Introduction to Programming

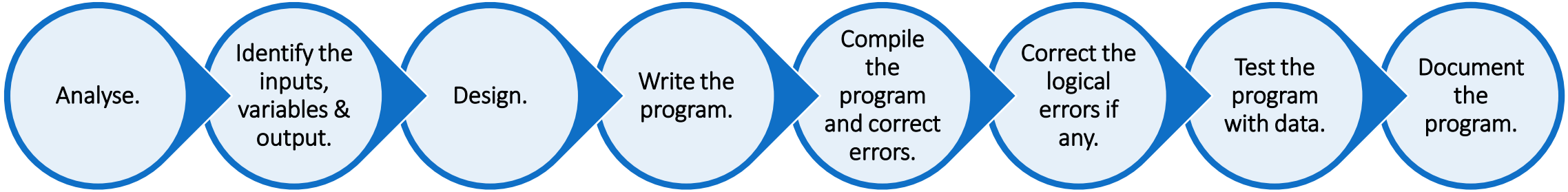
Sarath Tv

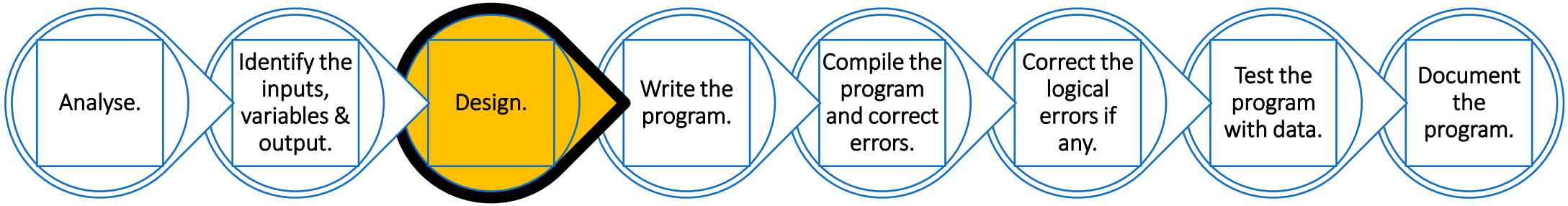
Agenda

Basics of programming.

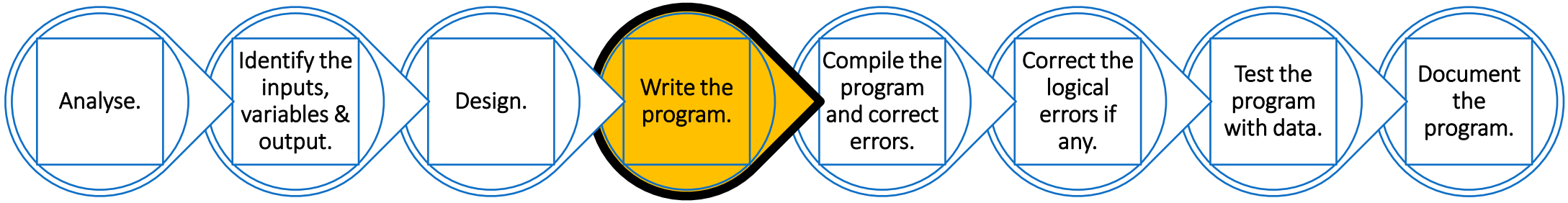
Introduction to C programming.

PROGRAMMING METHODOLOGY





- Pseudocodes :
  - *Plan your code.*
  - *Research on the problem.*
  - *Main topics/major talking points.*
- Flow charts
- Write up chronologically what the code should do.
- Functionality planning.
  - *Write the main feature you want in your program.*
  - Within each feature ,identify the necessary steps.
  - *Higher abstraction.*

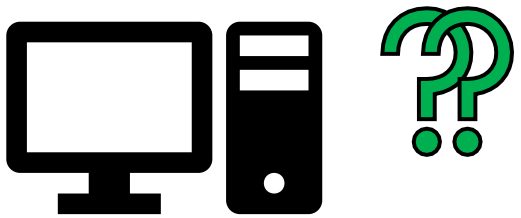


- Tool.
- Language.
- Compiler.
- Functions.
- Standards.
- Convention.
- Readability.
- Modular.

- General purpose machine.
- Any computational task.
- Program -Give it set of right instructions for the tasks.
- Manage hardware resources .
- Application programs.

What does the computer understand?

- Binary.
- Easy to physically implement.
- Machine code.
- Low level language
- High level language
- Rules/syntax



Lets eat, kids

Lets eat kids

Languages

- Machine language.
  - Feeding computer with 100000001111111100010011.
  - Machine language are specific to architecture of CPU. – LOW level
- Low Level Languages
  - Closer to machine language.
- High Level Languages
  - Abstracted from machine arch.
  - Closer to human languages.
  - More elements of natural language.
- Even if we write in these high level language ,the CPU can run or machine language.
- Complied language. – Program -Complier.- C, C++
- Interpreted Language. Don't generate executable. –within the interpreter- Python
- No good or bad language.

```
001010111000011
100010111001000
1011101000000000000000
```

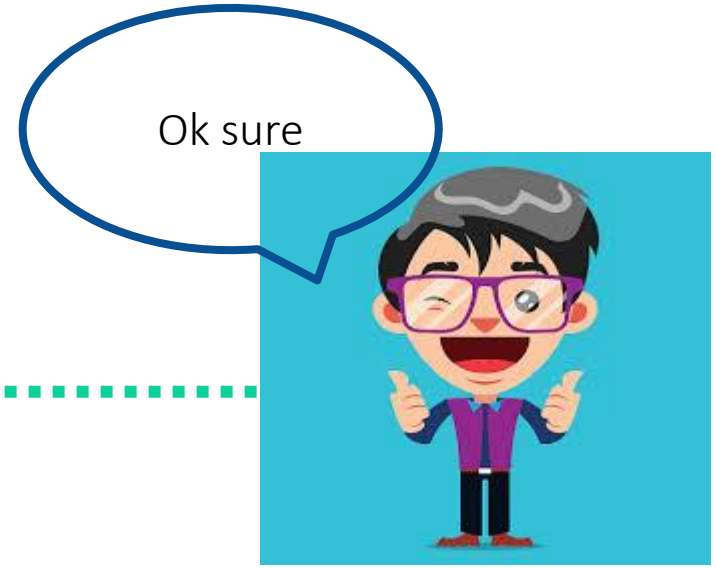
```
SUB AX,BX
MOV CX,AX
MOV DX,0
```

```
numbers = range(1, 50)
chosen = []

while len(chosen) < 6:
    number = random.choice(numbers)
    numbers.remove(number)
    chosen.append(number)

chosen.sort()

print "This week's numbers are", chosen
print "The bonus ball is", random.choice(numbers)
```

Compiler

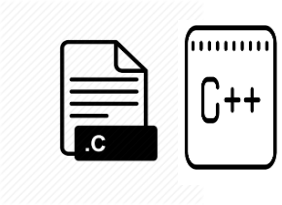

Mach diese Aufgabe


Hello
Bonjour


Ok sure

Compiler


IOIOIOIO
OIOIOIOI


.C
C++


Write the program.

Coding Standards

A coding standard makes sure that all the developers working on the project are following certain *specified guidelines*.

The code can be easily understood and *proper consistency* is maintained

Consistency has a positive impact on the quality of the program and one should maintain it while coding.

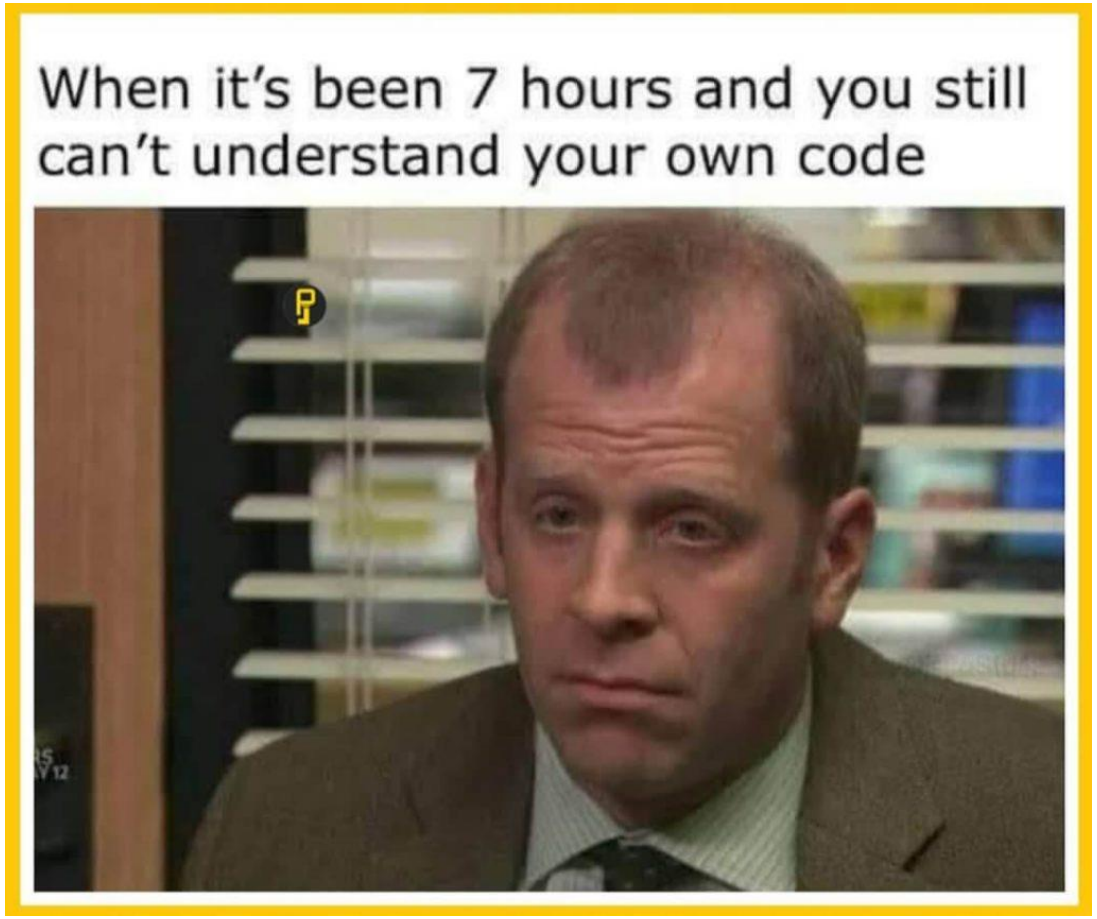
The guidelines are *homogeneously* followed across different levels of the system and they do not contradict each other.

The **finished program code should look like that it has been written by a single developer, in a single session**.

- The code should be *easy to be read*, for this:
  - Make use of *indentation for indicating the start and end of the control structures* along with a clear specification of where the code is between them
- There should be **consistency in the naming convention** of the variables throughout the code. Also, the data should be described that is there in the code
- **Name the functions** according to what they perform
- The code should be such that **one should be able to understand** it even after returning to it after **some time gap**, without that person having to look at every line of it
- Follow a **specific method for commenting** on the work.

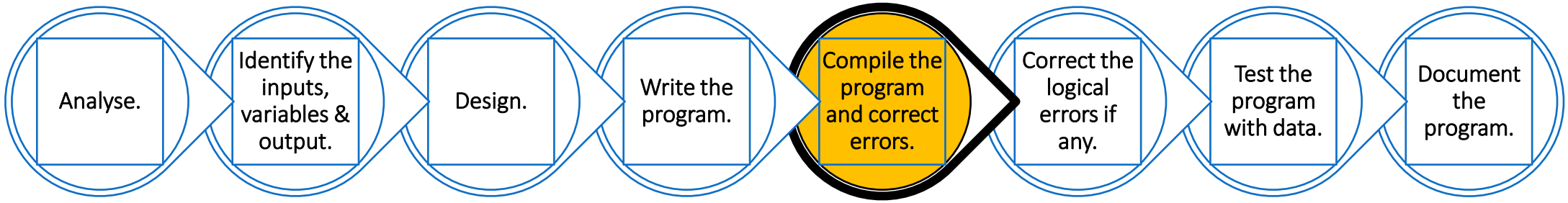
Best practices that are used to write better codes

- Code Comment.
- Use of Indentation
- Avoid Commenting on Obvious Things
- Grouping Code
- Proper and Consistent Scheme for Naming
  - **CamelCase**: This can be used for naming where the first letter of each word is capitalized except for the first word.
  - **UnderScore**: Name your function using an underscore between the words.
- Deep nesting structure should be avoided
- Proper organization of files and folder.
- Source code readability



Modular coding

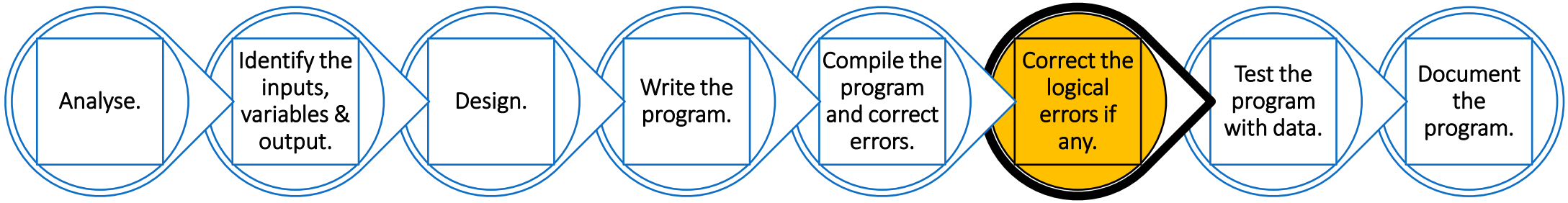
- Don’t clutter your Main application file.
- Use separate files.
- Write as functions.
- Use of file for other projects.



- Editor
- Compiler

Or

- IDE(Integrated Development Environment)
- IDE – Highlights syntax errors.



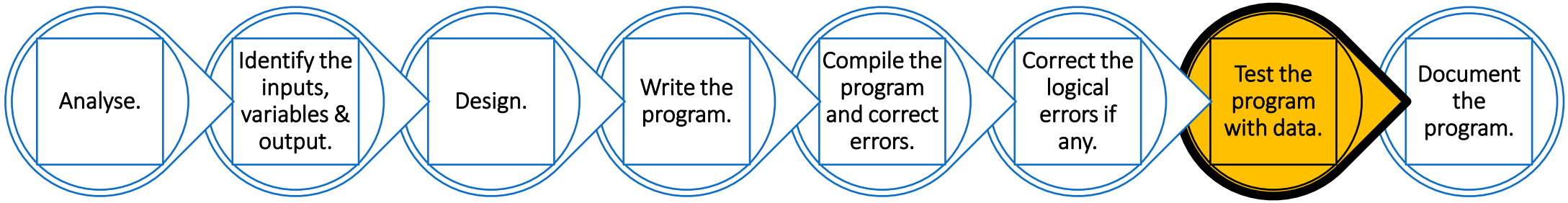
- How to debug logical errors???
- ✓ Print statements.
- ✓ Breakpoints.

- Once you spot the line which causes the error what should I do!
- ✓ Delete it ???
- ✓ Comment the code?

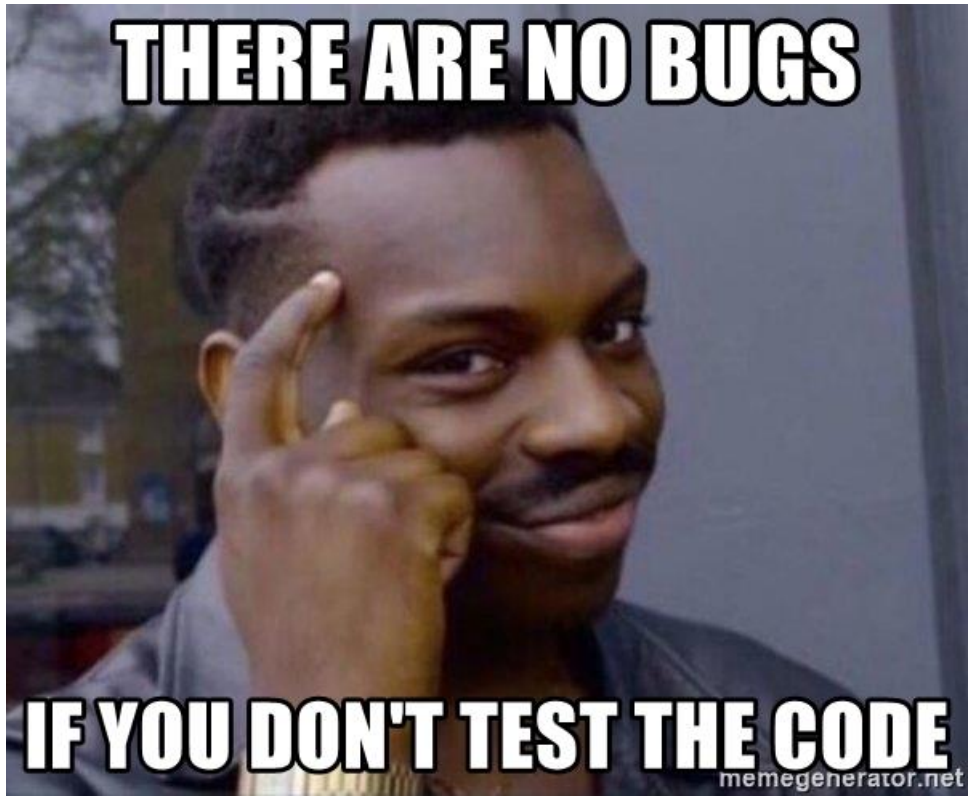
Preventing Errors

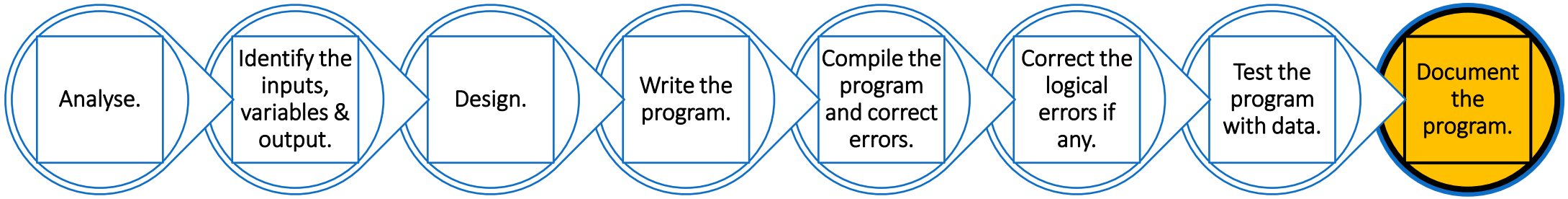
- ✓Back up codes.
- ✓Version managers –Git, subversions.
- ✓Helps you to get back checkpoints.
- ✓Run it thoroughly before backing it up.-Testing.
- ✓Testing Frameworks.

Correct the logical errors if any.



- System Testing.
- Create a scenario where you know the output for a given input.
- Test your program with this data as input.
- Compare expected and actual output.

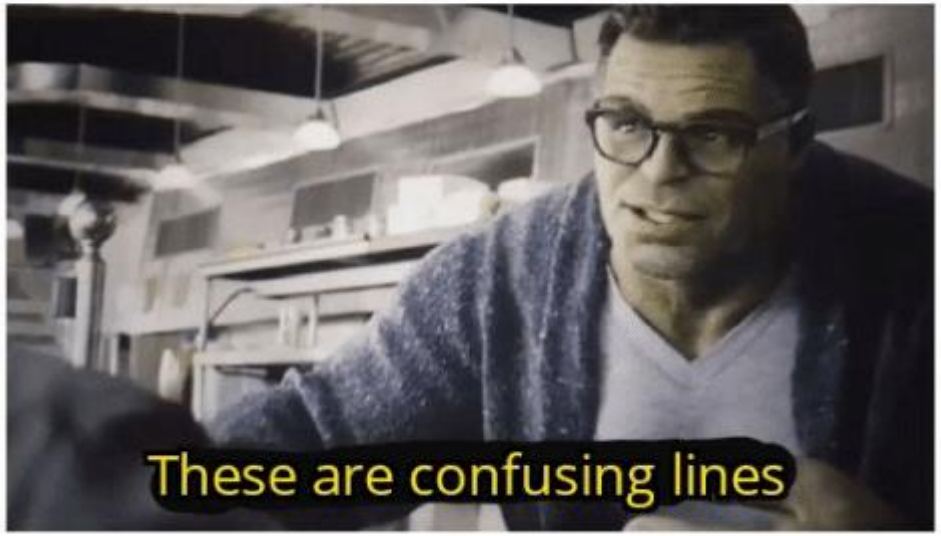




- One of the most important yet ignored step.
- Tools for extracting from source code to generate well documented project.

Me: writes code with no documentation

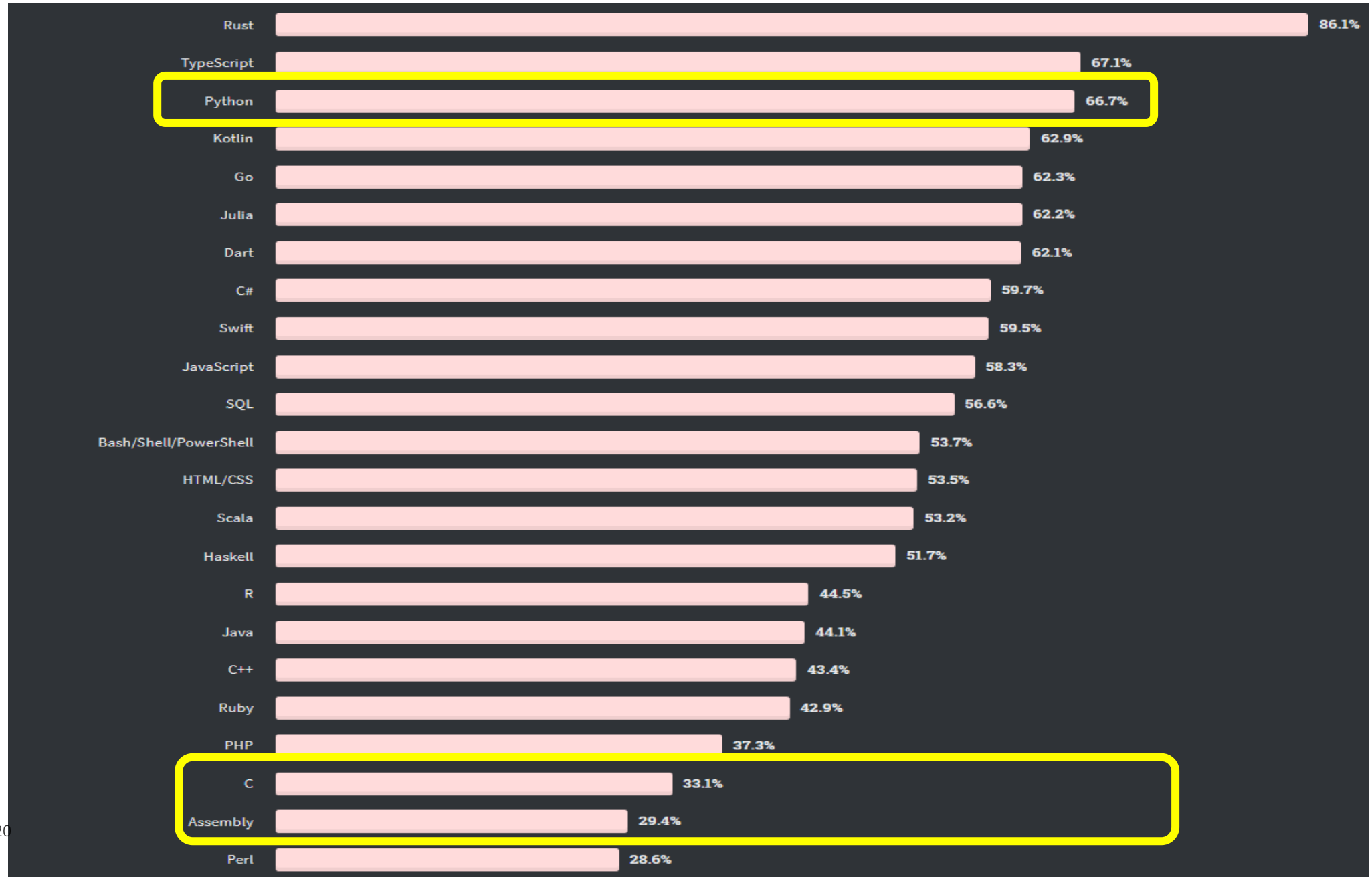
Me: *one month later*



Did you document your code: well no but actually no.

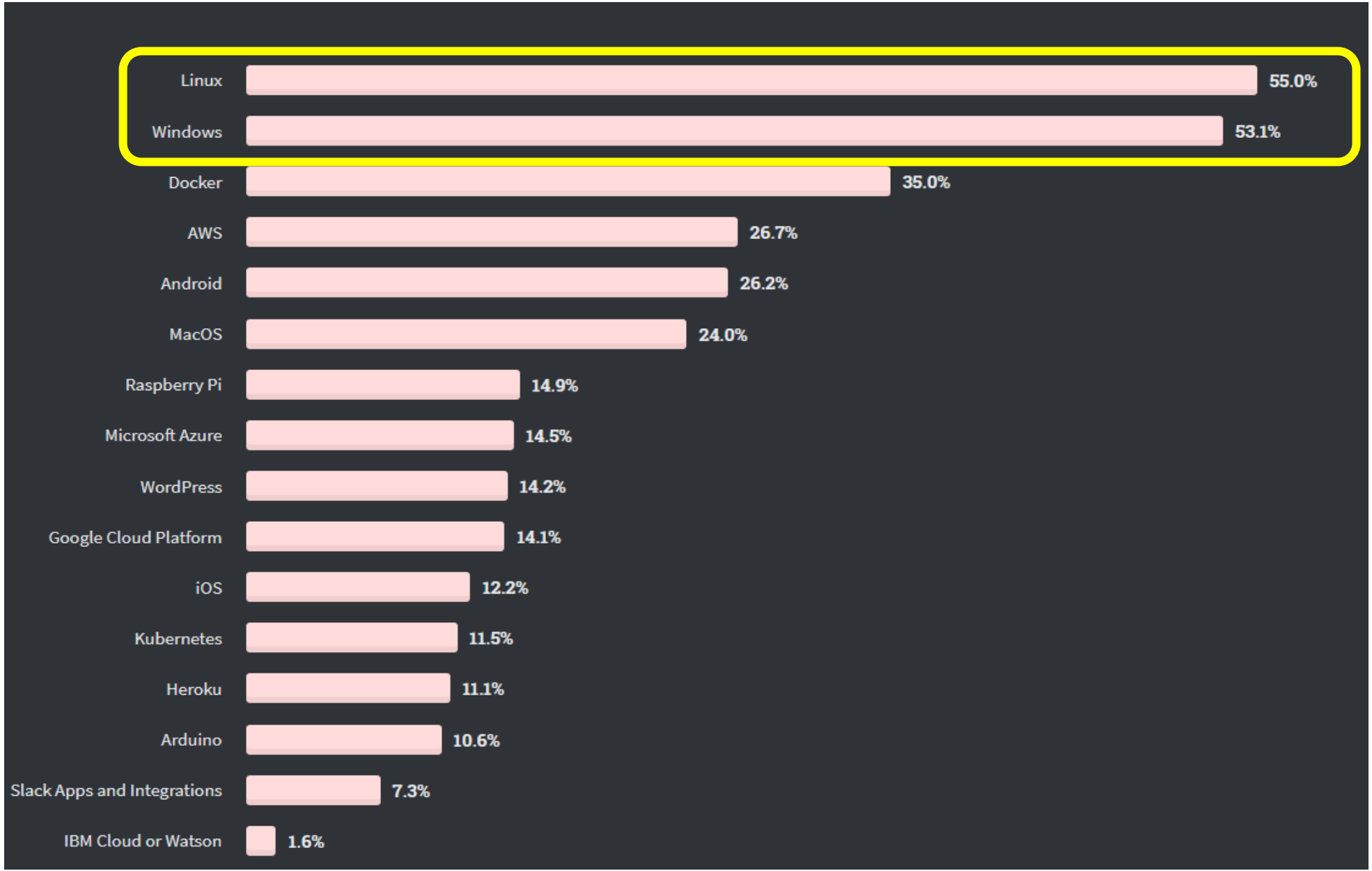
Introduction to C

Most Loved

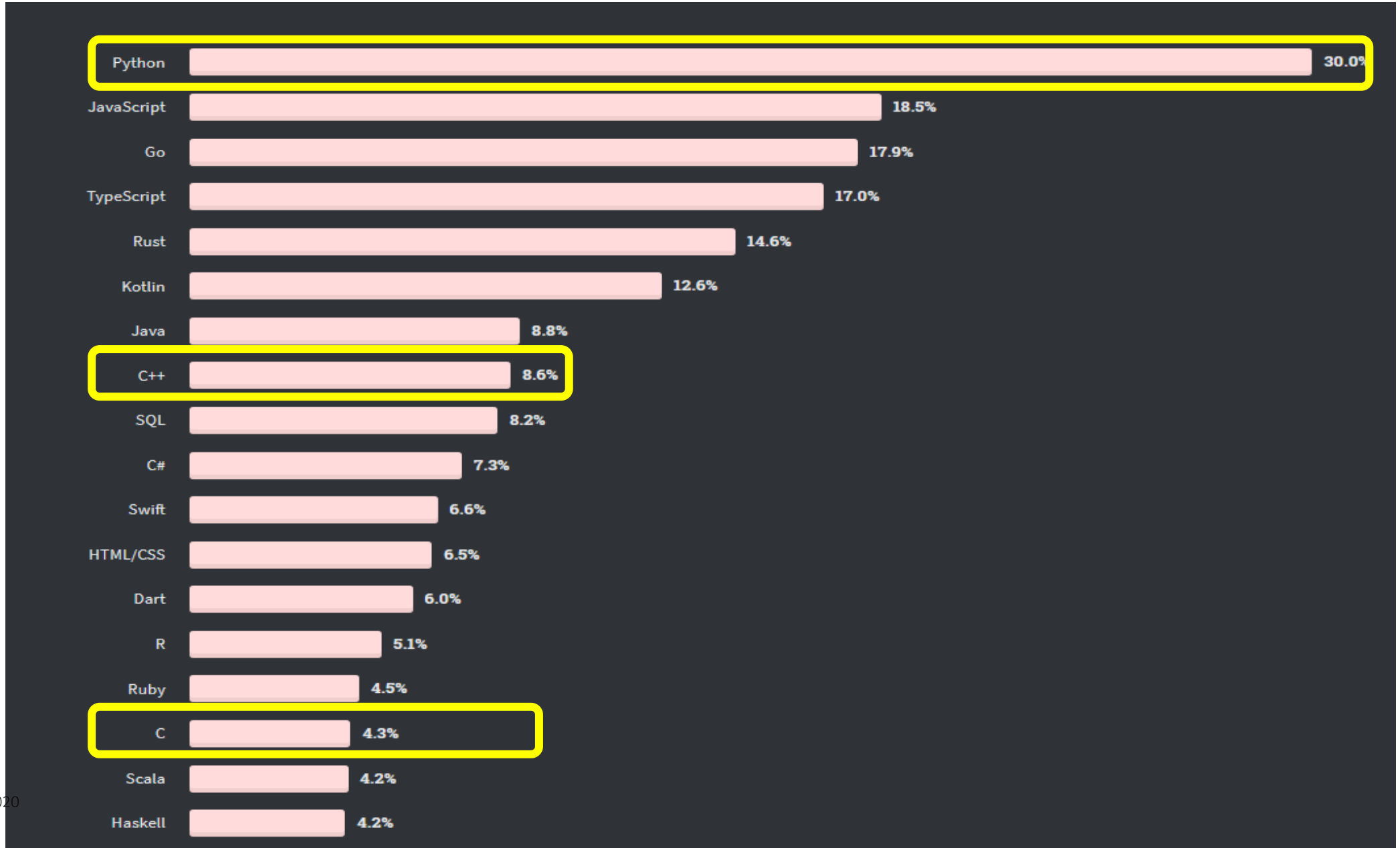

Rust 86.1%
TypeScript 67.1%
Python 66.7%
Kotlin 62.9%
Go 62.3%
Julia 62.2%
Dart 62.1%
C# 59.7%
Swift 59.5%
JavaScript 58.3%
SQL 56.6%
Bash/Shell/PowerShell 53.7%
HTML/CSS 53.5%
Scala 53.2%
Haskell 51.7%
R 44.5%
Java 44.1%
C++ 43.4%
Ruby 42.9%
PHP 37.3%
C 33.1%
Assembly 29.4%
Perl 28.6%

Most Dreaded

Platforms


Linux 55.0%
Windows 53.1%
Docker 35.0%
AWS 26.7%
Android 26.2%
MacOS 24.0%
Raspberry Pi 14.9%
Microsoft Azure 14.5%
WordPress 14.2%
Google Cloud Platform 14.1%
iOS 12.2%
Kubernetes 11.5%
Heroku 11.1%
Arduino 10.6%
Slack Apps and Integrations 7.3%
IBM Cloud or Watson 1.6%

Most Wanted


Python 30.0%
JavaScript 18.5%
Go 17.9%
TypeScript 17.0%
Rust 14.6%
Kotlin 12.6%
Java 8.8%
C++ 8.6%
SQL 8.2%
C# 7.3%
Swift 6.6%
HTML/CSS 6.5%
Dart 6.0%
R 5.1%
Ruby 4.5%
C 4.3%
Scala 4.2%
Haskell 4.2%

IDE.

Your first project.

Basic c structure.

Data types.

Loops.

Functions.

Multiple files.