100 Programming Languages

Resources

Learn all the Languages!!!

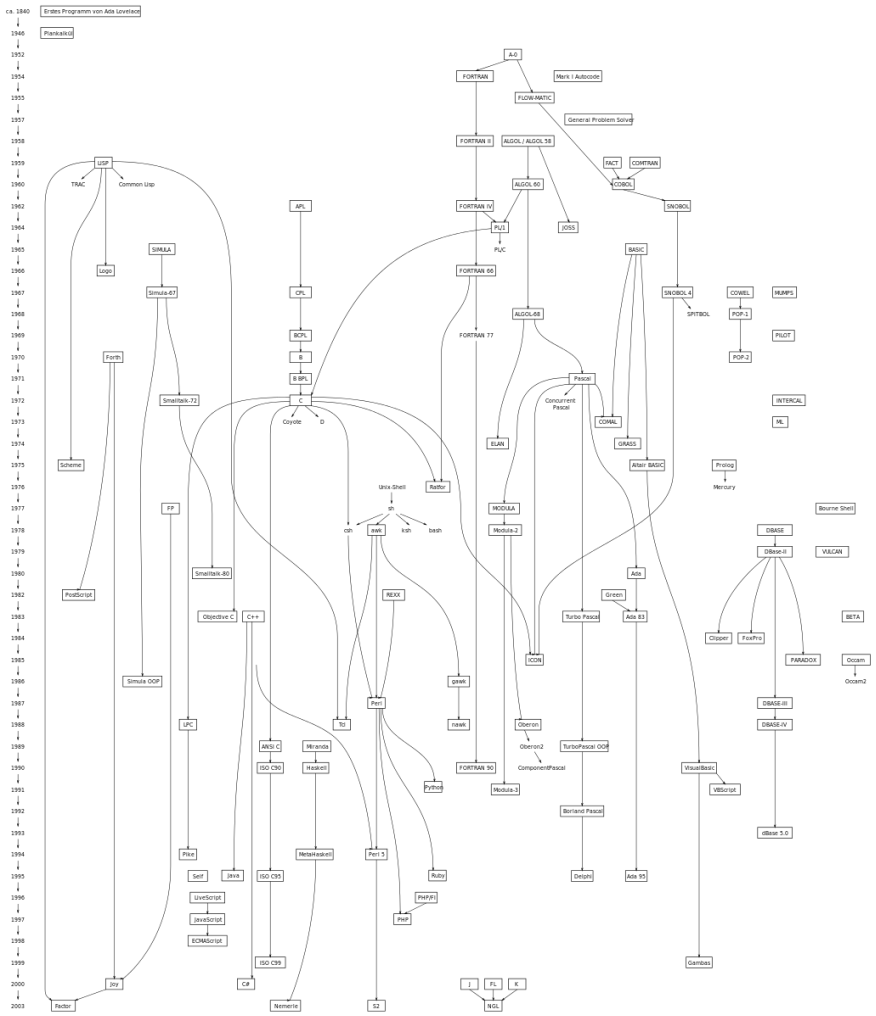
Collection of all the Languages!!!

Floating Point Math in multiple Languages!!!

Adobe Flash Substitute

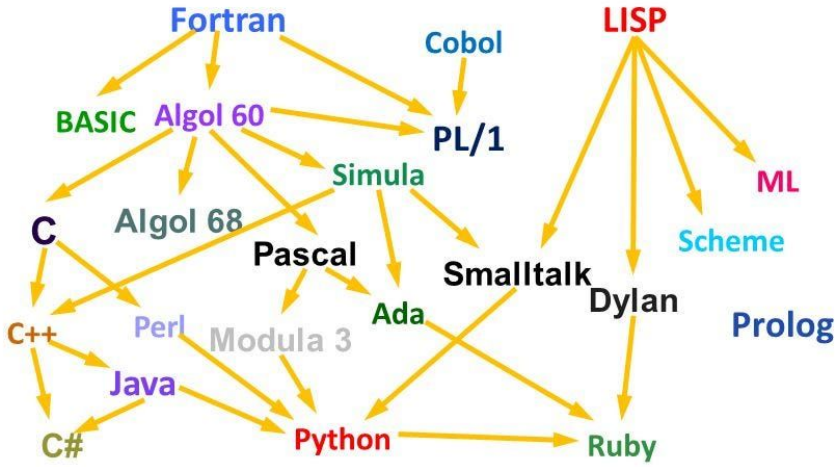
List of C Family Languages

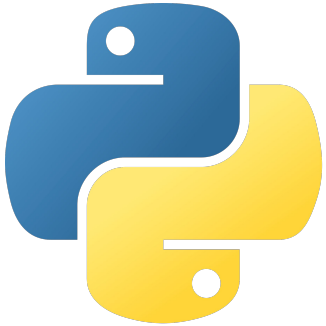
The Evolution of Programming Languages is quite long and complex



A family tree of languages

Some of the 2400 + programming languages





Python

Popularity: **1**

Year Created: **1991**

Type: **GPL, High-Level, PP, OOP**

Uses: **Security, Web Development, AI, Data Science, Scientific Computing, Scripting**

python

Copy code

```python
print("Hello, World!")
```

C

C

Popularity: **2**

Year Created: **1972**

Type: **GPL, PP**

Uses: **OS (Windows), Kernel (Windows), Embedded software, Database software**

c

Copy code

```c
#include <stdio.h>

int main() {
   printf("Hello, World!");
   return 0;
}
```

Java

Popularity: **3**

Year Created: **1995**

Type: **GPL, High-Level, OOP**

Uses: **Android Applications, Web Applications, Desktop GUI Applications, Cloud Applications, AI**

typescript

Copy code

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

C++

Popularity: **4**

Year Created: **1985**

Type: **GPL, High Level, (C Family), OOP**

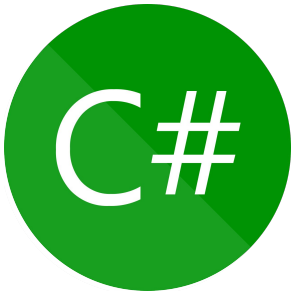
Uses: **OS (Windows), Embedded systems, Database software, Game Dev, AI, Web Browsers, Distributed Systems, VR, Compilers**

c

Copy code

```c
#include <iostream>

int main() {
    std::cout << "Hello, World!";
    return 0;
}
```



C#

Popularity: **5**

Year Created: **2000**

Type: **GPL, High Level, (.NET Family), OOP**

Uses: **.NET Framework, Windows applications, Web, Mobile & Desktop Applications, Cloud, Machine Learning**

csharp

Copy code

```csharp
using System;

class HelloWorld {
    static void Main() {
        Console.WriteLine("Hello, World!");
    }
}
```

Visual Basic (VB.NET)

Popularity: **6**

Year Created: **2002**

Type: **GPL, OOP (.NET Family)**

Uses: **Windows Applications, Web Applications, Mobile Applications, In-House Software**

vbnet

Copy code

```vbnet
Module HelloWorld
    Sub Main()
        Console.WriteLine("Hello, World!")
    End Sub
End Module
```

JavaScript

Popularity: **7**

Year Created: **1995**

Type: **FPL, High-Level, JIT, OOP**

Uses: **Web Development, Web Applications, Web Servers, Desktop Apps**

javascript

Copy code

```javascript
console.log("Hello, World!");
```

SQL (Structured Query Language)

Popularity: **8**

Year Created: **1974**

Type: **DSL**

Uses: **Databases, Data Manipulation & Analysis**

sql

Copy code

```sql
SELECT * FROM customers;
```

PHP

Popularity: **9**

Year Created: **1993**

Type: **GPL, SL, OOP**

Uses: **Web Servers (Facebook), Web Development**

php

Copy code

```php
<?php
echo "Hello, World!";
?>
```

Go (Golang)

Popularity: **10**

Year Created: **2009**

Type: **GPL, High Level, PP**

Uses: **Web Development (Google)**

go

Copy code

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello, World!")
}
```

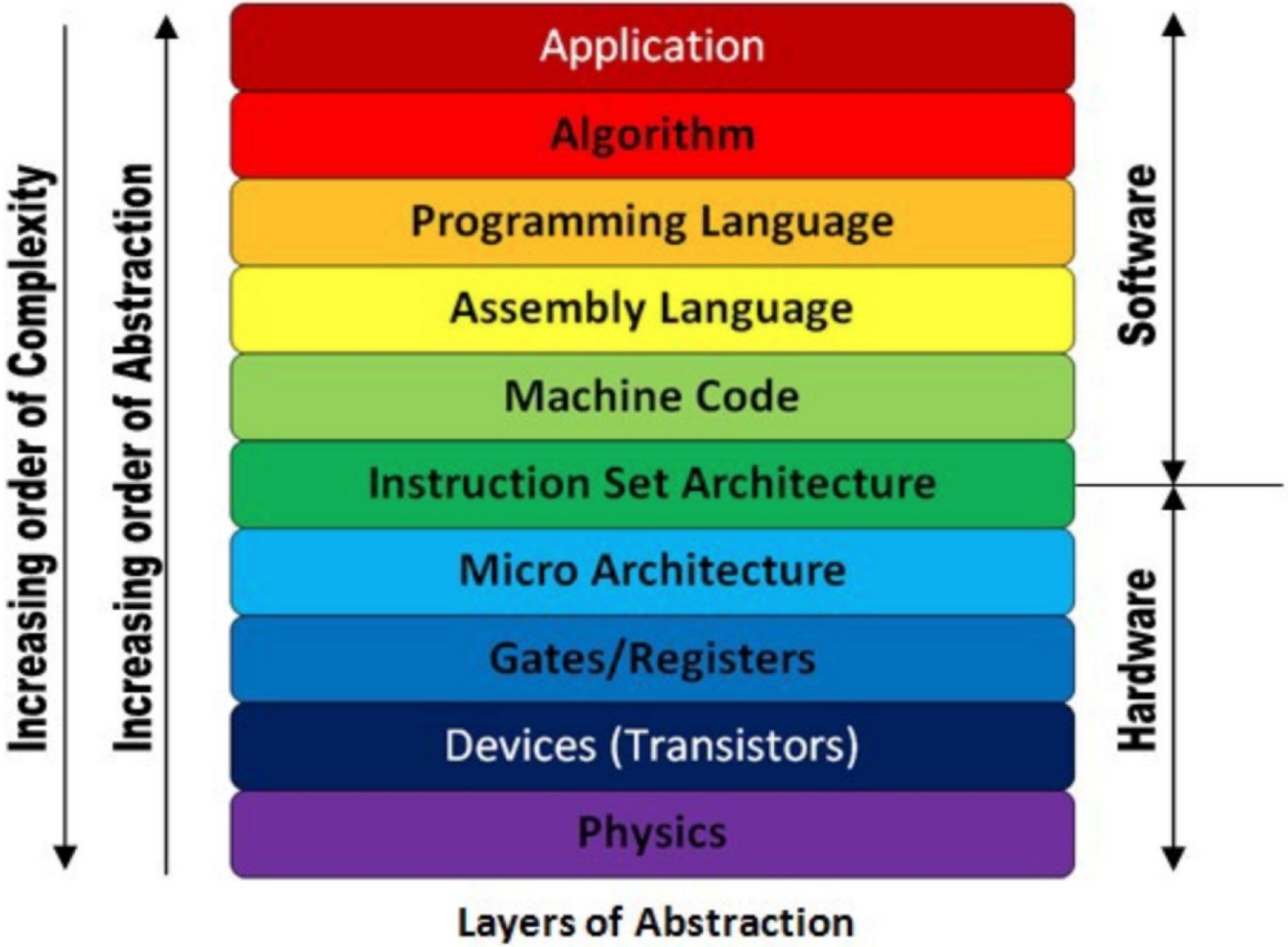
Assembly Language

Popularity: **11**

Year Created: **1947**

Type: **HDL, Low Level**

Uses: **Manipulate Computer Hardware, Processor, Compiler**


Increasing order of Complexity
Increasing order of Abstraction
Application
Algorithm
Programming Language
Assembly Language
Machine Code
Instruction Set Architecture
Micro Architecture
Gates/Registers
Devices (Transistors)
Physics
Software
Hardware
Layers of Abstraction

vbnet

Copy code

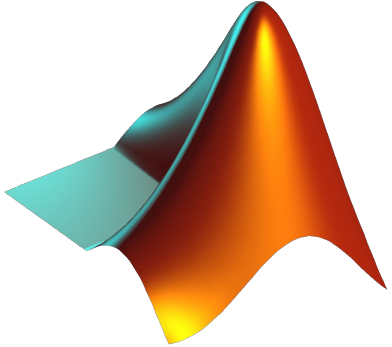
```
section .data
    msg db 'Hello, World!',0
section .text
    global _start
_start:
    ; write message to stdout
    mov eax, 4        ; system call for write
    mov ebx, 1        ; file descriptor for stdout
    mov ecx, msg      ; message to print
    mov edx, 13       ; message length
    int 0x80          ; call kernel

    ; exit program
    mov eax, 1        ; system call for exit
    xor ebx, ebx      ; exit code 0
    int 0x80          ; call kernel
```

This program first defines a data section where the message to print is stored as a null-terminated string. Then, it defines a text section where the program starts executing from the `_start` label.

In the `_start` section, the program first calls the `write` system call to print the message to stdout, using the x86 registers `eax`, `ebx`, `ecx`, and `edx` to pass the necessary arguments. Then, it calls the `exit` system call to terminate the program with an exit code of 0.

When you assemble and run this program, you should see the message "Hello, World!" printed to the console. Note that the exact syntax and system call numbers may vary depending on the operating system and architecture you are using.



MATLAB

Popularity: **12**

Year Created: **1984 (70s)**

Type: **GPL, High Level, OOP**

Uses: **Mathematics, Engineering, Science, Simulations**

```css
a = 10;
b = 20;
c = a + b;
disp(c);
```

MATLAB R2021a - academic use

HOME | PLOTS | APPS | EDITOR | PUBLISH | VIEW

Search Documentation

New | Open | Save | Find Files | Compare | Print — FILE

Go To | Find — NAVIGATE

Insert | Comment | Indent — EDIT

Breakpoints — BREAKPOINTS

Run | Run and Advance | Run Section | Advance | Run and Time — RUN

Current Folder

Name

GS2DAnd3DPlotsExample.mlx

PlotSphere.m

Details

mySin.m | PlotSphere.m

```
% Define the radius of the sphere
r = 1;

% Create a grid of x, y, and z values
[x,y,z] = sphere(50);

% Scale the grid by the radius
x = r*x;
y = r*y;
z = r*z;

% Create a 3D plot of the sphere
surf(x,y,z)
axis equal
title('Sphere')
xlabel('X')
ylabel('Y')
zlabel('Z')
```

Command Window

```
>> y = mySin(pi/4, 10)

y =

    0.7071

>> PlotSphere
>>
```

Workspace

Name	Value
r	1
x	51x51 double
y	51x51 double
z	51x51 double

UTF-8 | script | Ln 18 | Col 12

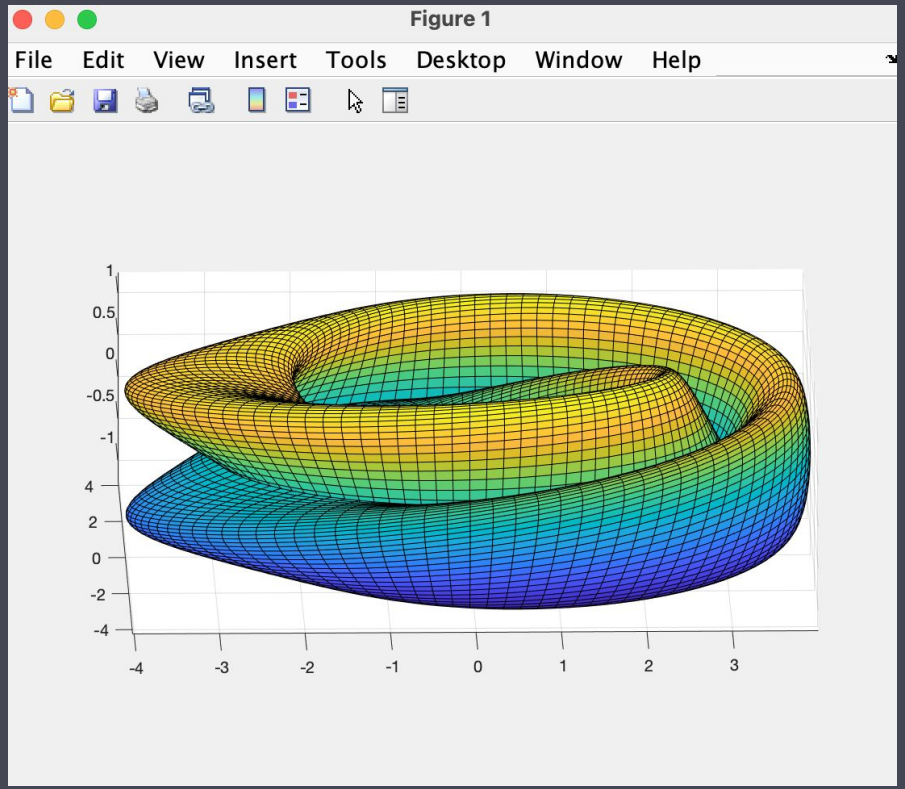


Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.5
0
-0.5
-1
4
2
0
-2
-4
-4 -3 -2 -1 0 1 2 3

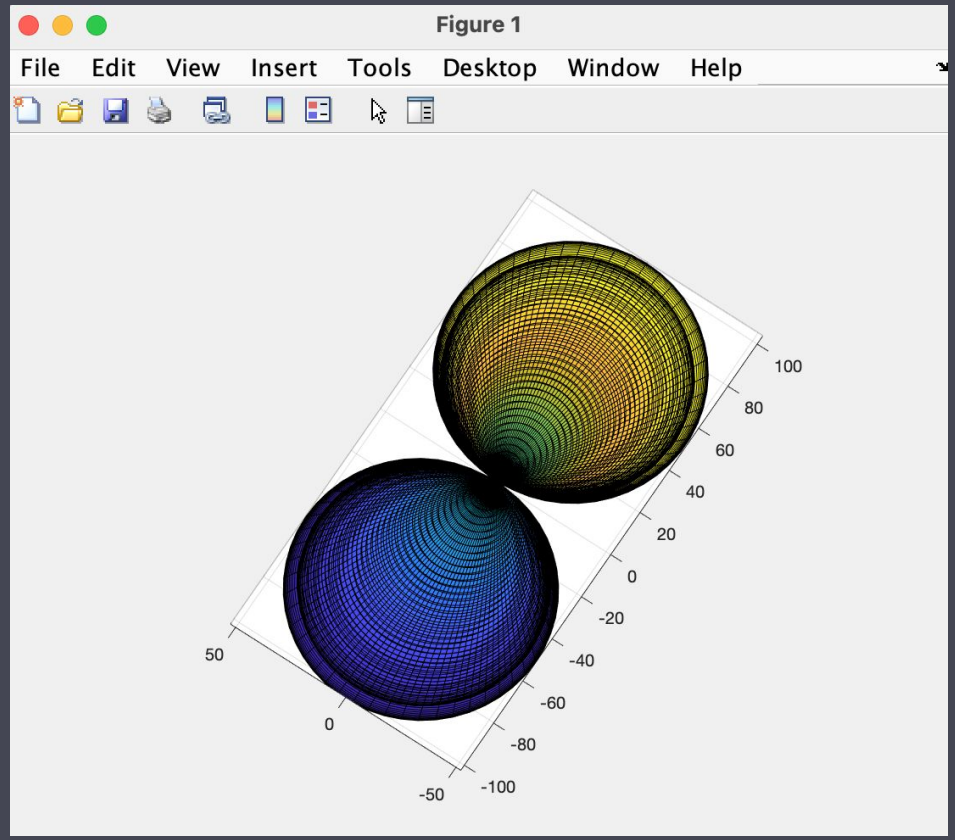


Figure 1
File Edit View Insert Tools Desktop Window Help
100
80
60
40
20
0
-20
-40
-60
-80
-100
50
0
-50

Delphi/Object Pascal

Popularity: **13**

Year Created: **1983**

Type: **GPL, OOP**

Uses: **Windows Applications, Mobile Applications, Security, Finance**

vbnet Copy code

```
unit HelloWorld;

interface

uses
  System.SysUtils, System.Classes, Vcl.Forms, Vcl.Dialogs;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  ShowMessage('Hello, World!');
end;

end.
```

When you run this program, it creates a new form and displays a message box with the text "Hello, World!" on it.

This program first defines a new form (`TForm1`) with a single event handler (`FormCreate`). In the `FormCreate` method, it calls the `ShowMessage` function to display a message box with the text "Hello, World!". The `$R *.dfm` line is a compiler directive that includes the form's resource file.

Note that this program was written using Delphi's Visual Component Library (VCL), which provides a set of pre-built components for building graphical user interfaces.

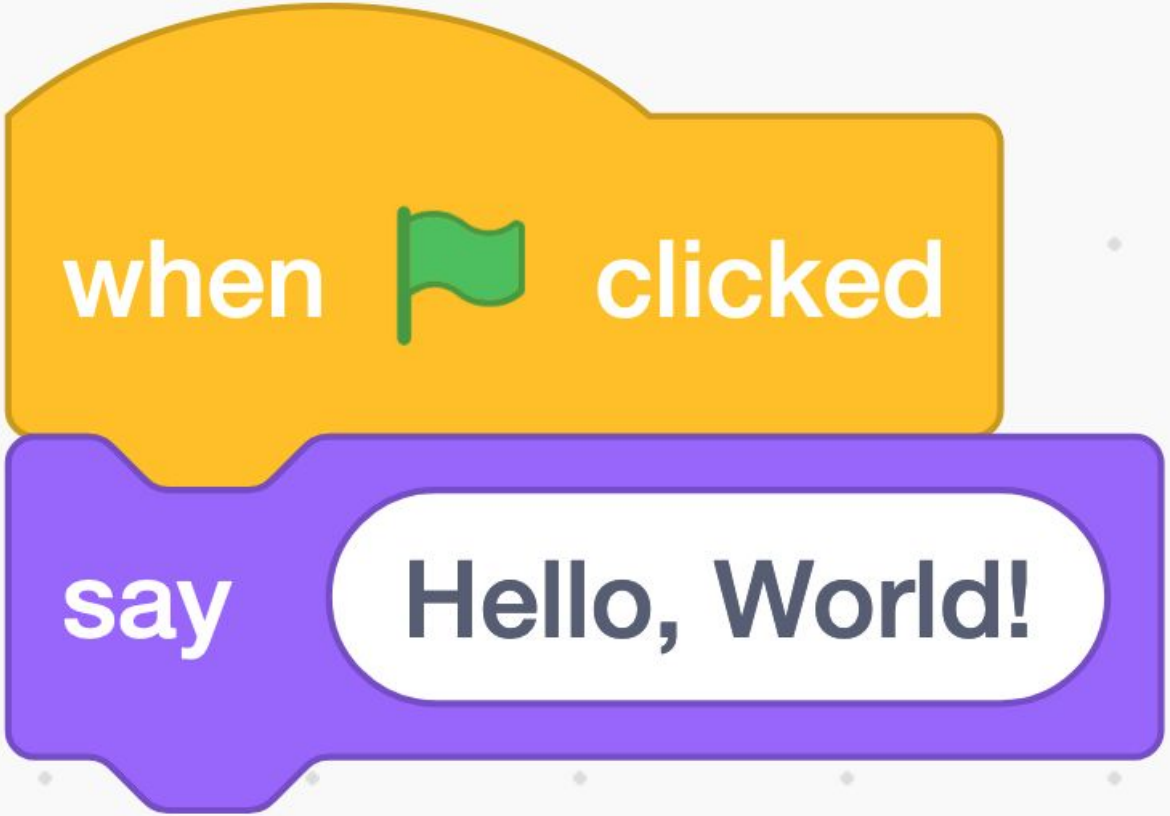
Scratch

Popularity: **14**

Year Created: **2003**

Type: **Visual**

Uses: **Used For Kids To Code, Games, Stories, Animations**


when clicked
say
Hello, World!

Classic Visual Basic

Popularity: **15**

Year Created: **1991**

Type: **GPL, OOP**

Uses: **COM (Microsoft), Desktop Apps**

vbnet

Copy code

```vbnet
Private Sub Form_Load()
    MsgBox "Hello, world!"
End Sub
```

R

Popularity: **16**

Year Created: **1993**

Type: **FPL, SC,**

Uses: **Statistical Computing & Graphics**

r

Copy code

```r
# Define a vector of numbers
numbers <- c(2, 4, 6, 8, 10)

# Compute the mean of the numbers
mean_numbers <- mean(numbers)

# Print the mean to the console
print(mean_numbers)
```

Fortran

Popularity: **17**

Year Created: **1957**

Type: **GPL, PP**

Uses: **Numeric Computations, Scientific Computing**

bash

Copy code

```
program add_numbers
  implicit none
  integer :: num1, num2, sum

  ! Prompt the user to enter two numbers
  write(*,*) "Enter the first number:"
  read(*,*) num1
  write(*,*) "Enter the second number:"
  read(*,*) num2

  ! Add the numbers together
  sum = num1 + num2

  ! Print the sum to the console
  write(*,*) "The sum of", num1, "and", num2, "is", sum

end program add_numbers
```

This program starts by defining the `add_numbers` program using the `program` keyword. It also includes an `implicit none` statement, which forces the programmer to declare all variables explicitly.

Next, the program declares three integer variables using the `integer` keyword: `num1`, `num2`, and `sum`.

The program then prompts the user to enter two numbers using the `write()` function to display a message to the console, and the `read()` function to read in the user's input.

After the user enters the numbers, the program adds them together and stores the result in the `sum` variable.

Finally, the program uses the `write()` function to print the sum of the two numbers to the console.

When you run this program in Fortran, it will prompt the user to enter two numbers, add them together, and print the result to the console.

Ruby

Popularity: **18**

Year Created: **1995**

Type: **FPL, High Level, OOP**

Uses: **Garbage Collection, JIT, Web applications, Application Frameworks**

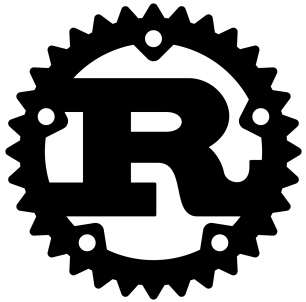
python

Copy code

```python
# Prompt the user to enter their name
print "Enter your name: "

# Read in the user's name from the console
name = gets.chomp

# Print a personalized greeting
puts "Hello, #{name}! Nice to meet you."
```



Rust

Popularity: **19**

Year Created: **2015**

Type: **FPL, High Level, PP**

Uses: **Memory Security, OS, Game Dev, Web Dev, High Performance, APIs**

rust

Copy code

```rust
use std::io;

fn main() {
    // Prompt the user to enter their name
    println!("Enter your name:");

    // Read in the user's name from the console
    let mut name = String::new();
    io::stdin().read_line(&mut name).expect("Failed to read line");

    // Print a personalized greeting
    println!("Hello, {}! Nice to meet you.", name.trim());
}
```

This program starts by importing the `io` module from the standard library using the `use` keyword.

Next, the program defines the `main()` function, which is the entry point for the program.

The program then uses the `println!()` macro to prompt the user to enter their name.

After prompting the user, the program reads in the user's name from the console using the `read_line()` function from the `io` module. The `read_line()` function reads in a line of text from the console and appends it to a `String` object.

Finally, the program uses the `println!()` macro to print a personalized greeting to the console. The greeting includes the user's name, which is trimmed using the `trim()` method to remove any whitespace characters at the beginning or end of the string.

When you run this program in Rust, it will prompt the user to enter their name, read in their input, and print a personalized greeting to the console.

Swift

Swift

Popularity: **20**

Year Created: **2014**

Type: **GPL, High Level, OOP**

Uses: **Apple OS, iOS, macOS, iPadOS, tvOS, watchOS**

swift

Copy code

```swift
import Foundation

// Prompt the user to enter their name
print("Enter your name:")

// Read in the user's name from the console
let name = readLine()!

// Print a personalized greeting
print("Hello, \(name)! Nice to meet you.")
```

This program starts by importing the `Foundation` framework, which provides access to core iOS and macOS APIs.

Next, the program uses the `print()` function to prompt the user to enter their name.

After prompting the user, the program reads in the user's name from the console using the `readLine()` function, which reads a line of text from the console and returns it as an optional `String` object.

Finally, the program uses the `print()` function to print a personalized greeting to the console. The greeting includes the user's name, which is interpolated into the string using the `\(name)` syntax.

When you run this program in Swift, it will prompt the user to enter their name, read in their input, and print a personalized greeting to the console.

sas

Popularity: **21**

Year Created: **1976**

Type: **GPL, High Level, D**

Uses: **Spreadsheets, Databases, Statistical Analysis**

kotlin

Copy code

```
/* Read in the dataset */
data mydata;
  infile 'path/to/mydata.csv' dlm=',' firstobs=2;
  input var1 var2 var3;
run;

/* Calculate the mean of var1 */
proc means data=mydata mean;
  var var1;
run;
```

In this program, we first read in a dataset called `mydata` using the `infile` statement. The `dlm=','` option specifies that the fields in the file are delimited by commas, and `firstobs=2` specifies that we want to skip the first line of the file (assuming it contains column headers).

Next, we use the `proc means` statement to calculate the mean of `var1` in the `mydata` dataset. The `mean` option tells SAS to calculate the mean, and `var var1` specifies that we want to calculate the mean of `var1`.

When we run this program, SAS will read in the `mydata` dataset, calculate the mean of `var1`, and output the result to the SAS log.

Visual FoxPro

Popularity: **22**

Year Created: **1984**

Type: **GPL, High Level, OOP**

Uses: **Desktop Applications, Databases**

```
* Create a new form
CREATE FORM myForm

* Add a button to the form
ADD OBJECT myButton AS COMMANDBUTTON WITH ;
    CAPTION "Click me!", ;
    LEFT 50, TOP 50, ;
    WIDTH 100, HEIGHT 25

* Define the button's click event
PROCEDURE myButton.Click
    MESSAGEBOX("Hello, World!")
ENDPROC

* Show the form
myForm.SHOW
```

This program creates a new form with a button labeled "Click me!" and sets its position and size. When the button is clicked, the `myButton.Click` procedure is called, which displays a message box with the text "Hello, World!". Finally, the form is displayed using the `myForm.SHOW` command.

COBOL

Popularity: **23**

Year Created: **1959**

Type: **GPL, PP**

Uses: **Business World, Payroll Programs, Government Funds, Banking**

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SQUARE-PROGRAM.
AUTHOR. YOUR NAME.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-PC.
OBJECT-COMPUTER. IBM-PC.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 NUM PIC 9(4).
01 SQUARE PIC 9(6).

PROCEDURE DIVISION.
MAIN-LOGIC.
    DISPLAY "Enter a number: ".
    ACCEPT NUM.
    COMPUTE SQUARE = NUM * NUM.
    DISPLAY "The square of " NUM " is " SQUARE.

    STOP RUN.
```

This program starts by prompting the user to enter a number using the `DISPLAY` statement. The user's input is then accepted using the `ACCEPT` statement and stored in the `NUM` variable.

The program then calculates the square of the input number using the `COMPUTE` statement and stores the result in the `SQUARE` variable. Finally, the program displays the result to the user using the `DISPLAY` statement.

Note that in COBOL, indentation is important and must be consistent throughout the program. Additionally, the `IDENTIFICATION DIVISION`, `ENVIRONMENT DIVISION`, and `DATA DIVISION` sections are required and must be included at the beginning of the program.

Perl

Popularity: **24**

Year Created: **1987**

Type: **FPL, High Level, PP**

Uses: **Game Servers, Manipulate Text Files, Interact with other programs**

perl

Copy code

```perl
#!/usr/bin/perl

print "Enter a string: ";
my $input = <STDIN>;
chomp $input;
print "You entered: $input\n";
```

This program starts by printing the message "Enter a string: " using the `print` statement. The `my` keyword is used to declare a variable named `$input` to store the user's input. The `<STDIN>` statement reads a line of text from the user and stores it in the `$input` variable.

The `chomp` function is then used to remove the newline character from the end of the input string. Finally, the program displays the input string back to the user using the `print` statement and the variable `$input`.

Note that in Perl, semicolons are used to separate statements and parentheses are used to group expressions. Also, Perl is an interpreted language, so you need to make sure that Perl is installed on your system and that the first line of the program `#!/usr/bin/perl` is correct for your system's Perl installation.

F#

Popularity: **25**

Year Created: **2005**

Type: **FPL, GPL, OOP/F**

Uses: **CLI (.NET), GPU, Machine Learning, Scientific Computing, Data**

java

Copy code

```
open System

[<EntryPoint>]
let main argv =
    Console.Write("Enter a number: ")
    let input = Console.ReadLine()
    let num = int input
    let square = num * num
    Console.WriteLine("The square of {0} is {1}", num, square)
    0 // Exit code
```

This program starts by using the `open System` statement to include the `Console` class from the `System` namespace. The `[<EntryPoint>]` attribute is used to indicate that the `main` function is the entry point for the program.

The `Console.Write` function is used to print the message "Enter a number: " to the console, and the `Console.ReadLine` function is used to read a line of text from the user.

The `int` function is then used to convert the input string to an integer, and the square of the input number is calculated using the `*` operator.

Finally, the `Console.WriteLine` function is used to display the result to the user, and the program exits with a code of `0`.

Note that F# is a functional programming language, so it uses functions as first-class values and emphasizes immutability and composition. Also, the F# compiler can infer types automatically, so explicit type annotations are often unnecessary.

[Objective-C]

Popularity: **26**

Year Created: **1984**

Type: **GPL, High Level, OOP**

Uses: **macOs, iOS**

objectivec

Copy code

```objectivec
#import <Foundation/Foundation.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        NSLog(@"Enter a number: ");
        char input[10];
        fgets(input, 10, stdin);
        int num = atoi(input);
        int square = num * num;
        NSLog(@"The square of %d is %d", num, square);
    }
    return 0;
}
```

This program starts by importing the `Foundation` framework using the `#import` statement. The `main` function is then defined, with the `@autoreleasepool` block to manage memory.

The `NSLog` function is used to print the message "Enter a number: " to the console. The `fgets` function is then used to read a line of text from the user and store it in the `input` array. The `atoi` function is used to convert the input string to an integer, which is stored in the `num` variable.

The square of the input number is calculated using the `*` operator and stored in the `square` variable. Finally, the `NSLog` function is used to display the result to the user.

Note that Objective-C is a compiled language and is often used for developing iOS and macOS applications. It is also a superset of C, so C syntax can be used in Objective-C programs.

Transact-SQL

Popularity: **27**

Year Created: **1989**

Type: **DSL**

Uses: **Databases, Data Manipulation & Analysis**

sql

Copy code

```sql
-- Create a table called "customers"
CREATE TABLE customers (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50)
);

-- Insert some data into the "customers" table
INSERT INTO customers (id, name, email)
VALUES (1, 'John Smith', 'john@example.com'),
       (2, 'Jane Doe', 'jane@example.com'),
       (3, 'Bob Johnson', 'bob@example.com');
```

This program creates a table called "customers" with three columns: "id", "name", and "email". The "id" column is set as the primary key. Then, the program inserts three rows of data into the "customers" table, specifying the values for each column in each row.

Ada

In Strong Typing We Trust

Ada

Popularity: **28**

Year Created: **1980**

Type: **GPL, High Level, OOP**

Uses: **Safety, Security, Finance**

scss

Copy code

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Hello is
   Name : String(1..20);
begin
   Put("What is your name? ");
   Get_Line(Name, Last => 20);
   Put("Hello, ");
   Put(Name);
   New_Line;
end Hello;
```

This program uses the Ada.Text_IO package to perform input and output operations. It first declares a variable called Name of type String with a maximum length of 20 characters. It then prompts the user for their name using the Put procedure and reads in their input using the Get_Line procedure, which takes two parameters: the input buffer (in this case, the Name variable) and the maximum number of characters to read (in this case, 20).

Finally, the program uses the Put procedure again to output the greeting, which consists of the string "Hello, " followed by the user's name, and a newline character to move to the next line.

Lisp

Popularity: **29**

Year Created: **1960**

Type: **High Level, FPL**

Uses: **Machine Learning, AI, Quantum Computing**

python

Copy code

```
(defun add (a b)
  (+ a b))

(format t "Enter first number: ")
(setq num1 (read))
(format t "Enter second number: ")
(setq num2 (read))

(setq sum (add num1 num2))

(format t "The sum of ~a and ~a is ~a." num1 num2 sum)
```

When run, this program will prompt the user to enter two numbers, then calculate their sum and print it to the console.

Note that this program defines a function called `add` using the `defun` keyword, which takes two arguments (`a` and `b`) and returns their sum using the `+` operator. The `setq` keyword is used to assign values to variables, and the `format` function is used to output text to the console.

Lua

Popularity: **30**

Year Created: **1993**

Type: **High Level, PP, FPL**

Uses: **Video Games (GMOD), Applications**

lua

Copy code

```lua
print("Hello, World!")
```

Haskell

Popularity: **31**

Year Created: **1990**

Type: **High Level, FPL**

Uses: **Applications, Web Frameworks, Web Servers, Web Applications**

haskell

Copy code

```haskell
main = putStrLn "Hello, World!"
```

This program will output the text "Hello, World!" to the console when executed. In Haskell, the `main` function is the entry point of the program and it must have the type `IO ()`. The `putStrLn` function takes a string argument and outputs it to the console with a newline character at the end.

PL/SQL

Popularity: **32**

Year Created: **1995**

Type: **DSL**

Uses: **Databases, Data Manipulation & Analysis**

sql

Copy code

```sql
BEGIN
  DBMS_OUTPUT.PUT_LINE('Hello, World!');
END;
```

This program uses the `DBMS_OUTPUT` package in Oracle Database to write text to the console. The `PUT_LINE` procedure writes a line of text to the console with a newline character at the end. When you execute this program in an Oracle Database environment that supports `DBMS_OUTPUT`, you should see the message "Hello, World!" printed to the console.

julia

Popularity: **33**

Year Created: **2012**

Type: **FPL, High Level,**

Uses: **Math, Numerical Computing**

julia

Copy code

```julia
for i in 1:10
    println(i)
end
```

This program uses a `for` loop to iterate over the range 1 to 10, and prints out each number using the `println()` function.

You can run this program in the Julia REPL or by saving it to a file with a `.jl` extension and running it using the command `julia filename.jl`.

Groovy

Popularity: **34**

Year Created: **2003**

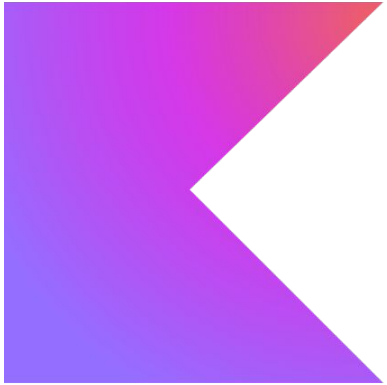
Type: **FPL, OOP (Java Family)**

Uses: **JVM, Scripts**

groovy

Copy code

```groovy
// define a class named HelloWorld
class HelloWorld {
    static void main(String[] args) {
        println("Hello, World!") // print "Hello, World!" to the console
    }
}
```



Kotlin

Popularity: **35**

Year Created: **2011**

Type: **GPL, OOP (Java Family)**

Uses: **JVM**

kotlin

Copy code

```kotlin
fun main() {
    println("Hello, World!")
}
```

Dart

Popularity: **36**

Year Created: **2011**

Type: **GPL, OOP**

Uses: **Web Apps, Mobile Apps, Javascript, Flutter**

dart

Copy code

```dart
void main() {
  for (int i = 1; i <= 10; i++) {
    print('$i ');
  }
}
```

This program uses a `for` loop to iterate over the integers from 1 to 10, and prints out each number using the `print()` function. The `$i` syntax is used to interpolate the value of `i` into the string that is printed.

CFML

Popularity: **37**

Year Created: **1995**

Type: **DSL, OOP**

Uses: **JVM, .NET Framework, Google App Engine**

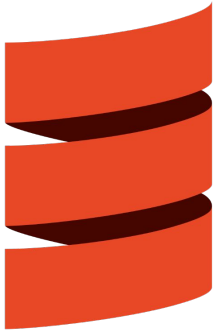
cfml

Copy code

```cfml
<cfoutput>
    <cfloop from="1" to="10" index="i">
        #i#
    </cfloop>
</cfoutput>
```

This program uses a `cfloop` tag to iterate over the integers from 1 to 10, and prints out each number using the `#i#` syntax within the `cfoutput` tag.

You can run this program in a CFML engine like Adobe ColdFusion or Lucee by saving it to a file with a `.cfm` or `.cfc` extension and running it using a web server like Apache Tomcat or IIS with the appropriate CFML engine installed.



Scala

Popularity: **38**

Year Created: **2004**

Type: **FPL, High Level, OOP (Java Family)**

Uses: **JVM, Javascript**

scala

Copy code

```scala
object Main {
  def main(args: Array[String]): Unit = {
    for (i <- 1 to 10) {
      println(i)
    }
  }
}
```

This program uses a `for` loop to iterate over the range from 1 to 10, and prints out each number using the `println()` method.

You can run this program in the Scala REPL or by saving it to a file with a `.scala` extension and running it using the `scalac` compiler followed by the `scala` command. Alternatively, you can use an IDE like IntelliJ IDEA or Eclipse to run Scala programs.

ProLog

Popularity: **39**

Year Created: **1972**

Type: **FPL, D**

Uses: **AI, Computational Linguistics**

prolog

Copy code

```prolog
count_down(N) :-
  between(1, N, X),
  write(X), nl,
  X = N.
```

scss

Copy code

```scss
?- consult('filename.pl').
?- count_down(10).
```

This program uses the `between/3` predicate to generate the numbers from 1 to N, and then prints out each number using the `write/1` predicate followed by a newline (`nl/0`). The `X = N` check ensures that the loop terminates after printing out the last number.

You can run this program in a Prolog interpreter like SWI-Prolog by loading it into the interpreter using the `consult` command and then calling the `count_down/1` predicate with an argument:

RPG

Popularity: **40**

Year Created: **1959**

Type: **GPL, High Level, PP**

Uses: **Business Applications**

sql Copy code

```
F* Hello World RPG program
F* Ask user for name and greet them
F                                       INFDS
FNAME           O    E              WORKSTN
C     *ENTRY    PLIST
C                   PARM                    NAME          10
 * Ask user for their name
C                   EVAL      NAME = ' ' + %TRIM(%UPPER(NAME)) + ','
C                   EXSR      GREET
C                   SETON                                        LR
 * Print personalized greeting
C     GREET       BEGSR
C                   EXSR      CLEAR
C                   EVAL      MSG = 'Hello, ' + NAME + ' welcome to RPG
C                   EXSR      DISPLAY_MSG
C                   SETON                                        LR
 * Clear message variable
C     CLEAR       BEGSR
C                   EVAL      MSG = ' '
C                   SETON                                        LR
 * Display message on work station file
C     DISPLAY_MSG BEGSR
C                   WRITE     MSG
C                   SETON                                        LR
```

sql Copy code

```
o World RPG program
user for name and greet them
                                INFDS
         O    E              WORKSTN
ENTRY    PLIST
             PARM                    NAME          10
user for their name
             EVAL      NAME = ' ' + %TRIM(%UPPER(NAME)) + ','
             EXSR      GREET
             SETON                                        LR
t personalized greeting
REET       BEGSR
             EXSR      CLEAR
             EVAL      MSG = 'Hello, ' + NAME + ' welcome to RPG programming!'
             EXSR      DISPLAY_MSG
             SETON                                        LR
r message variable
LEAR       BEGSR
             EVAL      MSG = ' '
             SETON                                        LR
lay message on work station file
ISPLAY_MSG BEGSR
             WRITE     MSG
             SETON                                        LR
```

When you run this program and it's executed, it will ask the user for their name and then greet them with a personalized message.

Note that this program uses RPG IV syntax and assumes that you have a display file with a work station file defined that can be written to using the WRITE operation code.

OpenEdge ABL

Popularity: **41**

Year Created: **1981**

Type: **GPL, OOP**

Uses: **Web Applications, Systems**

sql

Copy code

```sql
/* Define the procedure */
PROCEDURE HelloWorld:
    MESSAGE "Hello, World!" VIEW-AS ALERT-BOX INFORMATION.
END.

/* Call the procedure */
RUN HelloWorld.
```

Copy code

```
pro HelloWord.p
```

Bash

Popularity: **42**

Year Created: **1989**

Type: **GPL, CL, PP**

Uses: **Unix Shell, OS**

bash

Copy code

```bash
echo "Hello, World!"
```

ABAP

Popularity: **43**

Year Created: **1983**

Type: **GPL, High Level, PP**

Uses: **Business Apps, SAP apps**

vbnet

Copy code

```
REPORT HelloWorld.

PARAMETERS name TYPE string LOWER CASE.

WRITE: 'Hello,', name, 'welcome to ABAP programming!'.
```

When you run this program, it will ask the user for their name and then greet them with a personalized message.

Note that this program assumes that you have access to an SAP system with the ABAP development environment. You can copy and paste the code into the ABAP Editor and run it from there.

Logo

Popularity: **44**

Year Created: **1967**

Type: **FPL, E, PP**

Uses: **Education, Drawings**

scss

Copy code

```scss
REPEAT 4 [  FORWARD 100  RIGHT 90]
```

AWK

Popularity: **45**

Year Created: **1977**

Type: **DSL, PP**

Uses: **Data, Text Processing**

kotlin

Copy code

```
awk '{ print $1 }' data.txt
```

When you run this program, it will read the `data.txt` file and print the first field of each line to the console.

Typescript

Popularity: **46**

Year Created: **2012**

Type: **FPL, High Level (Javascript Family), OOP**

Uses: **Web Apps, Server-side Apps**

typescript

Copy code

```typescript
// Define a function that prints a message to the console
function sayHello(name: string) {
  console.log(`Hello, ${name}!`);
}

// Call the function with a name argument
sayHello("World");
```

This program defines a function called `sayHello` that takes a string argument `name` and uses the `console.log` method to print a message to the console. The program then calls the `sayHello` function with the argument `"World"`, which prints the message "Hello, World!" to the console.

Note that TypeScript is a superset of JavaScript, so any valid JavaScript code is also valid TypeScript code. However, TypeScript adds additional features and syntax to JavaScript, such as static type checking, that can help catch errors and improve code quality.

Emacs Lisp

Popularity: **47**

Year Created: **1985**

Type: **FPL, (Lisp Family)**

Uses: **Emacs Text Editor**

elisp

Copy code

```elisp
;; Define a function that computes the sum of integers from 1 to n
(defun sum-integers (n)
  "Compute the sum of integers from 1 to n."
  (let ((sum 0))
    (dotimes (i n)
      (setq sum (+ sum (1+ i))))
    sum))

;; Call the function with an argument of 10
(sum-integers 10)
```

This program defines a function called `sum-integers` that takes an integer argument `n` and computes the sum of the integers from 1 to `n`. The function uses the `let` special form to bind the variable `sum` to an initial value of 0, and then uses the `dotimes` macro to iterate `n` times and accumulate the sum of the integers in the `sum` variable. Finally, the program calls the `sum-integers` function with an argument of 10, which should return a value of 55 (the sum of the integers from 1 to 10).

D

Popularity: **48**

Year Created: **2001**

Type: **FPL, OOP**

Uses: **OS, Device Drivers, Game Engines**

D

Copy code

```
import std.stdio;

void main()
{
    writeln("Hello, World!");
}
```

This program uses the `import` statement to import the `std.stdio` module, which provides functions for input and output. The `void main()` function is the entry point of the program, and it simply calls the `writeln` function to print the string "Hello, World!" to the console.

Note that D is a modern systems programming language that is designed for performance, safety, and productivity. It features a concise syntax, automatic memory management, and built-in support for concurrency and parallelism.

LabVIEW

Popularity: **49**

Year Created: **1986**

Type: **VPL, HPL**

Uses: **Engineering & Science**

PowerShell

Popularity: **50**

Year Created: **2006**

Type: **GPL, CL, PP**

Uses: **Windows OS**

powershell Copy code

```powershell
Get-ChildItem
```

powershell Copy code

```powershell
Get-ChildItem -Recurse
```

This command uses the `Get-ChildItem` cmdlet, which retrieves the child items (files and directories) in the current directory. When you run this command in PowerShell, it will list the names of all the files in the current directory.

You can also use various parameters with the `Get-ChildItem` cmdlet to customize the output. For example, you can use the `-Recurse` parameter to list all files in the current directory and all subdirectories:

This will recursively list all files in the current directory and its subdirectories. You can also use other parameters, such as `-Filter` to specify a file name filter, and `-Exclude` to exclude specific files or directories.

ABC

Popularity: **51**

Year Created: **1987**

Type: **FPL, High Level, OOP**

Uses: **Teach Beginners**

arduino

Copy code

```arduino
PRINT "Hello, World!"
```

Action Script

Popularity: **52**

Year Created: **1998**

Type: **FPL, OOP**

Uses: **Adobe Flash Apps, Flash Games**


Uninstall Adobe Flash Player
Adobe
ADOBE®
FLASH® PLAYER
This program will remove Adobe® Flash® Player from your computer.
QUIT
UNINSTALL


BACK
FREE FOR ALL
3
-man survival fest!
MARIO
LUIGI
PEACH
BOWSER
YOSHI
WARIO
WALUIGI
DONKEY KONG
LINK
ZELDA
SHEIK
CAPTAIN FALCON
PIKACHU
JIGGLYPUFF
PICHU
LUCARIO
KIRBY
META KNIGHT
BANDANA DEE
FOX
FALCO
KRYSTAL
SAMUS
ZERO SUIT SAMUS
NESS
MARTH
PIT
ISAAC
CHIBI-ROBO
Mr. GAME & WATCH
SONIC
TAILS
MEGAMAN
RYU
PAC-MAN
LLOYD
SIMON
BOMBERMAN
BLACK MAGE
GANONDORF
SORA
RAYMAN
SANDBAG
GOKU
LUFFY
NARUTO
ICHIGO
JOTARO
X
ZERO
CARTMAN
RANDOM
SILVER
LILAC
YUSUKE
JOSUKE
MAKOTO YUKI
RENA RYUUGU
YOKO
MIU
TSUMUGI
MAHIRU
MEWTWO
CELESTIA
CLOUD
HMN
P1
CPU
CPU
CPU
CPU
CPU
CPU
P1
CPU LV.
9
CPU LV.
9
CPU LV.
9

javascript Copy code

```javascript
import flash.display.Sprite;
import flash.events.Event;

var ball:Sprite = new Sprite();
ball.graphics.beginFill(0xFF0000);
ball.graphics.drawCircle(0, 0, 50);
ball.graphics.endFill();
addChild(ball);

var xSpeed:Number = 10;
var ySpeed:Number = 10;
var gravity:Number = 1.2;
var friction:Number = 0.9;

ball.addEventListener(Event.ENTER_FRAME, moveBall);

function moveBall(event:Event):void {
    ball.x += xSpeed;
    ball.y += ySpeed;
    ySpeed += gravity;

    if (ball.x + ball.width/2 > stage.stageWidth) {
        ball.x = stage.stageWidth - ball.width/2;
        xSpeed *= -friction;
    }
    else if (ball.x - ball.width/2 < 0) {
        ball.x = ball.width/2;
        xSpeed *= -friction;
    }

    if (ball.y + ball.height/2 > stage.stageHeight) {
        ball.y = stage.stageHeight - ball.height/2;
        ySpeed *= -friction;
    }
    else if (ball.y - ball.height/2 < 0) {
        ball.y = ball.height/2;
        ySpeed *= -friction;
    }
}
```

This code creates a red ball object using the Sprite class and sets its xSpeed, ySpeed, gravity, and friction properties. It then adds an event listener to the ball object, listening for the ENTER_FRAME event, and calls the moveBall function every time this event occurs.

The moveBall function updates the ball's position based on its xSpeed and ySpeed properties, adds gravity to the ySpeed, and checks if the ball has hit any of the stage edges. If the ball has hit the right or left edge, it bounces off and its xSpeed is reversed using the friction property. If the ball has hit the top or bottom edge, it bounces off and its ySpeed is reversed using the friction property.

Overall, this code creates a simple bouncing ball animation on the canvas of Adobe Animate.

Apex

Popularity: **53**

Year Created: **2007**

Type: **OOP**

Uses: **SalesForce.com**

typescript

Copy code

```
public class HelloWorld {
    public static void main(String[] args) {
        System.debug('Hello World!');
    }
}
```

This program defines a class called `HelloWorld` with a single method called `main`. When the program is executed, the `main` method will be called, which will print the string "Hello World!" to the debug log using the `System.debug` method.

Note that Apex is a language used specifically for developing applications on the Salesforce platform, so this program assumes that you have access to a Salesforce org to run it in.

Avenue

Popularity: **54 (Not Used)**

Year Created: **1995**

Type: **OOP**

Uses: **Arcview GIS**

arduino

Copy code

```
av.msgbox("Hello, World!")
```

This code will display a message box with the text "Hello, World!". Note that this code can only be run within the ArcView software, as Avenue is not a standalone language.

BCPL

Popularity: **55**

Year Created: **1967**

Type: **GPL, PP**

Uses: **Compilers & OS (70s-80s)**

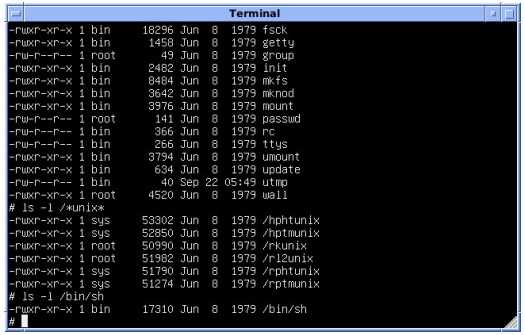
scss

Copy code

```
GET "LIBHDR"          (* Include standard library *)

LET STDOUT = #177777; (* Define output stream *)
WRITEF(STDOUT, "Hello, World!\n"); (* Write message *)
```

This program includes a standard library header file and defines an output stream to print the "Hello, World!" message to the console using the `WRITEF` function. The program should output the message "Hello, World!" followed by a newline character.



Bourne Shell

Popularity: **56**

Year Created: **1977**

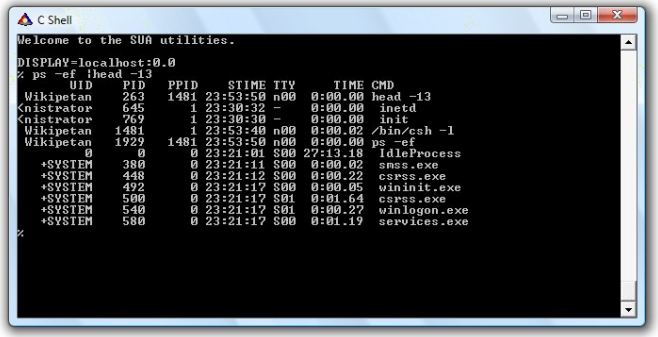
Type: **GPL, CL, PP**

Uses: **Unix Shell, OS**

bash Copy code

```bash
ls -l
```

This command lists the contents of the current directory in long format. The `ls` command is used to list the files and directories in a directory, while the `-l` option specifies the long format.



C Shell

Popularity: **57**

Year Created: **Late 1970s**

Type: **GPL, CL, PP**

Uses: **Unix Shell, OS, Shell Scripting**

bash Copy code

```bash
ls
```

Here's an example of a command in C shell (csh) that lists all files in the current directory:

CL (OS/400) (Control Language)

Popularity: **58**

Year Created: **1978**

Type: **GPL, CL, High Level, PP**

Uses: **Scripts, IBM, AS/400 OS**

scss

Copy code

```scss
CRTLIB LIB(MYLIB) TEXT('My library')
```

This command creates a new library called "MYLIB" with the description "My library".

Clips

Popularity: **59**

Year Created: **1985**

Type: **DSL**

Uses: **Research, Medical, Financial**

scss Copy code

```
(defrule greet
  (say-hello)
  =>
  (printout t "Hello, world!" crlf)
)
```

This program defines a single rule called "greet" that is triggered when a fact with the "say-hello" template is asserted. When the rule is triggered, it prints the message "Hello, world!" to the console.

The program then asserts a single fact with the "say-hello" template to trigger the rule, and calls the "run" function to start the CLIPS inference engine and execute the rule.

Clojure

Popularity: **60**

Year Created: **2007**

Type: **FPL, (Lisp Family)**

Uses: **Web Applications, Big Data, Machine Learning**

arduino

Copy code

```
(defn greet []
  (println "What is your name?")
  (let [name (read-line)]
    (println "Hello, " name "!")))
```

When this program is run, it will display the message "What is your name?" and wait for the user to enter their name. Once the user has entered their name, the program will print the message "Hello, [name]!" where [name] is the name that the user entered.

To run this program, you can save it to a file with a .clj extension (e.g., hello.clj) and then run it using the Clojure command-line interface. For example, if you saved the program to a file called hello.clj, you could run it using the following command:

Copy code

```
clojure hello.clj
```

This would execute the program and display the message "What is your name?" on the command line. You can then enter your name and press Enter to see the greeting.

CLU

Popularity: **61**

Year Created: **1974**

Type: **GPL, PP**

Uses: **High Level of Abstraction, OS, Compilers, Simulation Programs**

arduino

Copy code

```arduino
begin
  writeln("Hello, World!");
end
```

This program uses the `writeln` function to output the string "Hello, World!" to the console. The `begin` and `end` statements define a block of code to be executed.

CoffeeScript

CoffeeScript

Popularity: 62

Year Created: 2009

Type: FPL (JavaScript Family), OOP

Uses: (Syntactic Sugar)

c

Copy code

```c
console.log "Hello, world!"
```

Common Lisp

Popularity: **63**

Year Created: **1984**

Type: **FPL, OOP**

Uses: **AI, Computer Graphics, Numerical Analysis**

scss Copy code

```
(defun hello-world ()
  (format t "Hello, world!~%"))

(hello-world)
```

This program defines a function called `hello-world` that simply prints the message "Hello, world!" to the console using the `format` function. The `t` argument to `format` specifies that the output should go to the console, and the `~%` sequence is a newline character. Finally, the program calls the `hello-world` function to actually run it.

Crystal

Popularity: **64**

Year Created: **2014**

Type: **FPL, High Level, OOP**

Uses: **(Similar To Ruby) - better**

crystal

Copy code

```crystal
puts "Hello, world!"
```

cT

Popularity: **65**

Year Created: **1998**

Type: **FPL, 3D Animations & Simulations, PP**

Uses: **Engineering & Education**

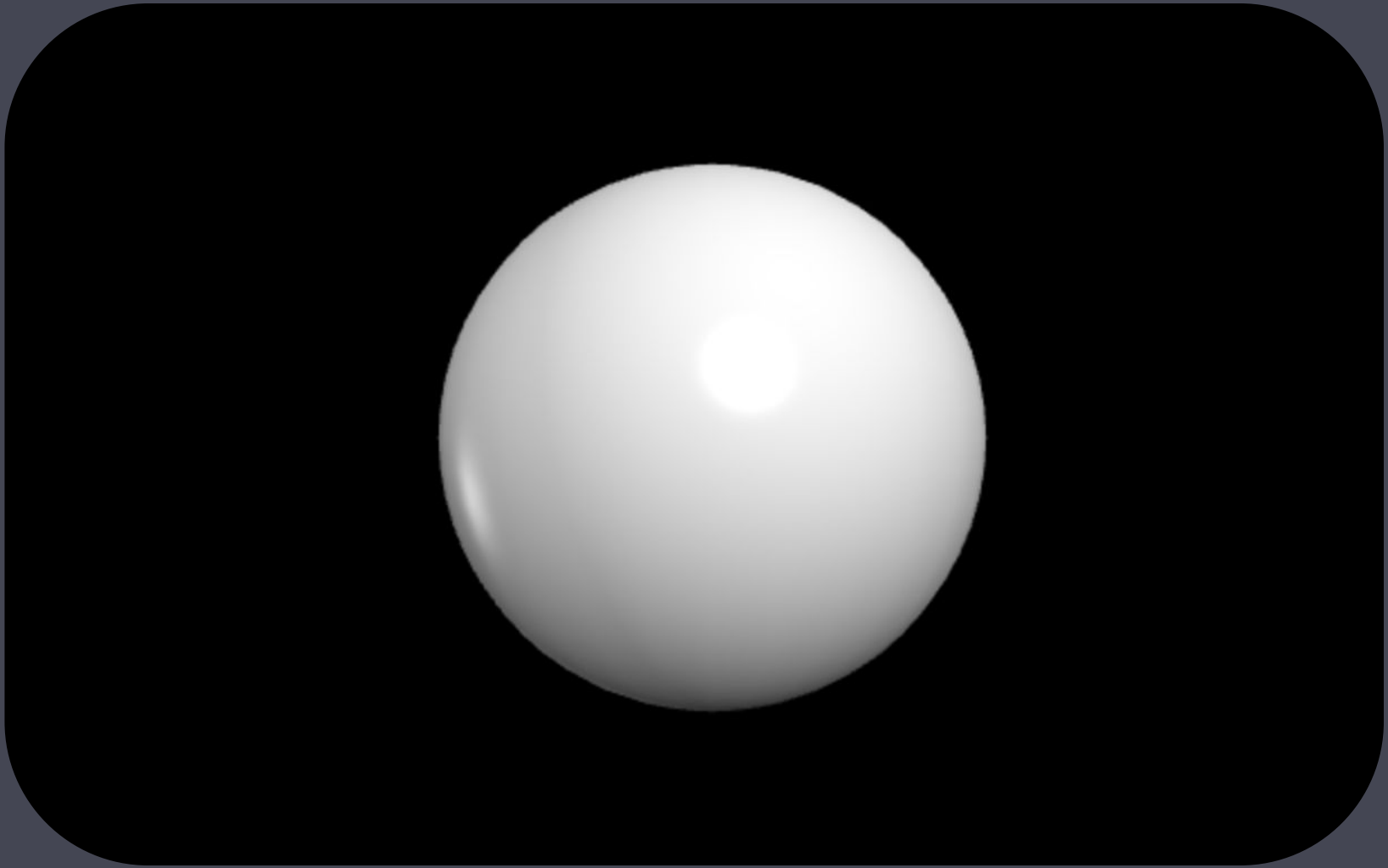
arduino

Copy code

```
from vpython import *

sphere(pos=vector(0, 0, 0), radius=0.5)
```

This program imports the VPython library and uses the `sphere` function to create a 3D sphere at the position `(0, 0, 0)` with a radius of `0.5`. The `vector` function is used to specify the position as a 3D vector.



```
Web VPython 3.2

# Set up the canvas and center point
scene = canvas(title='Atom Simulation', width=800, height=600)
center = vector(0, 0, 0)

# Create the nucleus with protons and neutrons
nucleus = compound([sphere(pos=vector(-0.5, 0, 0), radius=0.4, color=color.red),
                    sphere(pos=vector(0.5, 0, 0), radius=0.4, color=color.red),
                    sphere(pos=vector(0, 0.5, 0), radius=0.4, color=color.blue),
                    sphere(pos=vector(0, -0.5, 0), radius=0.4, color=color.blue)])

# Set up the initial position and velocity of the electrons
electron1 = sphere(pos=vector(1, 0, 0), radius=0.1, color=color.green, make_trail=True, trail_color=color.yellow)
electron1.velocity = vector(0, 0.2, 0)

electron2 = sphere(pos=vector(-1, 0, 0), radius=0.1, color=color.green, make_trail=True, trail_color=color.yellow)
electron2.velocity = vector(0, -0.2, 0)

# Define the circular path of the electrons
r = mag(electron1.pos - center)
omega = sqrt(1/r**3)
theta1 = 0
theta2 = pi

# Run the simulation
dt = 0.01
while True:
    rate(100)

    # Update the position of the electrons
    electron1.pos = vector(r * cos(theta1), r * sin(theta1), 0)
    electron2.pos = vector(r * cos(theta2), r * sin(theta2), 0)

    theta1 += omega * dt
    theta2 += omega * dt

    # Update the velocity of the electrons
    electron1.velocity = vector(-r * omega * sin(theta1), r * omega * cos(theta1), 0)
    electron2.velocity = vector(-r * omega * sin(theta2), r * omega * cos(theta2), 0)

    # Move the electrons to the other side of the nucleus if they pass through it
    if mag(electron1.pos - center) < nucleus.radius:
        electron1.pos = -electron1.pos
    if mag(electron2.pos - center) < nucleus.radius:
        electron2.pos = -electron2.pos
```

CMU CS ACADEMY

Graphics Reference Sheet

SHAPE PARAMETERS

		fill	border	borderWidth	opacity	rotateAngle	dashes	align	visible	roundness	size	font	bold	italic	lineWidth	arrowStart	arrowEnd
	default	'black'	None	2	100	0	False	'center'	True	None	12	arial	False	False	2	False	False
	Rect(left, top, width, height)	✓	✓	✓	✓	✓	✓	'left-top'	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Oval(centerX, centerY, width, height)	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Circle(centerX, centerY, radius)	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
	RegularPolygon(centerX, centerY, radius, points)	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Polygon(x1, y1, x2, y2, x3, y3, ...)	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
	Arc(centerX, centerY, width, height, startAngle, sweepAngle)	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
★	**Star**(centerX, centerY, radius, points)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
	Line(x1, y1, x2, y2)	✓	✗	✗	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓
text	**Label**(value, centerX, centerY)	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✗	✗	✗

Position keywords: 'center', 'left', 'right', 'top', 'bottom', 'left-top', 'right-top', 'left-bottom', 'right-bottom'

✓ Shape has this property

✗ Shape does not have this property

SHAPE METHODS

.toBack()
.toFront()
.hits(x, y)
.contains(x, y)
.hitsShape(shape)
.containsShape(shape)
.addPoint()

GROUP METHODS

.clear()
.add(shape)
.remove(shape)
.hitTest(x, y)

APP METHODS + PROPERTIES

.stop()
.group.hitTest(x, y)
.getTextInput()

.stepsPerSecond = 30
.background = None
.group = Group()
.paused = False

COMMON COLORS

red	rgb(255, 0, 0)
darkRed	rgb(139, 0, 0)
orange	rgb(255, 165, 0)
darkOrange	rgb(255, 140, 0)
orangeRed	rgb(255, 59, 0)
khaki	rgb(240, 230, 140)
yellow	rgb(255, 255, 0)
gold	rgb(255, 215, 0)
limeGreen	rgb(124, 252, 0)
lime	rgb(0, 255, 0)
green	rgb(0, 128, 0)
darkGreen	rgb(0, 100, 0)
lightCyan	rgb(224, 255, 255)
aquaMarine	rgb(127, 255, 212)
skyBlue	rgb(135, 206, 235)
aqua	rgb(0, 255, 255)
turquoise	rgb(64, 224, 208)
blue	rgb(0, 0, 255)
navy	rgb(0, 0, 128)
violet	rgb(238, 130, 238)
orchid	rgb(218, 112, 214)
magenta	rgb(255, 0, 255)
blueViolet	rgb(138, 43, 226)
darkViolet	rgb(148, 0, 211)
purple	rgb(128, 0, 128)
indigo	rgb(75, 0, 130)
pink	rgb(255, 192, 203)
hotPink	rgb(255, 105, 180)
deepPink	rgb(255, 20, 147)
burlywood	rgb(222, 184, 135)
sandyBrown	rgb(244, 164, 96)
goldenRod	rgb(218, 165, 32)
peru	rgb(205, 133, 63)
chocolate	rgb(210, 105, 30)
sienna	rgb(160, 82, 45)
maroon	rgb(128, 0, 0)
white	rgb(255, 255, 255)
gainsboro	rgb(220, 220, 220)
darkGrey	rgb(169, 169, 169)
gray	rgb(128, 128, 128)
slateGray	rgb(112, 128, 144)
dimGray	rgb(105, 105, 105)
black	rgb(0, 0, 0)

EVENT FUNCTIONS

onMousePress(mouseX, mouseY)
onMouseRelease(mouseX, mouseY)
onMouseMove(mouseX, mouseY)
onMouseDrag(mouseX, mouseY)
onKeyPress(key)
onKeyHold(keys)
onKeyRelease(key)
onStep()

MATH FUNCTIONS

distance(x1, y1, x2, y2)
angleTo(x1, y1, x2, y2)
getPointInDir(centerX, centerY, angle, distance)
makeList(row, col)

MEDIA OBJECTS

Image(url, left, top)
track = Sound(url)
track.play(restart=True, loop=True)
track.pause()

Elixir

Popularity: **66**

Year Created: **2011**

Type: **FPL, High Level,**

Uses: **Scalable, Fault-tolerant, and Distributed Systems**

arduino

Copy code

```
IO.puts("Hello, world!")
```

ERLANG

Erlang

Popularity: **67**

Year Created: **Late 1980s**

Type: **FPL, High Level**

Uses: **Scalable, Fault-tolerant, and Distributed Systems**

scss

Copy code

```
-module(my_module).
-export([sum/2]).

sum(A, B) -> A + B.
```

To run this program, save it to a file called `my_module.erl`, and then compile it by running `erlc my_module.erl`. This will generate a file called `my_module.beam`.

Once the file has been compiled, you can start an Erlang shell by running `erl`. From the shell, you can load the module by running `my_module:sum(2, 3).` and it should output `5`.

Forth

Popularity: **68**

Year Created: **1968**

Type: **GPL, Stack-based, PP**

Uses: **Embedded Systems, Robotics, Low-Level**

arduino

Copy code

```arduino
." Hello, World!"
```

GAMS

Popularity: **69**

Year Created: **1984**

Type: **DSL, High Level**

Uses: **MATH, Engineering, Science, Economics**

scss

Copy code

```
set i /1,2,3/;
parameters c(i) /1 2, 2 3, 3 1/;
variables x(i);
equation obj;
obj.. sum(i, c(i)*x(i)) =E= 0;
model lp /obj/;
solve lp using lp minimizing obj;
display x.l;
```

This program defines a set `i` with three elements, a parameter `c` which is a two-dimensional parameter with values for each combination of `i` values, and three variables `x`. The program then defines an equation `obj` that sets the objective function as the sum of `c(i)*x(i)` over all values of `i`. The program then defines a model `lp` with the objective function, and solves the model using the linear programming solver (`using lp`) and minimizing the objective function. Finally, the program displays the values of the `x` variables.

This particular program solves a simple linear optimization problem with three variables and three constraints, but GAMS can be used to solve more complex optimization problems with a wide range of objectives and constraints.

Hack

Popularity: **70**

Year Created: **2014**

Type: **GPL, Dialect of PHP, PP**

Uses: **Facebook, Web Applications**

php

Copy code

```php
<?hh
  echo "Hello, World!";
?>
```

Inform

Popularity: **71**

Year Created: **1993**

Type: **GPL, IF, PP**

Uses: **Z-code, Glulx (creating interactive fiction games)**

vbnet

Copy code

```
"Hello, World!" by Your Name

The story headline is "A Simple Example".

When play begins:
    say "Hello, World!".
```

This program defines a story with a headline of "A Simple Example". When the game starts, the `play begins` rule is triggered and the message "Hello, World!" is printed to the screen.

Io

Popularity: **72**
Year Created: **2002**
Type: **FPL, PP, OOP**
Uses: **Web Apps, Smalltalk, Self, Lua, Lisp, Act1, and NewtonScript**

```
for(i, 1, 10, write(i * i, " "))
```

Ioke

Ioke

Popularity: **73**

Year Created: **2008**

Type: **FPL, PP, OOP**

Uses: **DSL, JVM, Scripting**

arduino

Copy code

```arduino
"Hello, world!" println
```

J

Popularity: **74**

Year Created: **1990**

Type: **FPL, PP, OOP, High-Level**

Uses: **Scientific, Financial, Machine Learning, AI**

```
1+i.10
```

Explanation:

- `i.10` generates the sequence of integers from 0 to 9.
- `1+` adds 1 to each element of the sequence.
- The result is automatically printed to the console.

J#

Popularity: **75**

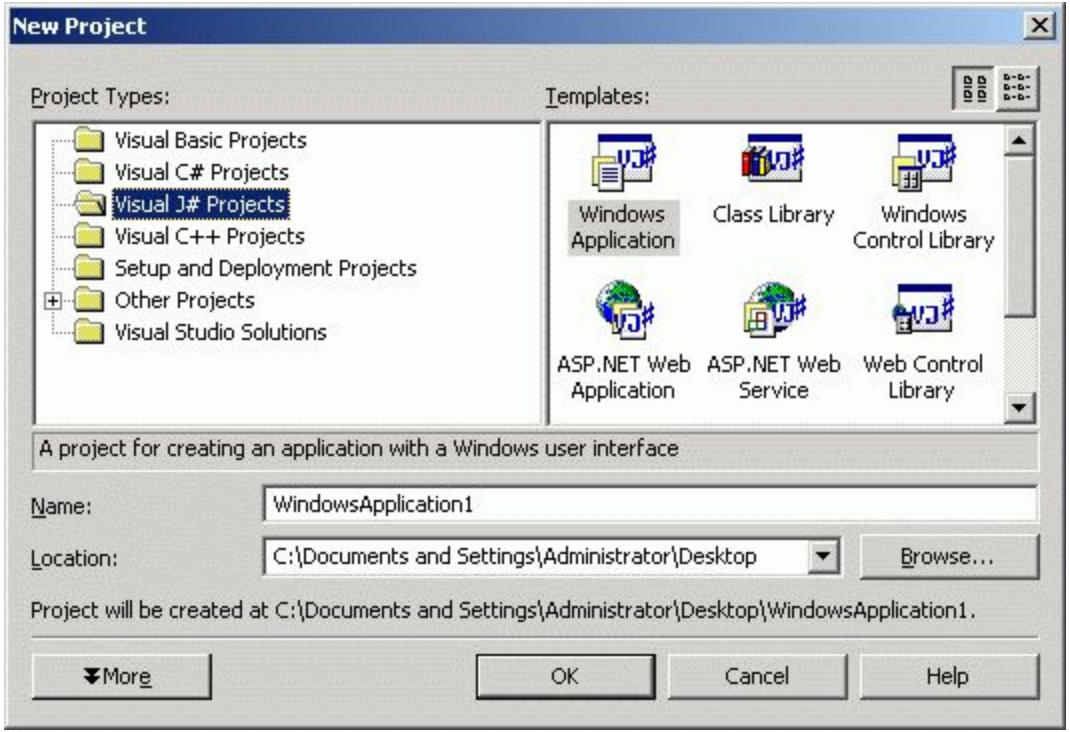
Year Created: **2002 (DISCONTINUED**

Type: **GPL, OOP**

Uses: **Microsoft variation of the Java language that runs on the .NET Framework**

```
    // The main entry point for the application.
    /** @attribute System.STAThreadAttribute() */
    public static void main(String[] args)
    {
        Application.Run(new Form1());
    }

    private void button1_Click (System.Object sender, System.EventArgs e)
    {
        System.Windows.Forms.MessageBox.Show("Hello, World!");
    }
}
```



https://www.codeproject.com/Articles/1440/Introduction-to-Visual-J-NET

JScript

Popularity: **76**

Year Created: **1996**

Type: **FPL, High Level, OOP**

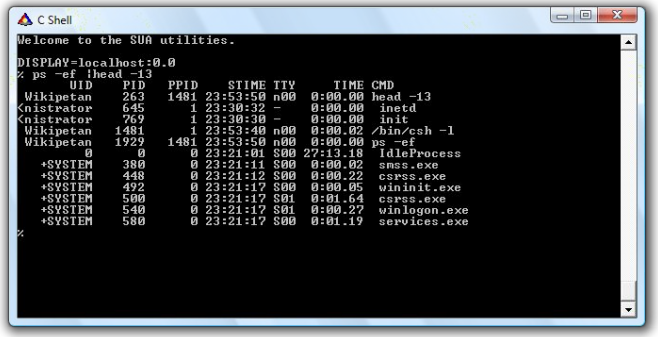
Uses: **Microsoft, Web Applications, Internet Explorer**

csharp

Copy code

```
// Prompt the user for their name
var name = prompt("What is your name?");

// Display a personalized greeting message
alert("Hello, " + name + "! Welcome to JScript!");
```



Korn Shell

Popularity: **77**

Year Created: **1993**

Type: **GPL, CL, PP**

Uses: **Unix Scripting, Unix OS**

ksh

Copy code

```ksh
#!/bin/ksh
# This program greets the user with their name

echo "What is your name?"
read name

echo "Hello, $name!"
```

This program prompts the user for their name, reads the input, and then greets the user with their name using the `echo` command. The `#!/bin/ksh` line at the beginning of the program specifies that the script should be run using the Korn Shell interpreter.

Ladder Logic

Popularity: **78**

Year Created: **Late 1960s**

Type: **VPL, HPL**

Uses: **PLCs, Relay Stacks, Machines, Robotics**


How It's
Made

Name	Type	Value
Cycle	Timer	
Z0	Bool	False
Z1	Bool	False
Z2	Bool	False
X0	Bool	False
TimeOnZ0	Time	3000
TimeOnZ1	Time	5000
TimeOnZ2	Time	7000
SystemOn	Bool	False
2ClkTmr	Timer	
Counter	Counter	
2Clicks	Bool	False
0	Number	0

Add new variable Timer Submit

Name	Type	Value
Cycle	Timer	
Z0	Bool	False
Z1	Bool	False
Z2	Bool	False
X0	Bool	False
TimeOnZ0	Time	3000
TimeOnZ1	Time	5000
TimeOnZ2	Time	7000
SystemOn	Bool	False
2ClkTmr	Timer	
Counter	Counter	
2Clicks	Bool	False
0	Number	0

Add new variable Timer Submit

Live Code

Popularity: **79**

Year Created: **1993**

Type: **GPL, High Level, PP**

Uses: **Apps, Easy**

vbnet

Copy code

```vbnet
on mouseUp
  put "Hello, World!" into field "output"
end mouseUp
```

This program creates a button in a LiveCode interface. When the button is clicked, it sets the text of a field named "output" to "Hello, World!".

ML

Popularity: **80**

Year Created: **Late 1970s**

Type: **FPL**

Uses: **Machine Learning, Research, Compilers**

kotlin

Copy code

```kotlin
fun main () = print "Hello, World!\n";
```

MQL5

Popularity: **81**

Year Created: **2010**

Type: **GPL, High Level, OOP**

Uses: **Developing trading robots, Technical market indicators, Scripts**

csharp

Copy code

```csharp
// MQL5 program to print "Hello, world!" to the terminal

void OnStart()
{
   Print("Hello, world!");
}
```

NATURAL

Popularity: **82**

Year Created: **1970**

Type: **GPL, PP**

Uses: **Business & Databases**

sql

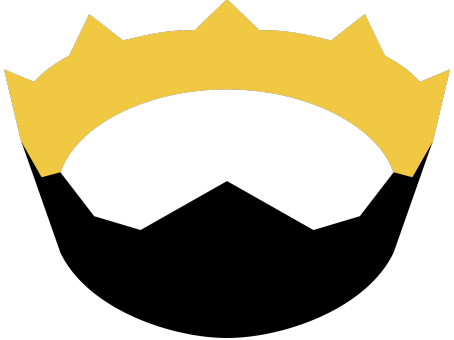
Copy code

```sql
DEFINE DATA LOCAL
1 #NAME (A20)
END-DEFINE

WRITE 'What is your name?'
INPUT #NAME

WRITE 'Hello ' #NAME ', nice to meet you!'
```

This program first declares a local variable `#NAME` of type `A20`, which can hold a string of up to 20 characters. It then uses the `WRITE` statement to prompt the user for their name and read their input into `#NAME` using the `INPUT` statement. Finally, it displays a personalized message using the `WRITE` statement and concatenating the value of `#NAME` with the rest of the message.



NIM

Popularity: **83**

Year Created: **2008**

Type: **FPL, High-Level, PP**

Uses: **Web Dev, Game Dev, Science**

nim

Copy code

```nim
echo "Hello, world!"
```

OCaml

Popularity: **84**

Year Created: **1996**

Type: **FPL, High-Level**

Uses: **Compilers, OS, Numbers, Tools**

This program uses the `print_string` function to output the message "Hello, World!" to the console, followed by a newline character (`\n`) to move the cursor to the next line.

ocaml Copy code

```ocaml
print_string "Hello, World!\n";;
```

occam

Popularity: **85**

Year Created: **1983**

Type: **HPL, PP, High-Level**

Uses: **Compilers, OS, Drivers, Network Apps**

lua

Copy code

```lua
-- OCCAM program that prints "Hello, world!" to the console

VAL
  msg IS STRING "Hello, world!"
:
PROC main()
  SEQ
    -- Print the message to the console
    stdout(msg)
  :
:
```

In this program, we declare a string variable `msg` and initialize it with the value "Hello, world!". Then, we define the `main` process, which uses the `stdout` command to print the message to the console. Finally, we terminate the program using the `:` symbol.

Note that OCCAM is a language that emphasizes concurrency, so programs written in OCCAM usually involve multiple processes running concurrently. However, this simple program uses only a single process to print the message to the console.

PILOT (Programmed Inquiry Learning Or Teaching)

Popularity: **86**

Year Created: **1966**

Type: **GPL, PPL**

Uses: **Education, Text-based Games**

markdown

Copy code

```
1. PROCEDURE MAIN
2.   PRINT "What is your name?"
3.   INPUT NAME
4.   PRINT "Hello, " + NAME + "!"
5. ENDPROC
```

PL/I (Programming Language One)

Popularity: **87**

Year Created: **1966**

Type: **GPL, PP**

Uses: **Scientific, Engineering, Business, Governments**

sql Copy code

```sql
/* PL/I program to print "Hello, world!" */

PUT SKIP LIST('Hello, world!');
END
```

This program uses the `PUT` statement to print the string "Hello, world!" to the console, followed by a newline character (`SKIP`). The `LIST` keyword specifies that the output should be sent to the console.

The `END` statement signals the end of the program.

PWCT (Programming Without Coding Technology)

Popularity: **88**

Year Created: **2008**

Type: **VPL, DSL**

Uses: **C++, Java, Python, Ada**

lua

Copy code

```lua
function main()
    num1 = input("Enter first number: ")
    num2 = input("Enter second number: ")
    sum = num1 + num2
    output("The sum is: " + sum)
end function
```

Q

Popularity: **89**

Year Created: **2003**

Type: **High Level, FPL**

Uses: **Finance, Healthcare, Telecommunications**

```
sum: {[x;y] x+y}

sum[2;3]
```

This program defines a function `sum` that takes two arguments `x` and `y` and returns their sum. The function is defined using Q's square-bracket syntax, which is a shorthand for defining a lambda function. The first line defines the function, and the second line calls it with the arguments 2 and 3. When you run this program, it should print out the result `5`.

Racket

Popularity: **90**

Year Created: **1995**

Type: **FPL, GPL (Lisp Family), OOP**

Uses: **Research, Apps, Games**

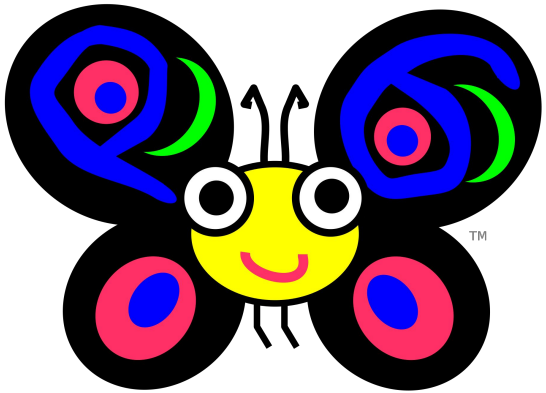
arduino

Copy code

```
#lang racket

(displayln "Hello, World!")
```

This program uses the `displayln` function to output the string "Hello, World!" to the console. The `#lang racket` line at the top of the file tells Racket which language to use to interpret the program.



Raku (Perl 6)

Popularity: **91**

Year Created: **2000**

Type: **FPL, MPL, OOP**

Uses: **Web Development, Science, Database**

javascript Copy code

```
#!/usr/bin/env raku

say "Hello, World!";
```

The first line of the program (`#!/usr/bin/env raku`) is called a shebang line and tells the operating system to use the Raku interpreter to execute the program. The `say` keyword is used to print a string to the console, and the string `"Hello, World!"` is enclosed in double quotes to indicate that it should be treated as a string literal.

λ

Scheme

Popularity: **92**

Year Created: **1975**

Type: **FPL (Lisp Family)**

Uses: **Research, Education, Science**

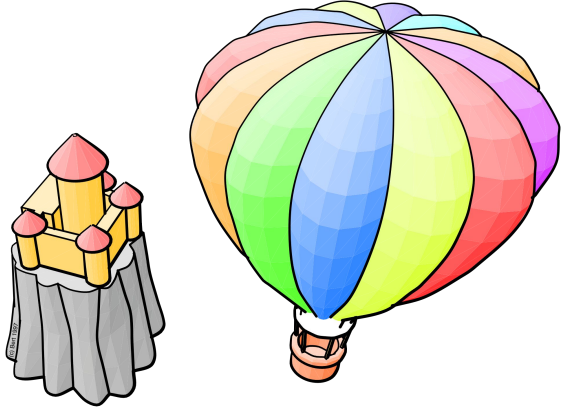
scheme Copy code

```scheme
(define (factorial n)
  (if (= n 0)
      1
      (* n (factorial (- n 1)))))
```

- The `define` statement defines a function named `factorial` that takes one argument, `n`.
- The body of the function is an `if` statement. If `n` is equal to 0, the function returns

scheme Copy code

```scheme
(factorial 5)
```



Smalltalk

Popularity: **93**

Year Created: **1972**

Type: **OOP**

Uses: **Finance, Insurance, & Education, Apps**

go

Copy code

```
| name |
name := UIManager default request: 'Enter your name:'.
Transcript show: 'Hello, ', name, '!', cr.
```

The first line declares a variable `name`. The `:=` operator assigns the value returned by the `request:` message to `name`. The `request:` message displays a dialog box with the given prompt and returns the string entered by the user.

The second line prints a greeting to the Smalltalk `Transcript`, which is similar to a console output. The `show:` message concatenates the string `'Hello, '`, the value of the `name` variable, and the string `'!'`, and prints the result to the transcript. The `cr` message adds a newline character.

To run this program, you can copy and paste it into a Smalltalk environment such as GNU Smalltalk or Squeak, or save it to a file with a `.st` extension and load it into the environment.

Solidity

Popularity: **94**

Year Created: **2014**

Type: **High Level, OOP, DSL**

Uses: **Smart Contracts - Ethereum**

csharp

Copy code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleContract {
    string public message;

    function setMessage(string memory newMessage) public {
        message = newMessage;
    }
}
```

This contract is called `SimpleContract` and has one public string variable called `message`. The `setMessage` function allows anyone to set the value of `message`.

Note that the first line `// SPDX-License-Identifier: MIT` is a special comment that indicates the license under which this code is released. The next line `pragma solidity ^0.8.0;` specifies the version of the Solidity compiler to use.

SPARK

Popularity: **95**

Year Created: **Mid-1990s**

Type: **Ada, PP, DSL**

Uses: **Aerospace, Defense, Transportation**

vbnet

Copy code

```
pragma SPARK_Mode (On);

function Add (X, Y : Integer) return Integer is
begin
  pragma Assert (X + Y <= Integer'Last);
  return X + Y;
end Add;
```

This program defines a function `Add` that takes two integer arguments `X` and `Y` and returns their sum. The `pragma SPARK_Mode (On)` statement enables formal verification using the SPARK toolset.

The `pragma Assert` statement ensures that the sum of `X` and `Y` does not exceed the maximum representable integer value, preventing integer overflow. If the assertion fails during verification, the program is considered unsafe and the verification fails.

Note that SPARK is a formally verified programming language that emphasizes safety and security. As such, writing even a simple program requires careful consideration of potential safety hazards and the use of formal verification tools to ensure correctness.

Tcl

Popularity: **96**

Year Created: **1988**

Type: **GPL, High Level, OOP, PP**

Uses: **Web Development, Desktop Apps**

c

Copy code

```c
puts "Hello, World!"
```

VHDL

Popularity: **97**

Year Created: **1980s**

Type: **HDL**

Uses: **ASICS, FPGAS, Digital Devices**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity OR_gate is
    port (a, b : in std_logic;
          c : out std_logic);
end OR_gate;

architecture Behavioral of OR_gate is
begin
    c <= a or b;
end Behavioral;
```

This code defines an entity named `OR_gate` that has two inputs, `a` and `b`, and one output, `c`. The architecture named `Behavioral` uses the logical OR operator to calculate the output `c` based on the inputs `a` and `b`. When either `a` or `b` is `1` (i.e., true), `c` will also be `1`.

X++

Popularity: **98**

Year Created: **1998**

Type: **DSL, OOP**

Uses: **Business, Microsoft Dynamics**

scss

Copy code

```
static void HelloWorld(Args _args)
{
    info("Hello World!");
}
```

This program defines a static method called `HelloWorld` that takes an `Args` parameter. Inside the method, the `info` function is called to display the message "Hello World!" in the Infolog.

When this program is executed, the message "Hello World!" will be displayed in the Infolog.

Xojo (REALbasic)

Popularity: **99**

Year Created: **1996**

Type: **GPL, CP, OOP**

Uses: **Apps (Desktop & Web)**

vbnet Copy code

```vbnet
Sub Open()
  // Create a window
  Dim window As New Window
  window.Title = "My Window"
  window.Width = 200
  window.Height = 100

  // Create a button
  Dim button As New PushButton
  button.Caption = "Click Me"
  button.Left = 50
  button.Top = 20

  // Add a click event handler for the button
  AddHandler button.Action, AddressOf ButtonClicked

  // Add the button to the window
  window.AddControl(button)

  // Show the window
  window.Show
End Sub

Sub ButtonClicked()
  // Display a message box when the button is clicked
  MsgBox "Hello, world!"
End Sub
```

This program creates a window and a button, and adds a click event handler to the button. When the button is clicked, a message box is displayed with the text "Hello, world!".

ZIG

Popularity: **100**

Year Created: **2016 - 2018**

Type: **GPL, PP**

Uses: **©, Embedded systems, Networking, systems programming**

c

Copy code

```
const std = @import("std");

pub fn main() !void {
    const stdout = std.io.getStdOut().writer();
    try stdout.writeAll("Hello, world!\n");
}
```

This program imports the `std` library, which provides useful features like standard I/O. It defines a function called `main` that returns nothing (`!void`) and can potentially raise an error (`!`). Inside `main`, it gets a handle to the standard output stream (`stdout`) and writes the string "Hello, world!\n" to it.

When you run this program, it will print "Hello, world!" to the console.