 → skip

7.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 8

String Functions

length, substr, index, sub, gsub, match, toupper and tolower

length, substr and index

Call	Result	Note
`length("Kabul")`	5	characters
`substr("2026-10-24", 1, 4)`	2026	start at 1
`substr("Kabul", 3)`	bul	to the end
`index("Herat", "rat")`	3	position
`index("Herat", "x")`	0	not found

- length returns the number of characters. substr of a string, start, length cuts out a …
- index returns where one string first appears inside another, or zero if it is not there.

8.01

sub and gsub

clean.awk

```
{
    gsub(/,/, "", $2)              # 1,250 -> 1250
    sub(/^ +/, "")                 # trim leading spaces
    gsub(/[ \t]+$/, "")            # trim trailing spaces
    n = gsub(/error/, "ERROR")      # n = number replaced
    print
}
```

- sub replaces the first match of a regular expression, and gsub replaces every match. Both …
- gsub of slash comma slash with nothing removes all commas from dollar two, which is how you …

8.02

toupper, tolower and match

match.awk

```
tolower($1) == "mariam" { print "found" }

match($0, /[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/) {
    ip = substr($0, RSTART, RLENGTH)
    print ip
}
```

- toupper and tolower change case, which is useful before comparing names typed in different …
- match finds a regular expression and sets RSTART and RLENGTH, so you can cut out exactly …

8.03

Building Strings with sprintf

ids.awk

```
{
    id  = sprintf("STU-%04d", NR)        # STU-0001
    avg = sprintf("%.1f", $2 / 3)
    key = sprintf("%s|%s", $3, $1)        # Kabul|Mariam
    print id, key, avg
}
```

- sprintf formats a string just like printf, but stores the result instead of printing it.
- Use it to build keys, IDs with leading zeros, or rounded numbers that you keep for later.

8.04

Text Cleaning Recipes

Code	Does
`gsub(/^[ \t]+|[ \t]+$/, "")`	trim both ends
`gsub(/ +/, " ")`	squeeze spaces
`gsub(/"/, "")`	remove quotes
`$1 = toupper(substr($1,1,1)) substr($1,2);`	capitalize

- Here are recipes you will use again and again: trim spaces, normalize case, remove quotes, …
- Save them in a small file of AWK functions, which we will learn to write in the next lesson.

8.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 9

Functions and Script Files

User-defined functions, -f, shebang scripts and -v options

Writing Functions

names.awk

```
function trim(s) { gsub(/^[ \t]+|[ \t]+$/, "", s); return s }
function capitalize(s) { return toupper(substr(s, 1, 1)) tolower(substr(s, 2)) }

{ print capitalize(trim($1)) }

#  mARIAM  ->  Mariam
```

- Define a function with the word function, a name, and parameters in parentheses. Call it …
- Functions keep repeated logic in one place. Here trim and capitalize clean every name in …

9.01

Local Variables: the Extra-Parameter Trick

sum.awk

```
function sum_fields(from, to,    i, total) {   # i, total are local
    for (i = from; i <= to; i++) total += $i
    return total
}
{ print $1, sum_fields(2, NF) }
```

- In AWK every variable is global, except function parameters. So the convention is to add …
- Callers never pass those extra parameters. Arrays are passed by reference, so a function …

9.02

Executable AWK Scripts

terminal

```
$ cat report.awk
#!/usr/bin/awk -f
BEGIN { FS = "," }
{ total += $2 }
END { print "Total:", total }

$ chmod +x report.awk
$ ./report.awk sales.csv
Total: 48250
```

- Put a shebang line at the top, hash exclamation slash usr slash bin slash awk dash f, then …
- Now your AWK program runs like any command: dot slash report dot awk, then the data file.

9.03

Options and Libraries

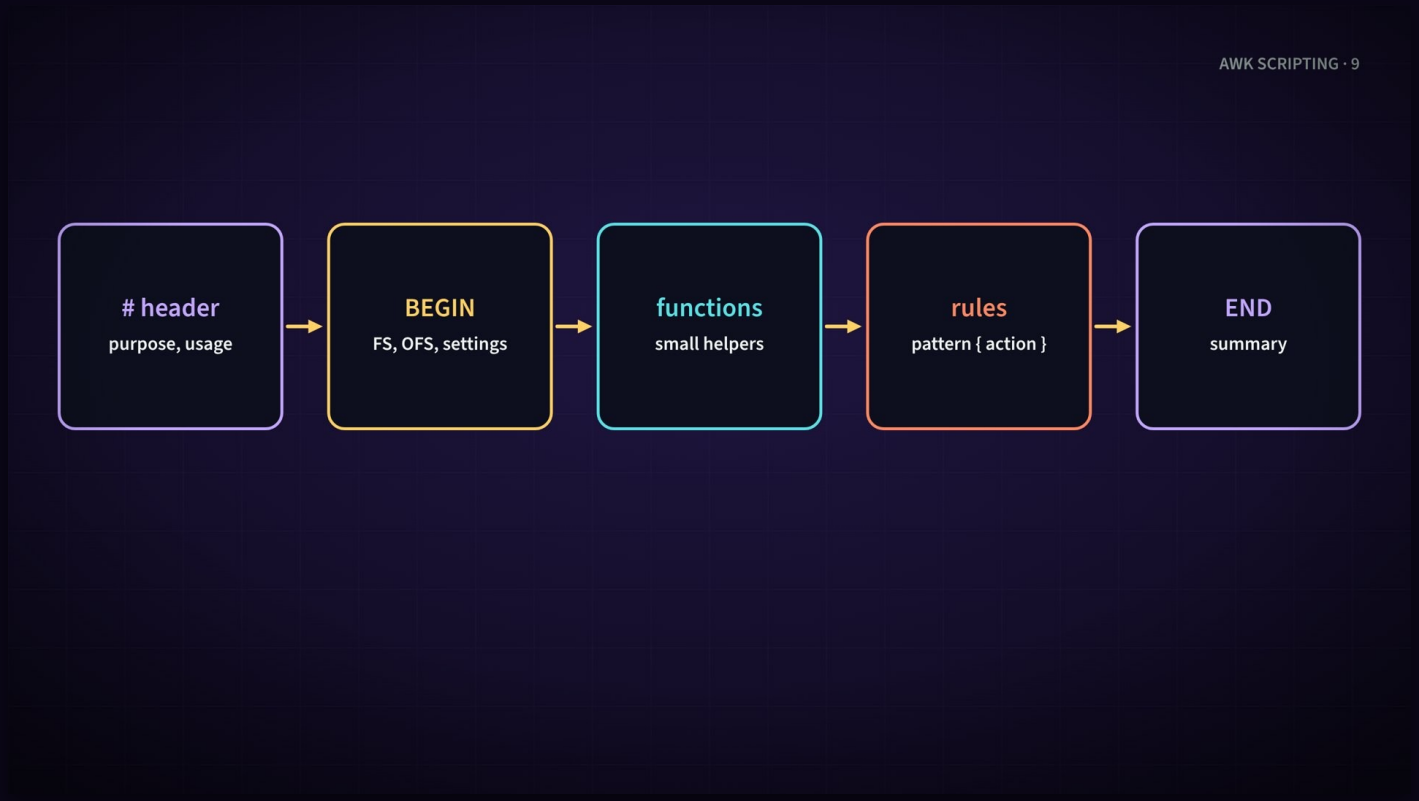
terminal

```
$ awk -f lib.awk -f report.awk data.txt
$ gawk -i lib.awk -f report.awk data.txt
$ gawk --lint -f report.awk data.txt
gawk: warning: reference to uninitialized variable `totl'
```

- Pass settings with dash v, and load several files with more than one dash f, so a file of …
- gawk also has dash i to include a library and dash dash lint to warn about mistakes. Always …

9.04

A Well-Organized Script



- # header — purpose, usage
- BEGIN — FS, OFS, settings
- functions — small helpers

9.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 10

Multiple Files and I/O

FNR, joining two files, getline, pipes and system()

FNR vs NR

terminal

```
$ awk 'FNR == 1 { print "==>", FILENAME } { print FNR, NR, $0 }' a.txt b.txt
==> a.txt
1 1 alpha
2 2 beta
==> b.txt
1 3 gamma
```

- When you give AWK several files, NR keeps counting across all of them, but FNR restarts at …
- So FNR equals one marks the start of every file, and FILENAME tells you which file you are in.

10.01

Joining Two Files

join.awk

```
# usage: awk -f join.awk names.txt scores.txt
NR == FNR { name[$1] = $2; next }      # first file: id -> name
{ print name[$1], $2 }                  # second file: id score

# names.txt : 101 Mariam      scores.txt: 101 95
# output    : Mariam 95
```

- The NR equals FNR trick is true only while reading the first file. Use it to load a lookup …
- Here the first file maps student IDs to names, and the second file has scores by ID. AWK …

10.02

getline and Pipes

stamp.awk

```
BEGIN {
    "date +%F" | getline today
    close("date +%F")
    "hostname" | getline host
}
{ print today, host, $0 }
```

- getline reads the next line yourself. The most useful form runs a command and reads its …
- This lets AWK ask the system for the date, the hostname, or any other value while it works.

10.03

Sending Output to Commands

pipes.awk

```
{ total[$3] += $2 }
END {
    for (c in total) print total[c], c | "sort -rn"
    close("sort -rn")
    system("echo report done")
}
```

- print can also pipe into a command: print, then a bar, then sort in quotes. AWK keeps that …
- system runs a shell command and returns its exit status. Use it with care, and never with …

10.04

AWK in Shell Pipelines

terminal

```
$ df -h | awk 'NR > 1 && $5+0 > 80 { print $6, $5 }'      # disks over 80%
$ ps aux | awk '{ mem[$1] += $6 } END { for (u in mem) print u, mem[u] }'
$ ls -l | awk '{ s += $5 } END { print s / 1024 / 1024, "MB" }'
```

- AWK is happiest in the middle of a pipeline: commands produce text, AWK shapes it, and …
- Here df, ps and ls are turned into clean answers with one short AWK step each.

10.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 11

Real-World AWK

Web server logs, CSV reports, system monitoring and data cleaning

Analyzing a Web Server Log

terminal

```
$ awk '{ ip[$1]++ } END { for (i in ip) print ip[i], i }' access.log | sort -rn | head -5
1520 203.0.113.7
 988 198.51.100.23

$ awk '{ code[$9]++ } END { for (c in code) print c, code[c] }' access.log
200 18233
404 412
500 9
```

- An access log has one request per line: IP address, date, request, status code and size. …
- In one line we find the top visitors, and in another we count each status code, so errors …

11.01

CSV Sales Report

sales.awk

```
BEGIN { FS = "," }
NR == 1 { next }
{ sales[$2] += $4; total += $4 }
END {
    for (r in sales) printf "%-10s %10.2f %5.1f%%\n", r, sales[r], 100 * sales[r] / total
    printf "%-10s %10.2f\n", "TOTAL", total
}
```

- A sales CSV has a header, then date, region, product and amount. We skip the header, sum by …
- Real CSV can contain quoted commas. gawk has FPAT for that, which describes what a field …

11.02

System Monitoring

terminal

```
$ df -P | awk 'NR > 1 { u = $5 + 0; if (u > 90) print "WARNING:", $6, u "%" }'
WARNING: /var 93%

$ free | awk '/Mem:/ { printf "memory used: %.1f%%\n", $3 / $2 * 100 }'
memory used: 61.4%
```

- AWK reads the files in slash proc and the output of commands like df and free, so it is …
- Here we warn when a disk is over 90 percent full and print memory use as a percentage.

11.03

Cleaning Data

clean.awk

```
BEGIN { FS = OFS = "," }
NF == 0 { next }                          # empty line
NF != 4 { print > "rejected.csv"; next }      # wrong shape
{
    for (i = 1; i <= NF; i++) gsub(/^ +| +$/, "", $i)
    $2 = toupper($2); print
}
```

- Before analysis, data must be clean. AWK can drop empty lines, fix spaces, standardize …
- Send bad rows to a separate file so nothing is lost and you can check them later.

11.04

Speed and Safety Tips



- LC_ALL=C — faster on big files
- filter early — pattern before action
- 'single quotes' — shell-safe program

11.05



A FREE MYWEBUNIVERSITY COURSE

AWK SCRIPTING

LESSON 12

gawk Power Features and Next Steps

Sorting arrays, time functions, AWK vs sed vs grep vs Python

Sorted Loops in gawk

top.awk

```
{ count[$1]++ }
END {
    PROCINFO["sorted_in"] = "@val_num_desc"     # gawk only
    for (ip in count) {
        print count[ip], ip
        if (++n == 5) break
    }
}
```

- Plain AWK loops over arrays in random order. gawk lets you choose the order with PROCINFO …
- gawk also has asort and asorti to sort an array's values or keys into a new array.

12.01

Time Functions

time.awk

```
BEGIN {
    now = systime()
    print strftime("%Y-%m-%d %H:%M", now)
    t = mktime("2026 01 01 00 00 00")
    printf "%d days since New Year\n", (now - t) / 86400
}
```

- gawk has systime for the current time in seconds, strftime to format dates, and mktime to …
- With these you can stamp reports, compute how many days passed between two dates, or group …

12.02

AWK, sed, grep or Python?

Tool	Best for
`grep`	finding lines that match
`sed`	search and replace in a stream
`awk`	columns, counting, reports
`python`	big programs, libraries, APIs

- Use grep to find lines, sed to edit text in a stream, and AWK when you need columns, …
- When a script grows past a page, needs libraries, or talks to databases and web APIs, move …

12.03

AWK Cheat Sheet

cheatsheet.awk

```
awk -F, -v min=80 '                  # separator, variable
BEGIN { OFS = "\t" }                  # before input
NR == 1 { next }                      # skip header
$2 >= min && $3 ~ /^K/ {              # pattern
    count[$3]++; sum[$3] += $2        # arrays
}
END { for (c in count) printf "%s\t%.1f\n", c, sum[c]/count[c] }
' data.csv
```

- Here is the whole course on one screen: fields, patterns, BEGIN and END, arrays, functions …
- Keep it next to your terminal while you practice, and soon you will write these from memory.

12.04

Projects and Next Steps



- 1 — Log analyzer
- 2 — Expense report
- 3 — Disk alert

12.05

Thank you for learning with us

MyWebUniversity.com

Learn it all @MyWebUniversity

Free technical education since 2000 — English and Dari