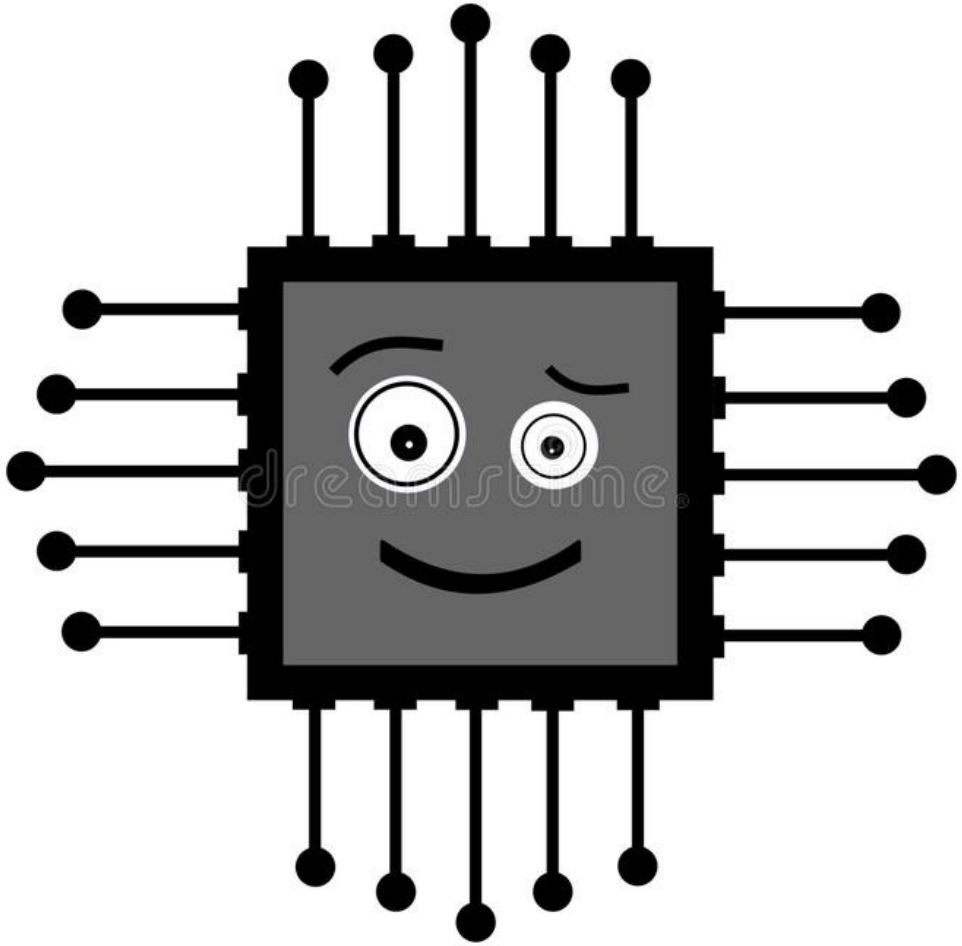
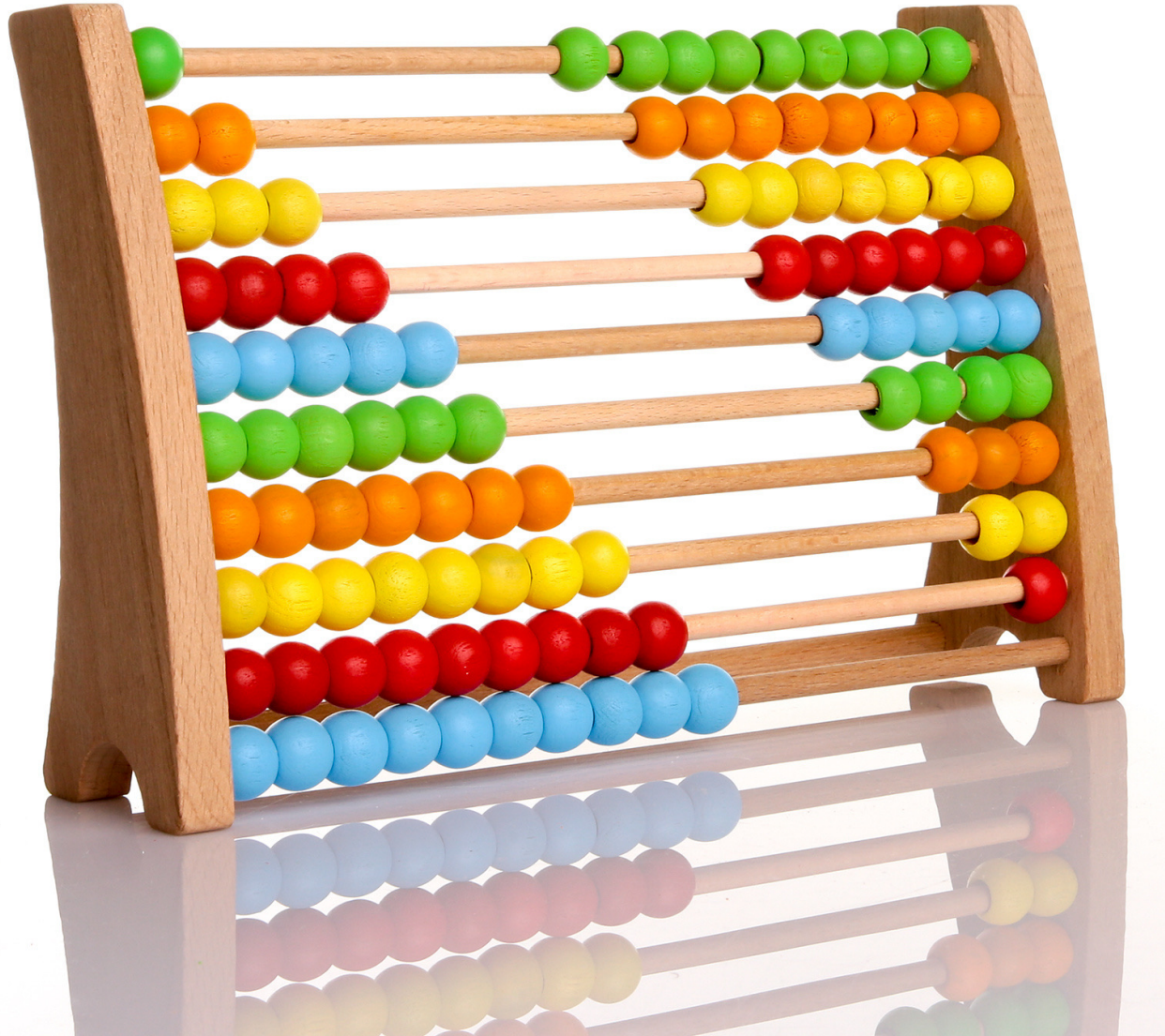
Introduction to Programming and R

Konuralp İlim

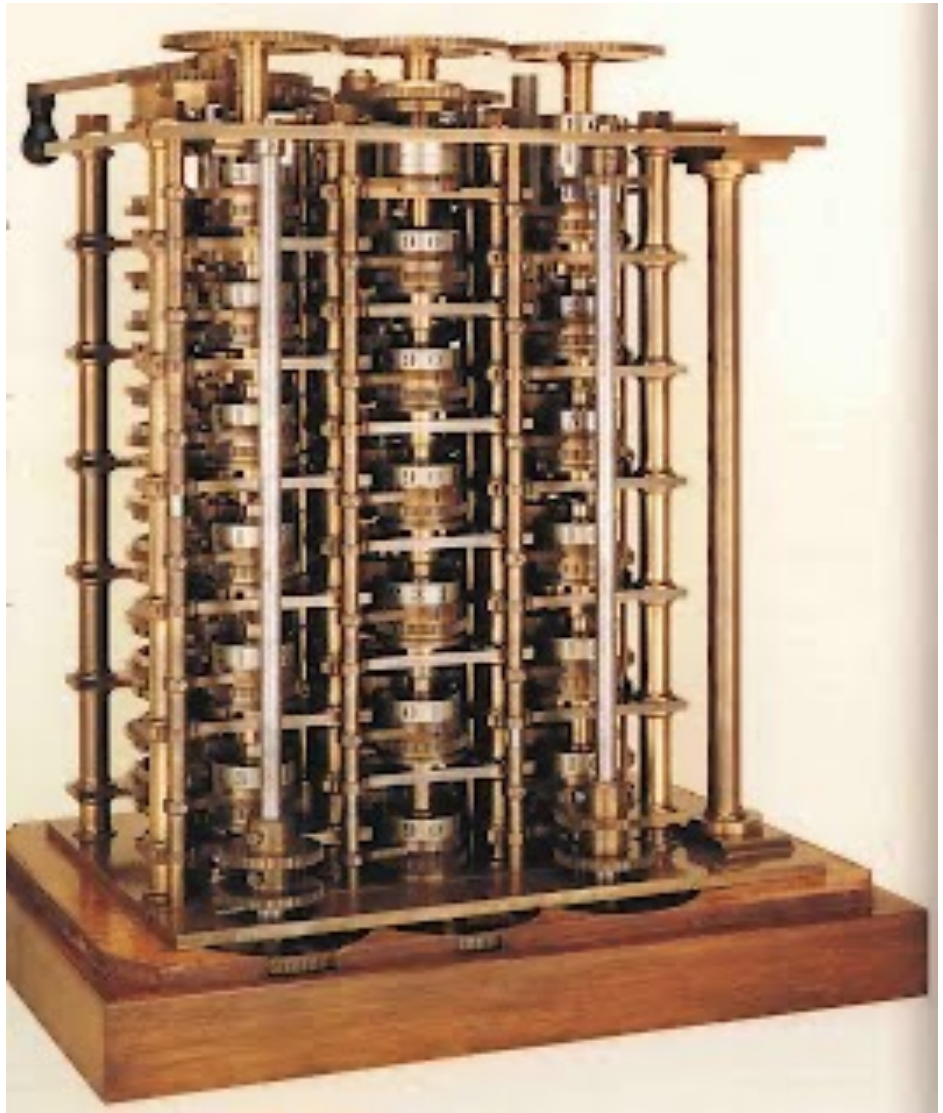
CPU



Electrical Fully Automatic Abacus



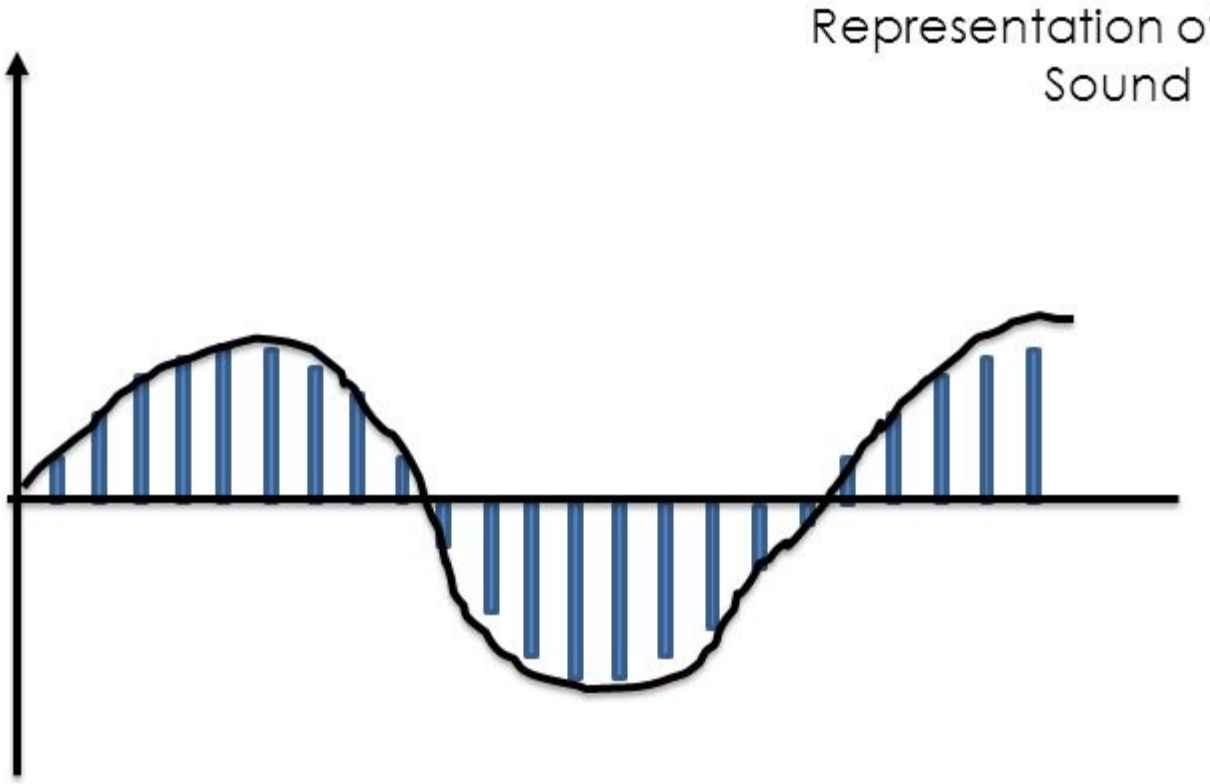
Early Mechanical Computers



- For the interested: https://www.youtube.com/watch?v=O5nskjZ_GoI

Sampling

Numeric Sound Representation



011010 010010 111010 111011 001011 111001
011110 001010 111010 110001 011011 011011
001010 110010 111010 101011 111011 011101
010010 110110 111011 110010 001111

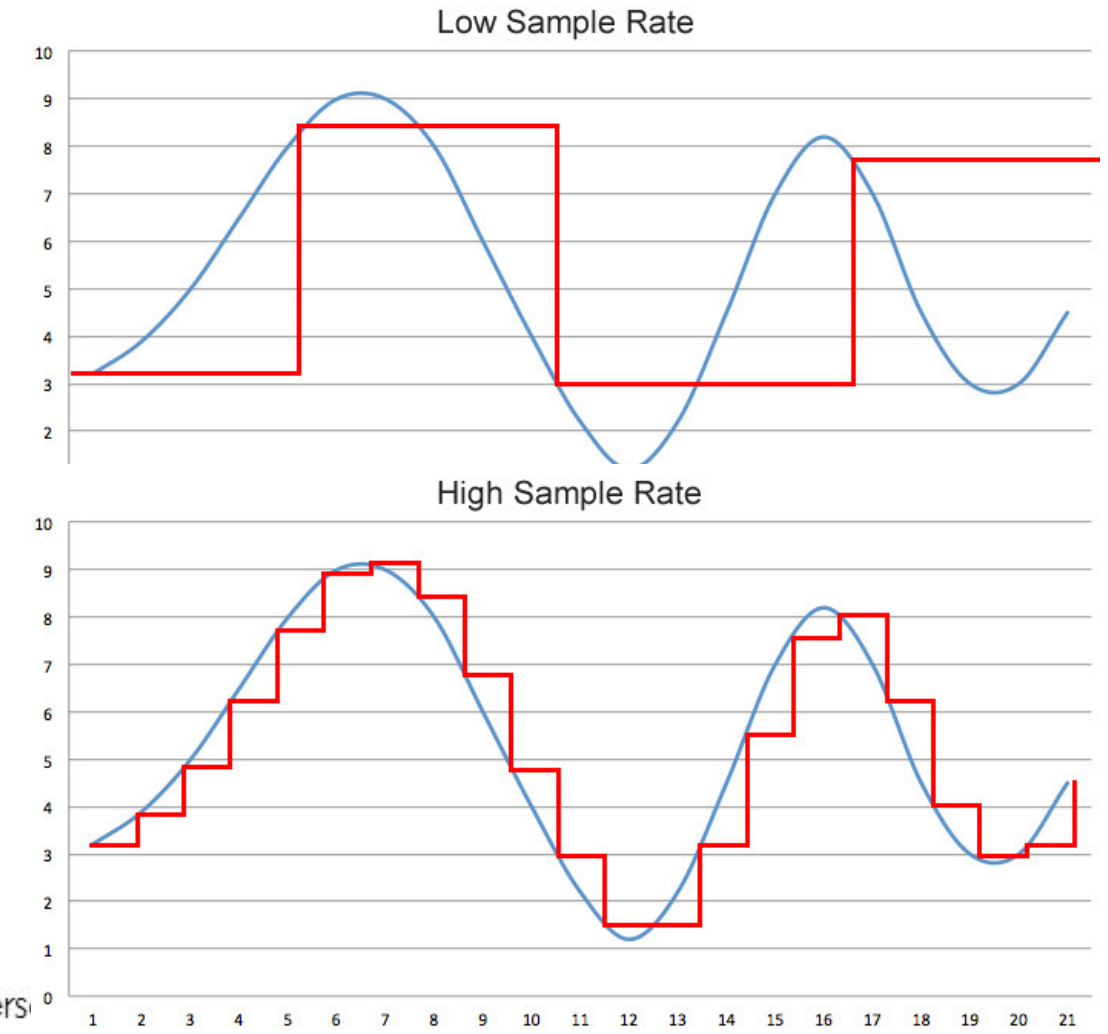
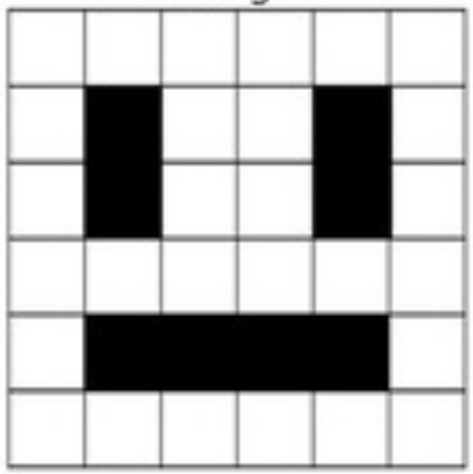


Image Representation

0	0	0	0	0	0
0	1	0	0	1	0
0	1	0	0	1	0
0	0	0	0	0	0
0	1	1	1	1	0
0	0	0	0	0	0

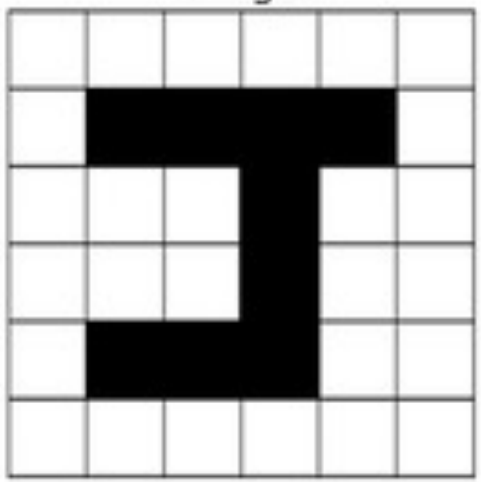
Image



Binary

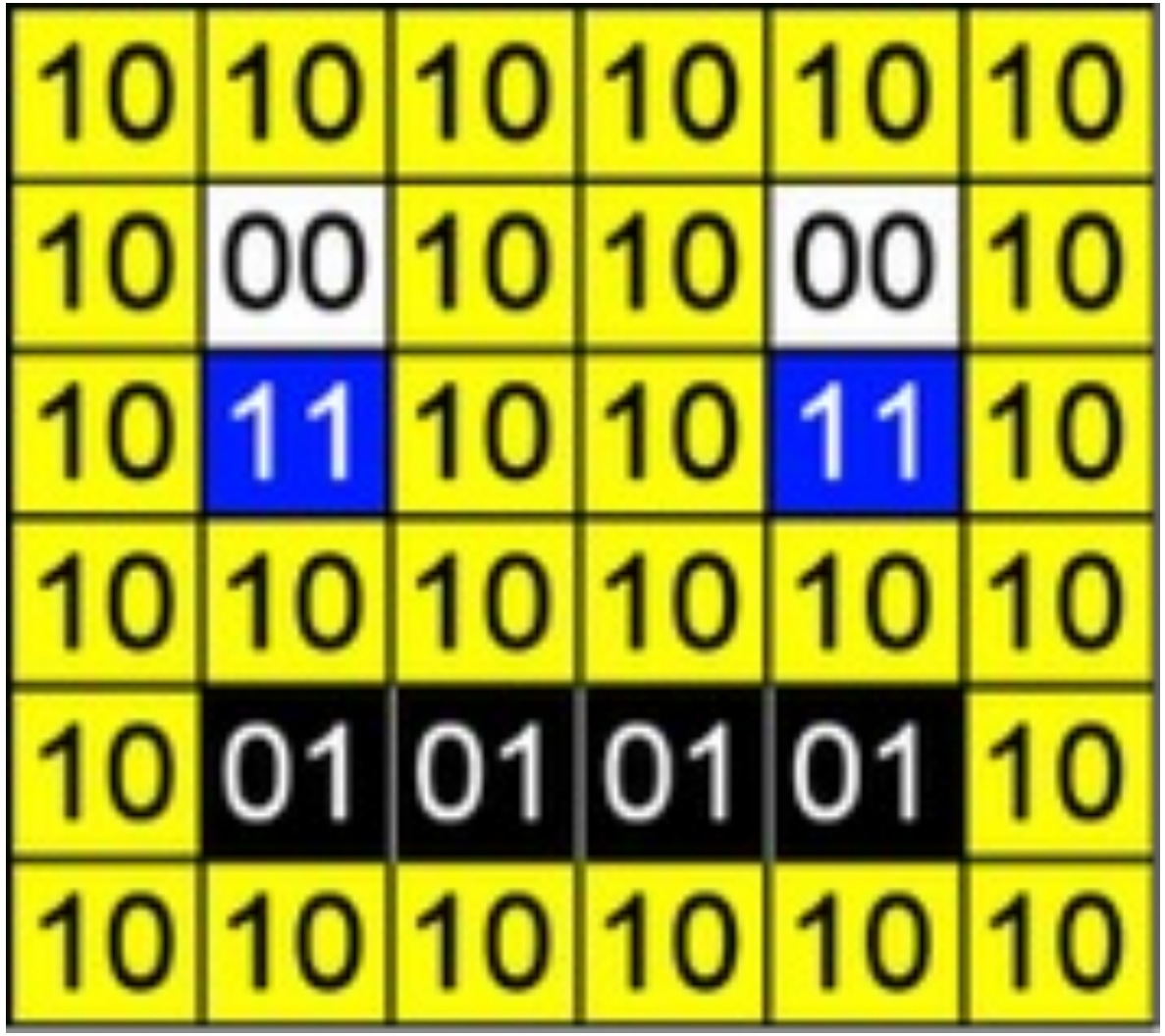
0 0 0 0 0 0
0 1 0 0 1 0
0 1 0 0 1 0
0 0 0 0 0 0
0 1 1 1 1 0
0 0 0 0 0 0

Image



Binary

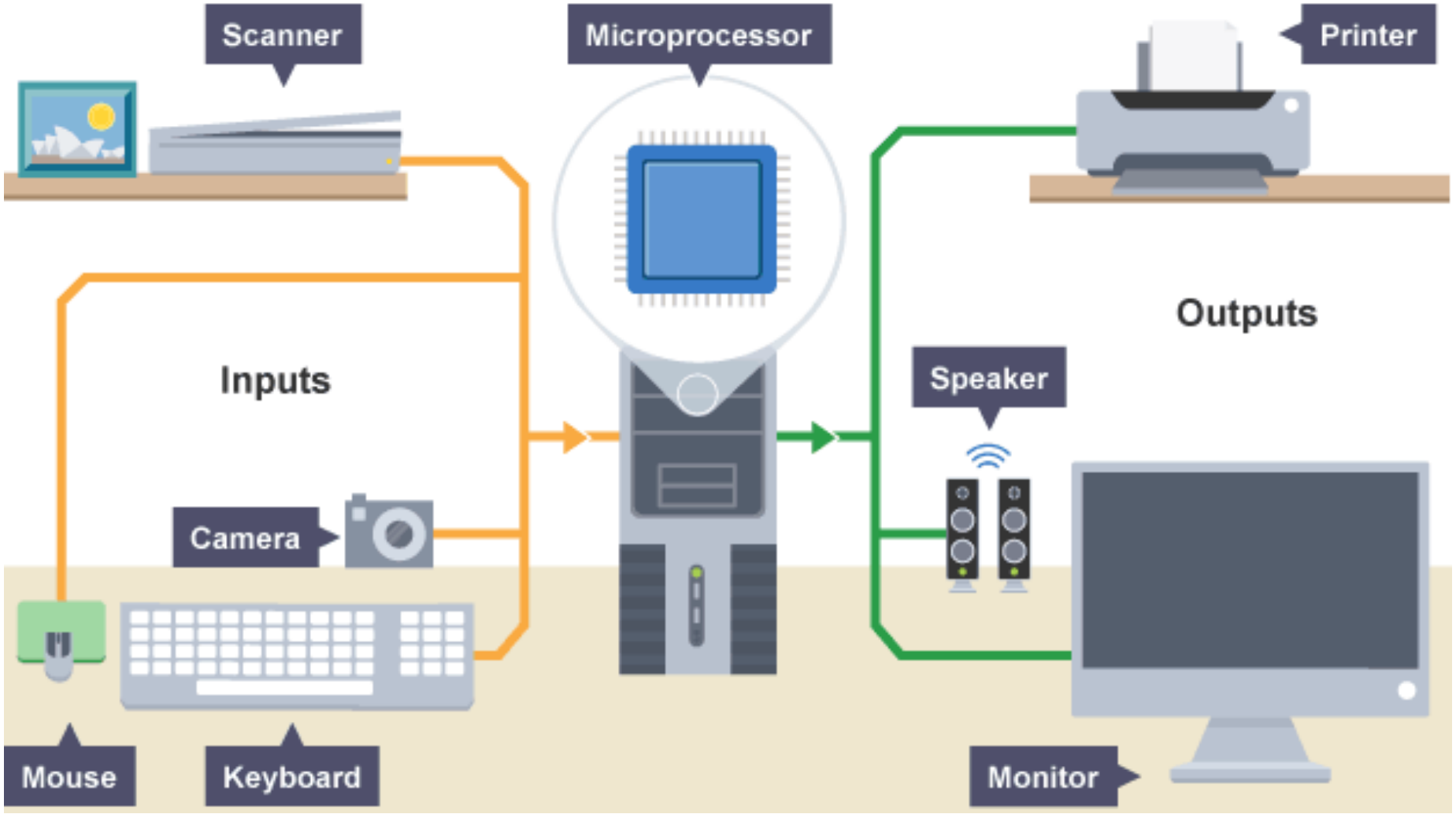
0 0 0 0 0 0
0 1 1 1 1 0
0 0 0 1 0 0
0 0 0 1 0 0
0 1 1 1 0 0
0 0 0 0 0 0


10 10 10 10 10 10
10 00 10 10 00 10
10 11 10 10 11 10
10 10 10 10 10 10
10 01 01 01 01 10
10 10 10 10 10 10

Computer Memory

What is RAM?

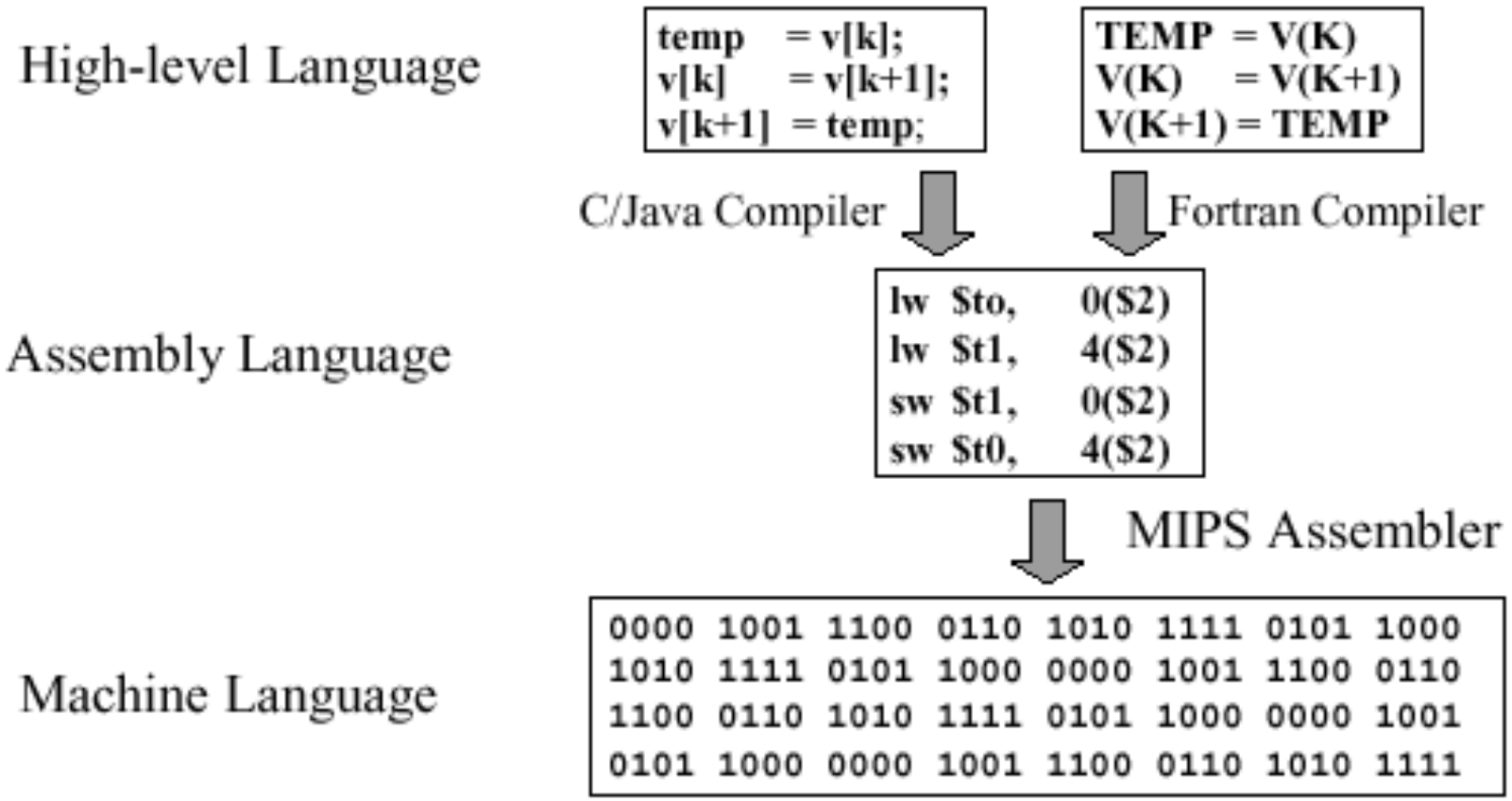
Input Output



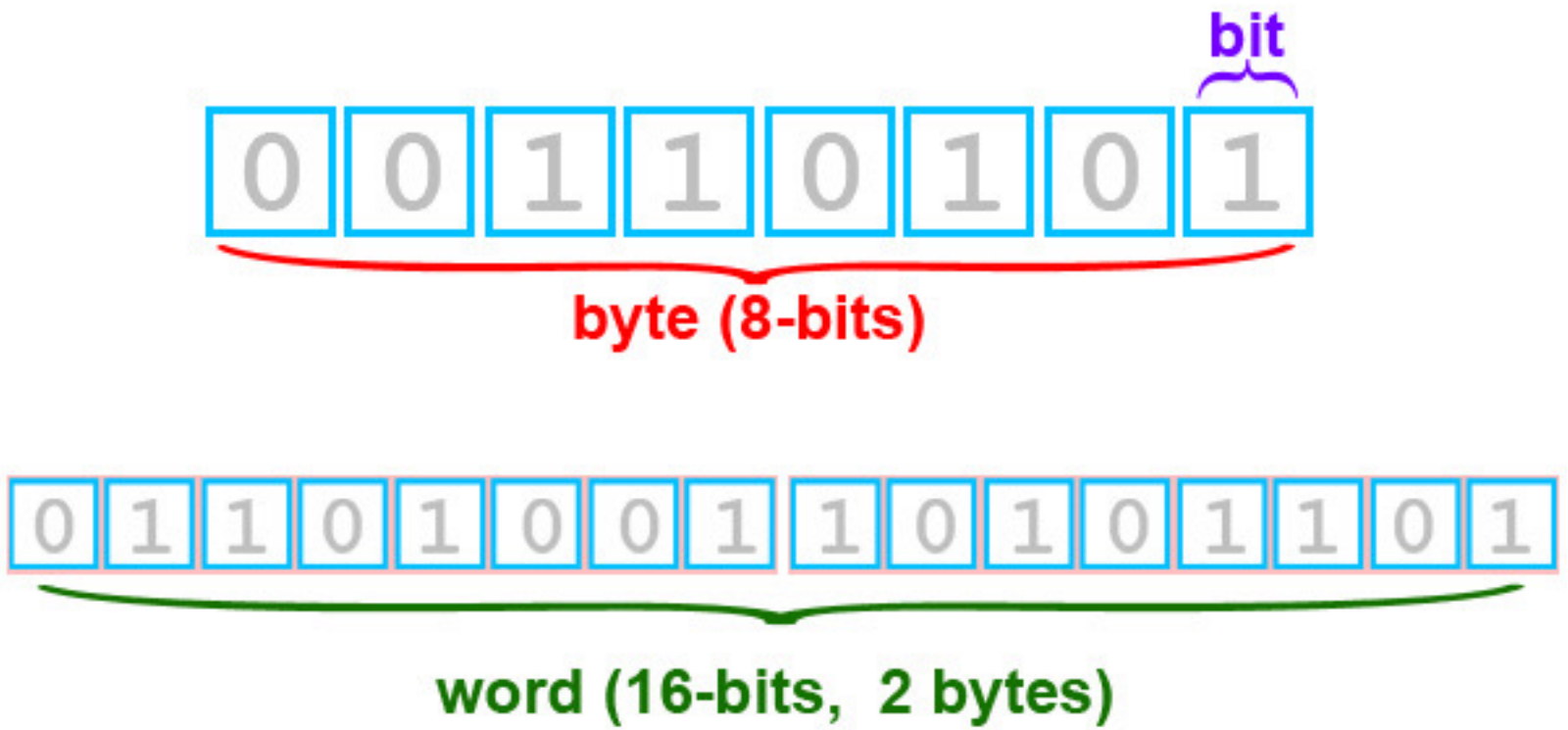
Instruction Set

- Input/Output (e.g. input from keyboard, output to monitor)
- Math Operations
  (eg. sum +,subtraction -, multiplication x, division /, modulus %)
- Logical Operations (e.g. and, or, xor)
- Control Operation (if, else, jump)

CPU can only run numbers: machine code



Smallest Units Of Information On Computer



- Compiled
  - C
  - C++
  - Haskell
  - Java
- Interpreted
  - Python
  - R
  - Javascript

R Programming Language

- “The R programming language is an open source scripting language for predictive analytics and data visualization.

- The initial version of R was released in 1995 to allow academic statisticians and others with sophisticated programming skills to perform complex data statistical analysis and display the results in any of a multitude of visual graphics. The "R" name is derived from the first letter of the names of its two developers, Ross Ihaka and Robert Gentleman, who were associated with the University of Auckland at the time.”

https://www.techtarget.com/searchbusinessanalytics/definition/R-programming-language

Syntax, semantics, pragmatics

- Syntax
  - How it looks, i.e. how we have to program to satisfy the compiler.
- Semantics
  - What it means / how it works
- Pragmatics
  - How to use it in the proper way.

https://slideplayer.com/amp/4172187/

- Reserved Words in R

if	else	repeat	While	Function	for	in
next	break	TRUE	FALSE	NULL	Inf	Nan
NA	NA_integer	NA_real	NA_complex	NA_character	.....	.....

https://learnetutorials.com/r-programming/identifiers-constants-reserved-words

What are identifiers in R?

In R programming language an identifier is a composition of alphabets (**a-z, A-Z**), digits (**0,1,…**),dot(.) or underscore(_).

Examples of acceptable identifiers in R are

```
site_123, Site, site, SiteName, Sitename, .site_123, Language_1
```

Examples of identifiers that are not acceptable are

```
_language, Language$1, 1language, .123_site
```

https://learnetutorials.com/r-programming/identifiers-constants-reserved-words

Primitives for R

- Integer: 11,10,1024,999999,01
- Float (decimal numbers): 1.234, 14.24123, 99.123, 9999.0
- Boolean: TRUE, FALSE
- Char and String: ‘H’, ‘This is Text.’, “123”, “999999”, “9999.0”
- Missing Values: NA
- Factors (don’t worry now)

Variables

- Following identifier naming rule like not containing weird characters in weird places, you can name a variable anything you want, whatever makes you work better, whatever helps you remember the structure and solve the problem better, clearly and faster
- But there are naming conventions that suggest but does not reinforce naming rules for different structures e.g. camelCase, underscore_naming_convention, snake_case, CapitalCase
- Variables are identifiers that are assigned data, and now holds data to be used on demand

myVariable

sarıÇizmeliMehmetAğa

asd

mehmetAğa2

sonucumunDoğruluğu

Assignment Operator

myVariable <- “text”

sarıÇizmeliMehmetAğa <- 123

asd = 123

mehmetAğa2 <- 321

sonucumunDoğruluğu = TRUE

sonucumunDoğruluğu <- TRUE

Operations

- Math
  - Sum: 3+5
  - Subtract: 5-3
  - Multiply: 3*5
  - Divide: 5/3
  - Modulus: 5%3

What is a Literal

myVariable <- 5

secondOne <- myVariable*myVariable

5 is a integer literal

myVariable and secondOne are integer variables

Operations

- Logic
  - AND: TRUE&&FALSE
  - OR: TRUE||FALSE
  - NOT: !TRUE
  - Greater than: 5>3
  - Less than: 5<3
  - Greater than or equal to: 5>=5
  - Less than or equal to: 5<=4
  - Equal to: 5==5
  - Not Equal to: 3!=3

Operator Precedence

Don’t need to memorize precedence

Use parentheses, because others don’t remember.

You can check the precedence rules here:

https://stat.ethz.ch/R-manual/R-devel/library/base/html/Syntax.html

```
a <- 5
b <- 3



c <- a*b

c?
```

(5>3) || (3!=3)

(5<3) && (3==3)

??

How To Make Comment and Leave Notes

While writing code in most languages, there are ways to leave text that won’t be tried to be turned into machine code and affect what program does. In R, we do this with “#” character.

```
myVariable <- 5
#Now I’m gonna take square of this variable
myVariable <- myVariable*myVariable
#MyVariable is square of what it was before, in this case: 25
print(myVariable)
#Now it prints 25
```

Block

```
3+5
10/2
2*3
```

3 expressions

```
{
3+5
10/2
2*3
}
```

1 expression

Function,Subroutine,Method

functionThatINamed(inputThatIWantToGive)

function(input)

which Returns a result, or just returns an empty, w/o a result

Example:

```
myFunc <- function(x) {return(x*x)}
myVar <- myFunc(5)
print(myVar)
#it will print 25
```

Control

- If Else

If(BOOLEAN) –EXPRESSION-

If(TRUE) 3+5

If(TRUE) {
3+5
2*3
10/2
} else {
print(“I was wrong”)
}

#While(BOOLEAN) –EXPRESSION-

```
myNumber <- 5
isMyNumberBigEnough <- myNumber == 100
while(!isMyNumberBigEnough){
        myNumber <- myNumber+1
        isMyNumberBigEnough <- myNumber == 100
}
Print(“my number is now 100”)
```

Data Structures

Arrays, tuples, constant size group of same type of data.
In R, we see arrays as vectors

```
myVector <- c(2,5,10)

#test the following
print(myVector+2)
print(myVector*myVector)

#To access Nth element use square brackets
print(myVector[2])
```

Data Structures

There are many types of list, a simple linked list contains elements that contain links(references) to other elements, thus doesn’t have to be constant size, and can be freely expanded.

An arraylist contains references to each element on an array, can creates a new array when there is a need to expand.

In R, we can make a list in multiple ways.

```
myList <- list()
#Or
myList <- vector(mode = "list)
#Then add or access elements via index using double brackets
myList[[1]] <- 5
myList[[2]] <- 10
print(myList[[1]])
```

Data Structures

#To turn list into a vector, use built in unlist() method

```
myList <- list()
myList[[1]] <- 5
myList[[2]] <- 10
myVector <- unlist(myList)
print(myVector)
```

A hashmap or a dictionary match keys and values, allowing faster access when searching for key, usually keys are strings, but don’t have to be

In R, we use dictionary by calling list

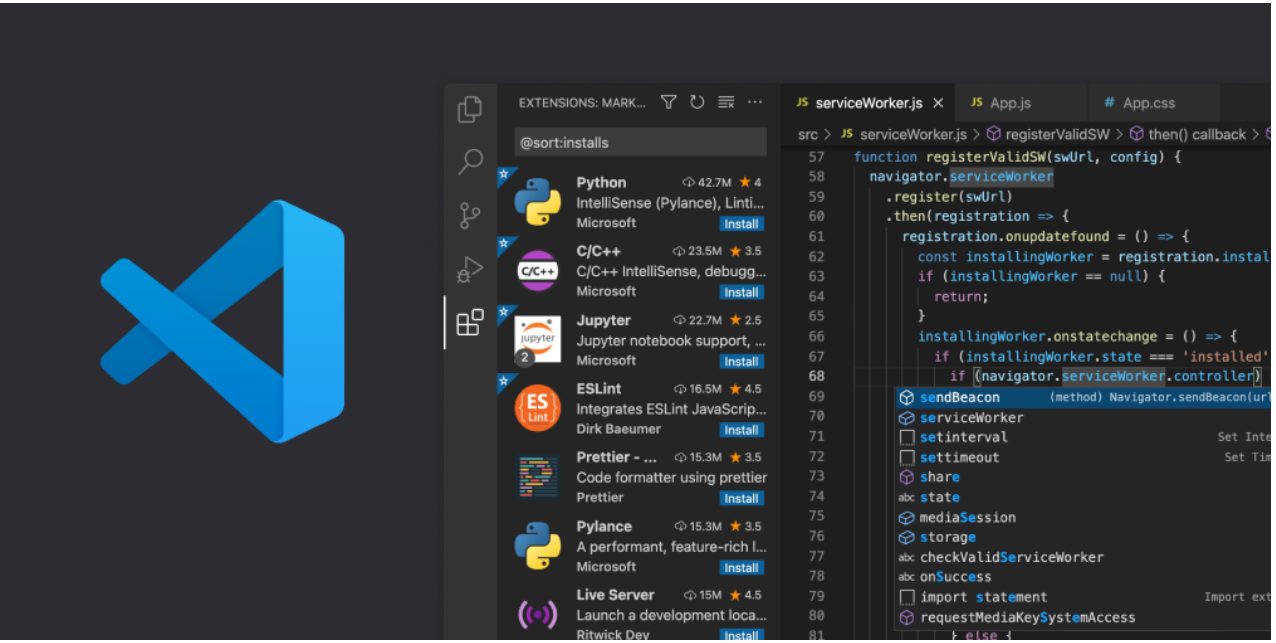
Example:

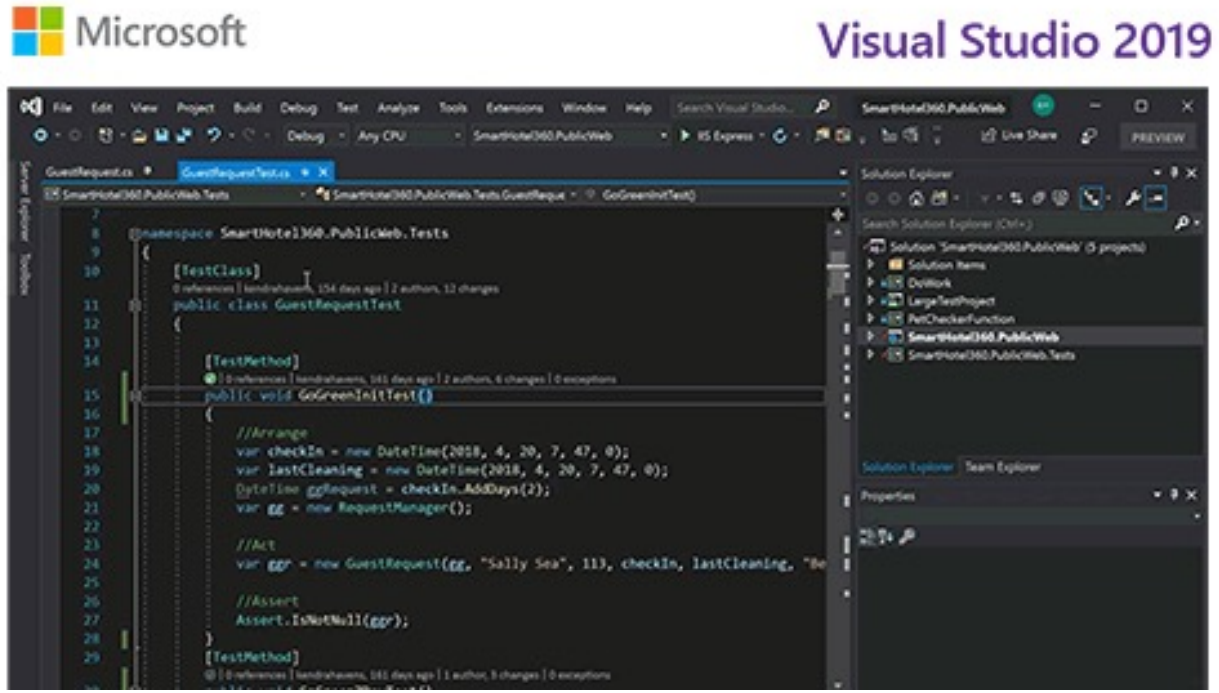
```
studentNumbers<- list()
studentNumbers[[“Mehmet Yılmaz”]] <- 2200039
studentNumbers[[“Ayşe Kaya”]] <- 2133385
#Now let’s get Mehmet’s number and add 2 to that, then print it
print(studentNumbers[[“Mehmet Yılmaz”]]+2)
```

Classes and Objects

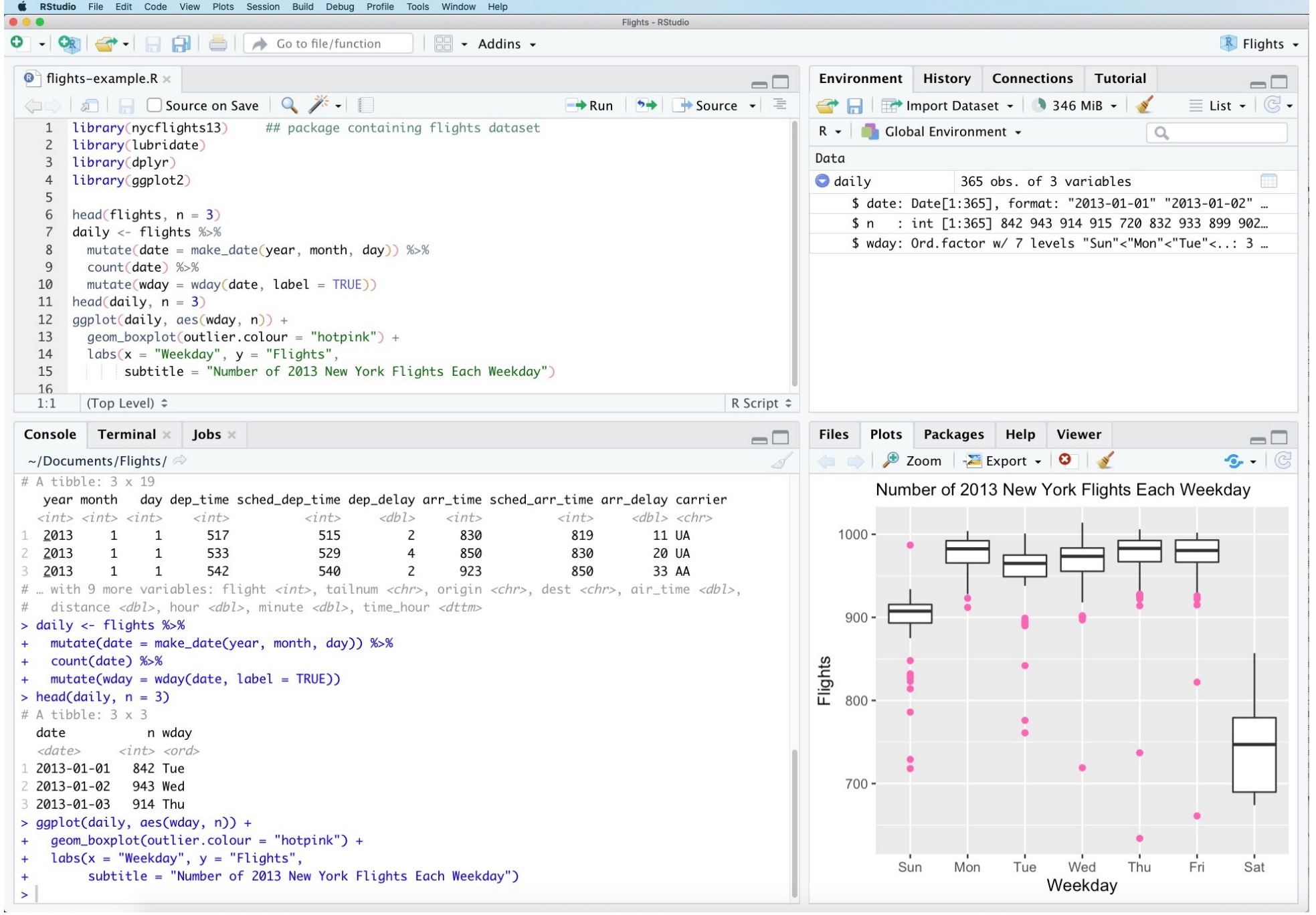
Structures that containerize data/variables and code/methods

Programming Environments, Code Editors, IDEs





Rstudio


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Flights - RStudio
Go to file/function
Addins
Flights
flights-example.R
Source on Save
Run
Source
1 library(nycflights13) ## package containing flights dataset
2 library(lubridate)
3 library(dplyr)
4 library(ggplot2)
5
6 head(flights, n = 3)
7 daily <- flights %>%
8 mutate(date = make_date(year, month, day)) %>%
9 count(date) %>%
10 mutate(wday = wday(date, label = TRUE))
11 head(daily, n = 3)
12 ggplot(daily, aes(wday, n)) +
13 geom_boxplot(outlier.colour = "hotpink") +
14 labs(x = "Weekday", y = "Flights",
15 subtitle = "Number of 2013 New York Flights Each Weekday")
16
1:1 (Top Level) R Script
Environment History Connections Tutorial
Import Dataset 346 MiB List
R Global Environment
Data
daily 365 obs. of 3 variables
$ date: Date[1:365], format: "2013-01-01" "2013-01-02" …
$ n : int [1:365] 842 943 914 915 720 832 933 899 902…
$ wday: Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 3 …
Console Terminal Jobs
~/Documents/Flights/
A tibble: 3 x 19
year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
<int> <int> <int> <int> <int> <dbl> <int> <int> <dbl> <chr>
1 2013 1 1 517 515 2 830 819 11 UA
2 2013 1 1 533 529 4 850 830 20 UA
3 2013 1 1 542 540 2 923 850 33 AA
… with 9 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
> daily <- flights %>%
+ mutate(date = make_date(year, month, day)) %>%
+ count(date) %>%
+ mutate(wday = wday(date, label = TRUE))
> head(daily, n = 3)
A tibble: 3 x 3
date n wday
<date> <int> <ord>
1 2013-01-01 842 Tue
2 2013-01-02 943 Wed
3 2013-01-03 914 Thu
> ggplot(daily, aes(wday, n)) +
+ geom_boxplot(outlier.colour = "hotpink") +
+ labs(x = "Weekday", y = "Flights",
+ subtitle = "Number of 2013 New York Flights Each Weekday")
>
Files Plots Packages Help Viewer
Zoom Export
Number of 2013 New York Flights Each Weekday
1000
900
800
700
Flights
Sun Mon Tue Wed Thu Fri Sat
Weekday

Download and Install R

Windows: https://cran.r-project.org/bin/windows/base/

Mac: https://cran.r-project.org/bin/macosx/

Then Download and Install Rstudio

https://www.rstudio.com/products/rstudio/download/#download