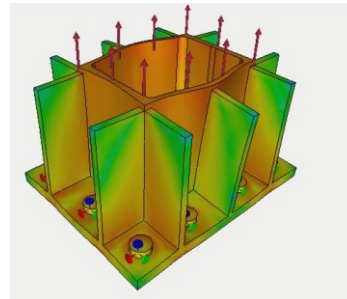





# Fundam Programación





TIOBE

(an official quality measure)

About us

Knowledge

News

Coding Standards

TIOBE Index
























Contact

Products

Quality Models

Markets

Schedule a demo

Jan 2025	Jan 2024	Change	Programming Language		Ratings	Change
1	1			Python	23.28%	+9.32%
2	3			C++	10.29%	+0.33%
3	4			Java	10.15%	+2.28%
4	2			C	8.86%	-2.59%
5	5			C#	4.45%	-2.71%
6	6			JavaScript	4.20%	+1.43%
7	11			Go	2.61%	+1.24%
8	9			SQL	2.41%	+0.95%
9	8			Visual Basic	2.37%	+0.77%
10	12			Fortran	2.04%	+0.94%
11	13			Delphi/Object Pascal	1.79%	+0.70%
12	10			Scratch	1.55%	+0.11%
13	7			PHP	1.38%	-0.41%

# Equipo Docente

Seccion	Dia	Hora	Profesor
Seccion A	Jueves	8 a 10	Claudio Barua
Seccion B	Lunes	730 a 930	Cristian Torres
Seccion c	Viernes	730 a 930	Viviana Ortellado
Seccion D	Miercoles	730 a 930	Osmar Cabrera
Seccion e	Miercoles	930 a 1130	Nestor Barreto
Seccion f	Lunes	730 a 930	Diego Stalder
Seccion G	Martes	730 a 930	Diego Pinto
Seccion h	Sabado	730 a 930	Juan Ovelar
Seccion I	Martes	730 a 930	David Fretes

Seccion	Dia	Horario	Auxiliar
Seccion A	Jueves	10 a 12	Marcos Benitez
Seccion B	Lunes	930 a 1130	Adolfo Monje
Seccion c	Viernes	930 a 1130	Viviana Ortellado
Seccion D	Miercoles	930 a 1130	David Fretes
Seccion e	Miercoles	1130 a 1330	Cristian Colbes
Seccion f	Lunes	930 a 1130	Alexander Martinez
Seccion G	Martes	930 a 1130	Samuel Grau
Seccion h	Sabado	930 a 1130	Osmar Cabrera
Seccion I	Martes	930 a 1130	Sixto Larrosa

Grupo WhatsApp DS



# Libros

## BIBLIOGRAFÍA BÁSICA

1. Guttag, John. Introduction to computation and programming using Python: With application to understanding data. 2da edición. MIT Press, 2016.  
[2]Downey, Allen B. Think Python: How to Think Like a Computer Scientist. O'Reilly Media, 2015.
2. Johansson, Robert. Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib. 2da edición. Apress, 2019.
3. Marzal, Andres; Gracia, Isabel; García, Pedro. Introducción a la programación con Python 3. Universitat Jaume I, 2014.
4. Matthes, Eric (traducción de Pineda, Beatriz). Curso intensivo de Python: Introducción práctica a la programación basada en proyectos. 2da edición. Anaya Multimedia, 2021.

## BIBLIOGRAFÍA COMPLEMENTARIA

1. Matthes, Eric. Python crash course: a hands-on, project-based introduction to programming. 2da edición. No Starch Press, 2019.
2. Gries, Paul; Campbell, Jennifer; Montoyo, Jason. Practical Programming: An Introduction to Computer Science Using Python 3.6. Pragmatic Bookshelf, 2017.
3. Bressert, Eli. SciPy and NumPy: an overview for developers. O'Reilly Media, Inc., 2012



# Contenido

## UNIDAD 1: INTRODUCCIÓN A LA COMPUTACIÓN

- 1.1. Importancia de la Computación. Aplicaciones en la vida moderna.
- 1.2. Conceptos de Computación, Programas y Algoritmos.
- 1.3. Pseudocódigo y Lenguajes de programación. Primer programa en Python.
- 1.4. Operaciones aritméticas, uso de variables.
- 1.5. Ambientes de Programación.

## UNIDAD 2: INTRODUCCIÓN A LA PROGRAMACIÓN EN PYTHON

- 2.1. Datos y Tipos de datos.
- 2.2. Instrucciones, Variables, Constantes, Operadores y Expresiones.
- 2.3. Sintaxis básica y semántica. Errores y depuración.
- 2.4. Cadenas y entrada de datos.
- 2.5. Conversión de tipos de datos.
- 2.6. Modelado y análisis de problemas, diseño de algoritmos y construcción de programas.
- 2.7 Resolución de problemas básicos con instrucciones secuenciales.

## UNIDAD 3: ELEMENTOS DE LA PROGRAMACIÓN ESTRUCTURADA

- 3.1. Estructuras condicionales.
  - 3.1.1. Representación.
  - 3.1.2. Operadores lógicos y relacionales.
  - 3.1.3. Expresiones booleanas.
  - 3.1.4. Estructuras condicionales simples, alternativas y anidadas.

# Contenido

## 3.2. Estructuras repetitivas.

- 3.2.1. Representación. Concepto de bucle e iteración.

- 3.2.2. Repetición simple y condicional.

- 3.2.3. Contadores, acumuladores y banderas.

- 3.2.4. Anidamiento.

- 3.2.5. Interrupciones y saltos.

## 3.3. Subprogramas.

- 3.3.1. Abstracción.

- 3.3.2. Procedimientos y funciones.

- 3.3.3. Definición e invocación en programas.

- 3.3.4. Parámetros, valores y ámbito de variables.

- 3.3.5. Utilización de funciones externas.

- 3.3.6. Manejo básico de módulos en Python.

- 3.3.7. Introducción a la recursividad.

## UNIDAD 4: INTRODUCCION A ESTRUCTURAS DE DATOS

- 4.1. Necesidad de estructuras de datos.

  - Presentación de las estructuras de datos básicas en Python.

- 4.2. Secuencias: Cadenas, tuplas y listas.

- 4.3. Concepto de mutabilidad.

- 4.4. Uso de secuencias en funciones.

- 4.5. Diccionarios.

# Contenido

## UNIDAD 5: ALGORITMOS DE BÚSQUEDA Y ORDENAMIENTO

- 5.1. Búsqueda lineal y binaria.
- 5.2. Algoritmos de Ordenamiento: Selección, Inserción, MergeSort y QuickSort.
- 5.3. Nociones de tiempo de ejecución.

## UNIDAD 6: INTRODUCCIÓN A NUMPY PARA MANEJO DE MATRICES

- 6.1. Arreglos unidimensionales y multidimensionales.
- 6.2. Generación y manipulación de datos.
- 6.3. Operaciones y funciones básicas del módulo.

## UNIDAD 7: MANEJO DE ARCHIVOS Y VISUALIZACIÓN DE DATOS

- 7.1. Lectura y Escritura de archivos de texto.
- 7.2. Manejo de Excepciones.
- 7.3. Uso de matplotlib para visualizar datos.
- 7.4. Histogramas, Líneas, Gráficos de Dispersión e Imágenes.

# Evaluaciones

Taller1	10
1er Parcial	30
Proyecto Desafío	20
2do Parcial	30
VPLs	10



# Algoritmo – Casos de éxito

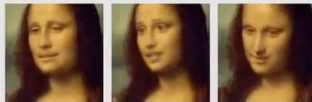


Spotify®



A  
P

NETFLIX



facebook®



Instagram





← → ↻ [github.com/deepseek-ai/DeepSeek-V3/blob/main/inference/generate.py](https://github.com/deepseek-ai/DeepSeek-V3/blob/main/inference/generate.py)

📦 YouTube 📍 Maps 📧 Gmail 🔄 ERA5 hourly data o... 🌤️ Space Weather | M... 📄 Adobe Acrobat

### Files

🔗 main 🔍

🔍 Go to file

- > .github
- > figures
- ▼ inference
  - > configs
    - convert.py
    - fp8\_cast\_bf16.py
    - generate.py**
    - kernel.py
    - model.py
    - requirements.txt
    - ...

### DeepSeek-V3 / inference / generate.py

📦 enochkan Enhance documentation and update .gitignore for model

**Code** Blame 185 lines (166 loc) · 7.41 KB

```
1 import os
2 import json
3 from argparse import ArgumentParser
4 from typing import List
5
6 import torch
7 import torch.distributed as dist
8 from transformers import AutoTokenizer
9 from safetensors.torch import load_model
10
11 from model import Transformer, ModelArgs
12
13
14 def sample(logits, temperature: float = 1.0):
15     """
```

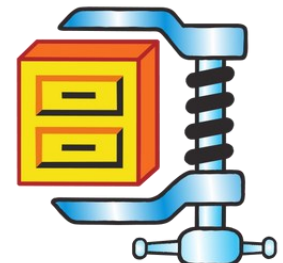
# Sistemas operativos



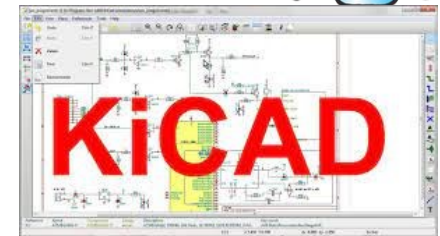
 Windows 10

Linux →  
UNIX

# Software de aplicación



CST® STUDIO SUITE® 2019



# Breve Historia: Programa almacenado

## Calculadoras, Máquinas



<http://users.telenet.be/d.rijmenants/en/enigmasim.htm>

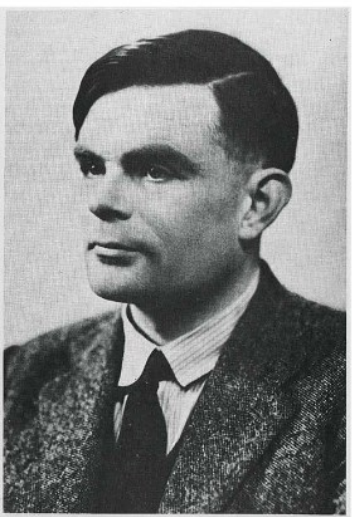


<https://twobithistory.org/2018/08/18/ada-lovelace-note-g.html>

# Breve Historia: Programa almacenado

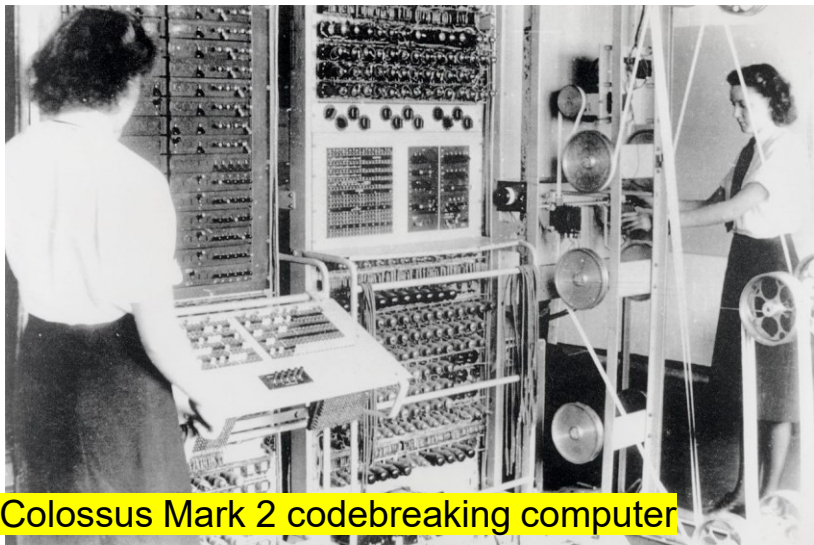


1936, On Computable Numbers, with an Application to the Entscheidungsproblem



Alan Mathison Turing

Máquina programable



Colossus Mark 2 codebreaking computer

Máquinas de computar



estado	vacío	1
A	mover a derecha cambiar a A	mover a derecha cambiar a B
B	escribir 1 mover a derecha cambiar a C	mover a derecha cambiar a B
C	mover a izquierda cambiar a D	mover a derecha cambiar a C
D	quieta cambiar a D	borrar quieta cambiar a D

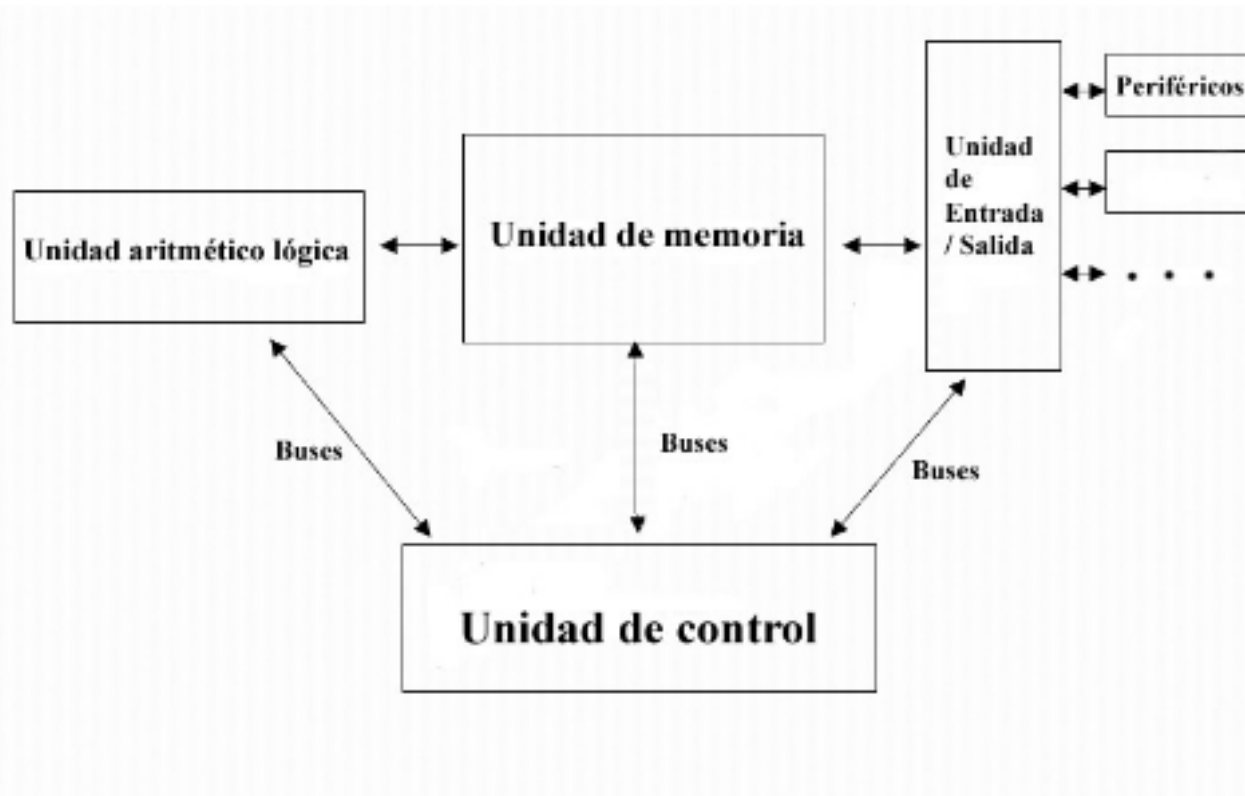
# Arquitectura Von Neumann

Von Neumann conoció a Turing en Princeton en el periodo 1937/38.

A finales de los 40 von Neumann colabora el proyecto ENIAC.

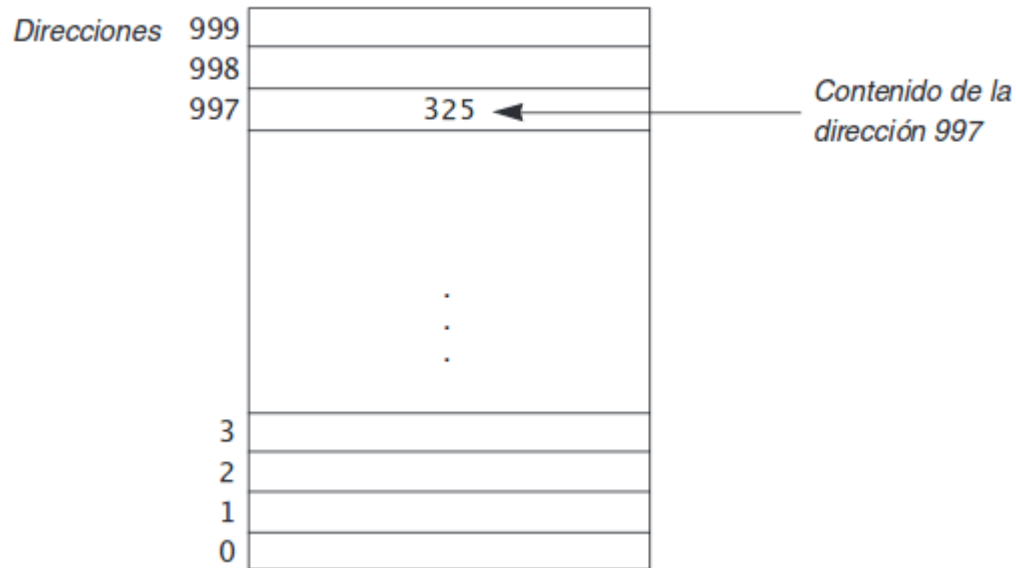


*John Von Neumann*



*Diagrama de la arquitectura de Von Neumann*

# Memoria



Byte	<b>Byte (B)</b>	<i>equivale a</i>	8 bits
Kilobyte	<b>Kbyte (KB)</b>	<i>equivale a</i>	1.024 bytes
Megabyte	<b>Mbyte (MB)</b>	<i>equivale a</i>	1.024 Kbytes
Gigabyte	<b>Gbyte (GB)</b>	<i>equivale a</i>	1.024 Mbytes
Terabyte	<b>Tbyte (TB)</b>	<i>equivale a</i>	1.024 Gbytes

---

**1 Tb = 1.024 Gb = 1.024 × 1.024 Mb = 1.048.576 Kb = 1.073.741.824 B**



# Ejemplos de Tipos de datos

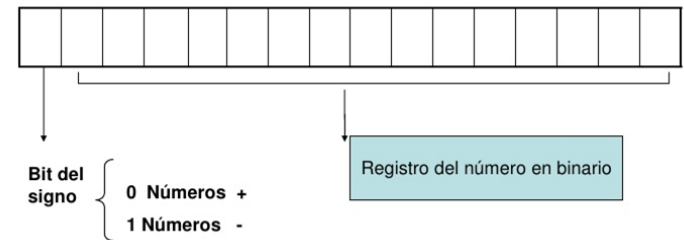
Nombre	Descripción	Ejemplos
Decimal	Entero en base 10	1234
Hexadecimal	Entero en base 16	0x1234
Octal	Entero en base 8	01234
Carácter	Byte en ASCII	'A'
Punto flotante	Número real	1.2 3.4e6 3.456-2
Cadena	Texto literal	"hola Futuros Ingenieros"



# Tipos de datos y su Representación

- Numéricos

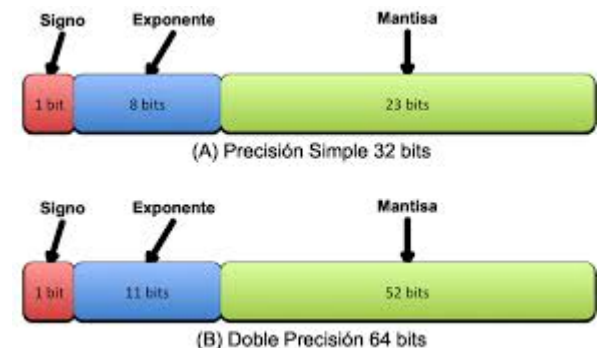
- Reales, binarios: Representación de punto Fijo



- Reales, Representación de punto flotante

- Caracteres (Codificación)  
A, b, c, ..

- Cadenas



# Datos: Codificación

1. **Caracteres alfabéticos** (letras mayúsculas y minúsculas, en una primera versión del abecedario inglés).

A, B, C, D, E, ... X, Y, Z, a, b, c, ... , X, Y, Z

2. **Caracteres numéricos** (dígitos del sistema de numeración).

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 *sistema decimal*

3. **Caracteres especiales** (símbolos ortográficos y matemáticos no incluidos en los grupos anteriores).

{ } Ñ ñ ! ? & > # ¢ ...

4. **Caracteres geométricos y gráficos** (símbolos o módulos con los cuales se pueden representar cuadros, figuras geométricas, iconos, etc.

| □ ▢ ♠ ~ ...

5. **Caracteres de control** (representan órdenes de control como el carácter para pasar a la siguiente línea [NL] o para ir al comienzo de una línea [RC, *retorno de carro*, «*carriage return*, CR»] emitir un pitido en el terminal [BEL], etc.).

**Código ASCII** (American Standard Code for Information Interchange).

El código ASCII básico utiliza 7 bits y permite representar 128 caracteres (letras mayúsculas y minúsculas del alfabeto inglés, símbolos de puntuación, dígitos 0 a 9 y ciertos controles de información tales como retorno de carro, salto de línea, tabulaciones, etc.). Este código es el más utilizado en computadoras, aunque el ASCII ampliado con 8 bits permite llegar a (256) caracteres distintos, entre ellos ya símbolos y caracteres especiales de otros idiomas como el español.

# Codificación ASCII

Caracteres ASCII de control			Caracteres ASCII imprimibles			ASCII extendido (Página de código 437)										
00	NULL	(carácter nulo)	32	espacio	64	@	96	`	128	Ç	160	á	192	Ł	224	Ó
01	SOH	(inicio encabezado)	33	!	65	A	97	a	129	ú	161	í	193	┐	225	Ô
02	STX	(inicio texto)	34	"	66	B	98	b	130	é	162	ó	194	└	226	Ø
03	ETX	(fin de texto)	35	#	67	C	99	c	131	â	163	û	195	┘	227	Ò
04	EOT	(fin transmisión)	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	ö
05	ENQ	(consulta)	37	%	69	E	101	e	133	à	165	Ñ	197	+	229	Õ
06	ACK	(reconocimiento)	38	&	70	F	102	f	134	ã	166	ª	198	ä	230	μ
07	BEL	(timbre)	39	'	71	G	103	g	135	ç	167	º	199	Ä	231	þ
08	BS	(retroceso)	40	(	72	H	104	h	136	ê	168	¿	200	⌋	232	ß
09	HT	(tab horizontal)	41	)	73	I	105	i	137	ë	169	®	201	⌈	233	Ú
10	LF	(nueva línea)	42	*	74	J	106	j	138	è	170	¬	202	⌊	234	Û
11	VT	(tab vertical)	43	+	75	K	107	k	139	ï	171	½	203	⌉	235	Ü
12	FF	(nueva página)	44	,	76	L	108	l	140	ì	172	¼	204	⌈	236	Ý
13	CR	(retorno de carro)	45	-	77	M	109	m	141	í	173	ı	205	=	237	Ÿ
14	SO	(desplaza afuera)	46	.	78	N	110	n	142	Â	174	«	206	≠	238	
15	SI	(desplaza adentro)	47	/	79	O	111	o	143	Ã	175	»	207	□	239	,
16	DLE	(esc.vínculo datos)	48	0	80	P	112	p	144	É	176	☼	208	◊	240	≡
17	DC1	(control disp. 1)	49	1	81	Q	113	q	145	æ	177	☾	209	Ⓓ	241	±
18	DC2	(control disp. 2)	50	2	82	R	114	r	146	Æ	178	■	210	Ê	242	_
19	DC3	(control disp. 3)	51	3	83	S	115	s	147	ô	179		211	È	243	¾
20	DC4	(control disp. 4)	52	4	84	T	116	t	148	õ	180	┘	212	É	244	℥
21	NAK	(conf. negativa)	53	5	85	U	117	u	149	ò	181	Á	213	Ì	245	§
22	SYN	(inactividad sinc)	54	6	86	V	118	v	150	ú	182	Â	214	Í	246	÷
23	ETB	(fin bloque trans)	55	7	87	W	119	w	151	ù	183	À	215	Î	247	˙
24	CAN	(cancelar)	56	8	88	X	120	x	152	ý	184	©	216	Ï	248	˚
25	EM	(fin del medio)	57	9	89	Y	121	y	153	Ö	185	¶	217	¸	249	ˆ
26	SUB	(sustitución)	58	:	90	Z	122	z	154	Ü	186		218	┘	250	˜
27	ESC	(escape)	59	;	91	[	123	{	155	ø	187	¶	219	█	251	¹
28	FS	(sep. archivos)	60	<	92	\	124		156	£	188	¶	220	■	252	²
29	GS	(sep. grupos)	61	=	93	]	125	}	157	∅	189	¢	221	▬	253	³
30	RS	(sep. registros)	62	>	94	^	126	~	158	×	190	¥	222	▬	254	⁴
31	US	(sep. unidades)	63	?	95	_			159	f	191	γ	223	▬	255	nbspace
127	DEL	(suprimir)														



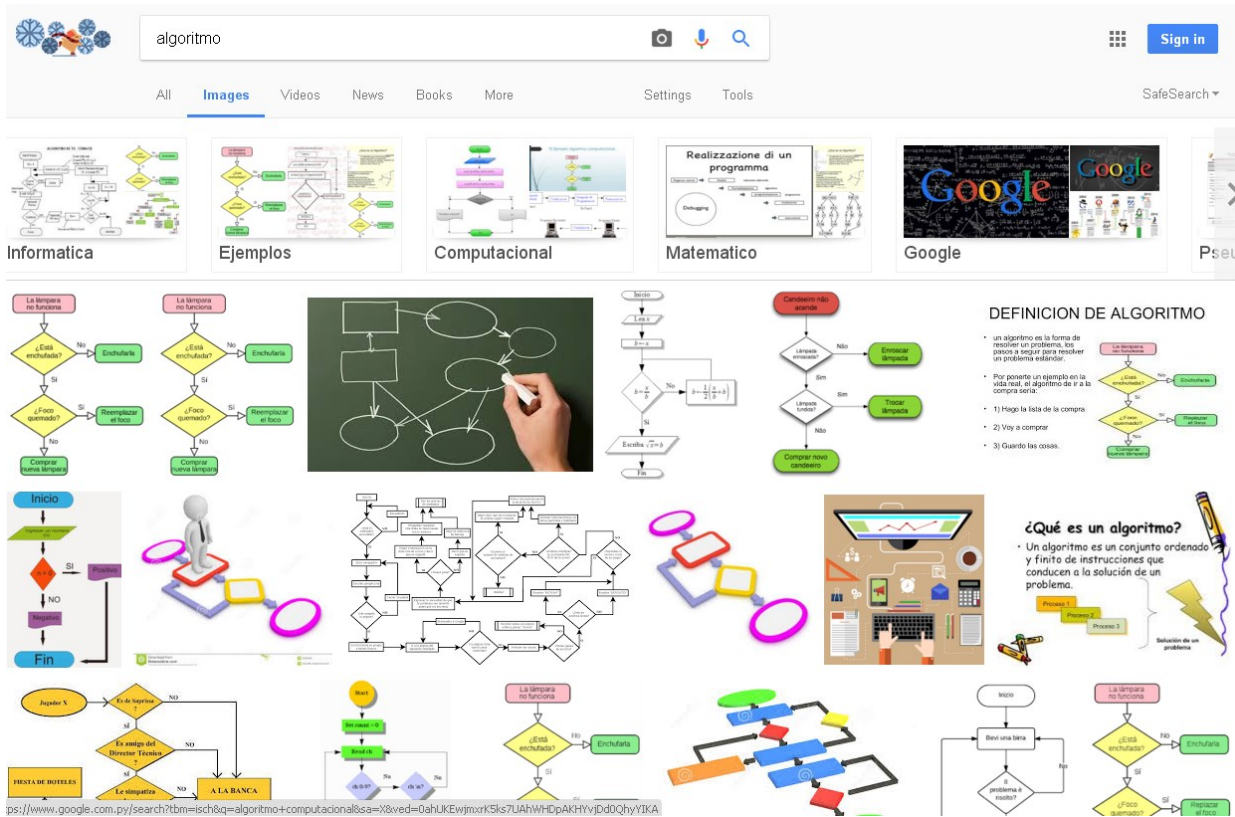
Lastimosamente todavía no podemos programar utilizando el lenguaje natural

# Introducción: Lenguaje de Programación



**Cualquier problema computable en un lenguaje es computable en cualquier otro lenguaje de programación.**

# Algoritmo

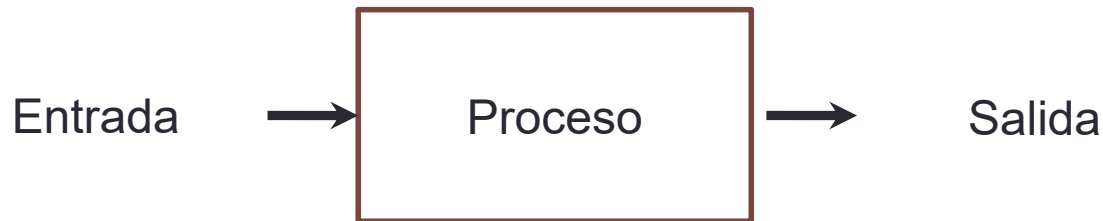


# Un algoritmo es un método para resolver un problema.

# Algoritmos + Estructuras de datos = Programas

# Algoritmo

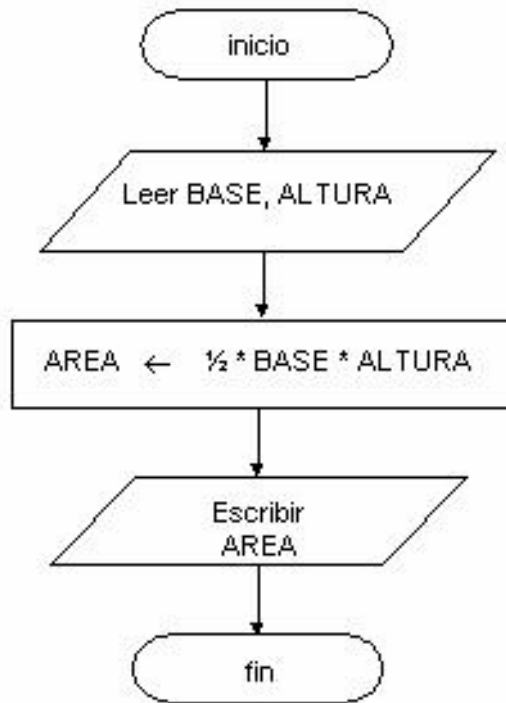
- Un algoritmo debe ser **preciso** e indicar el **orden** de realización de cada paso.
- Un algoritmo debe estar **definido**. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Un algoritmo debe ser **finito**. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos



$1+2+3=$  un algoritmo



# Algoritmo calcular el área de un triángulo



Entrada: BASE, ALTURA (real)

Salida: AREA (real)

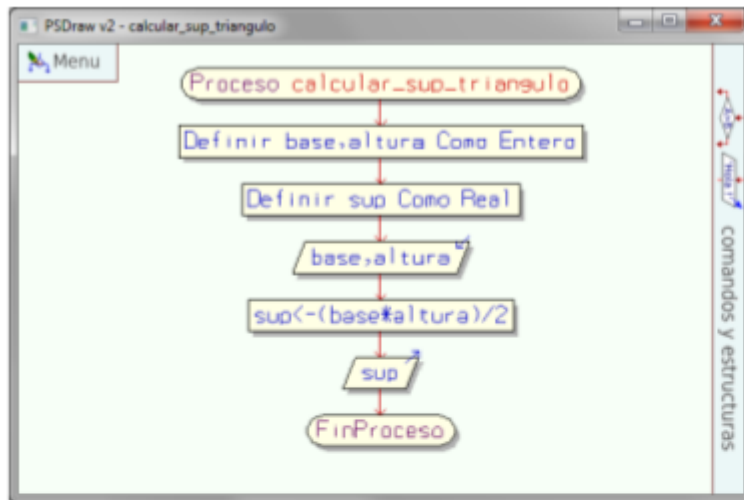
## Pseudo Código

Leer BASE ALTURA

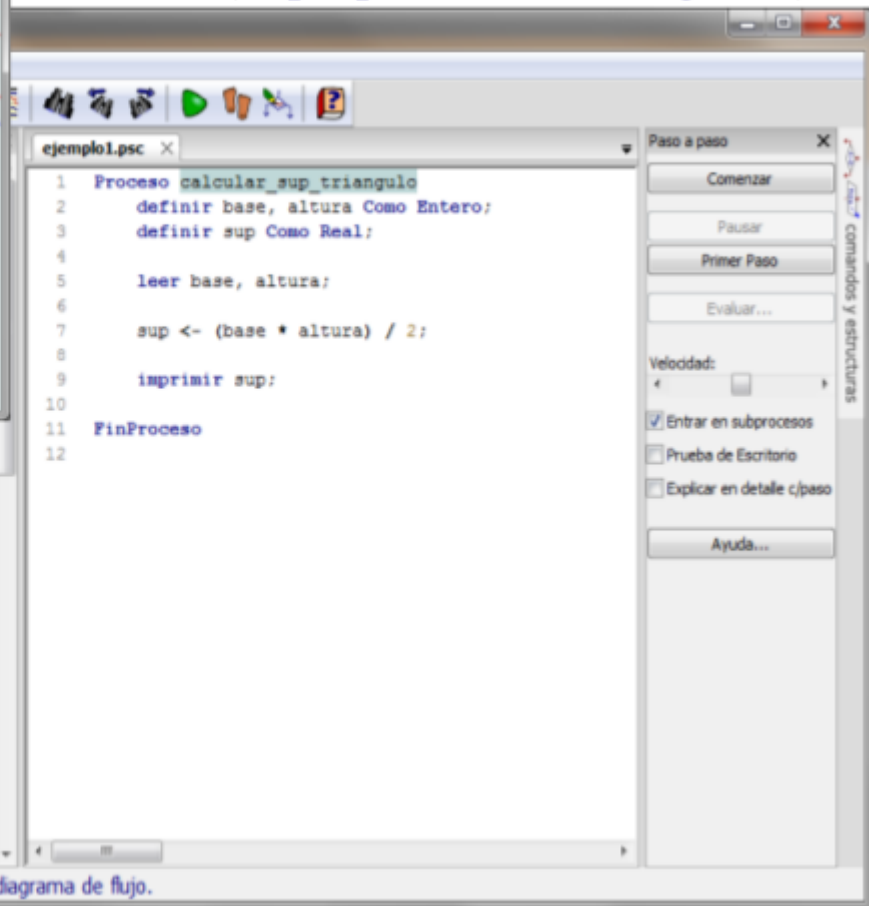
AREA= (BASE \* ALTURA)/2

Escribir AREA

# Ejemplo en PSeInt



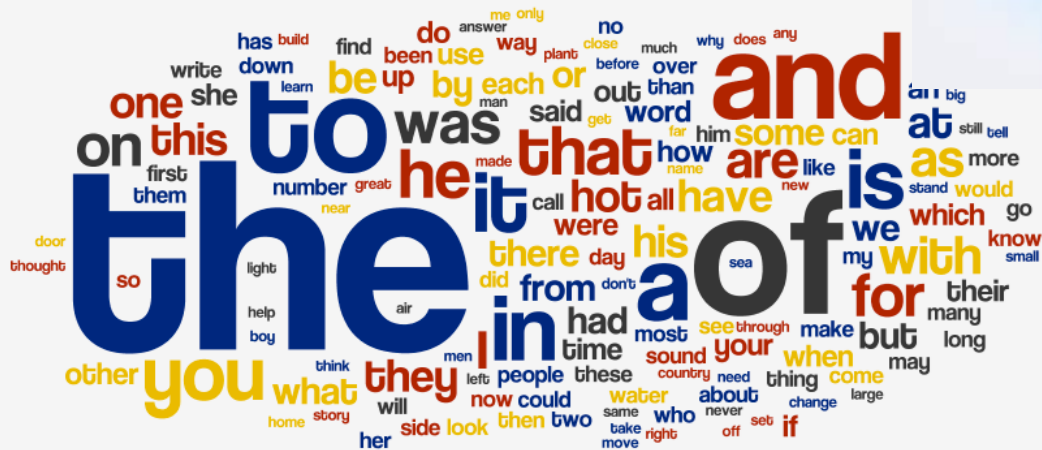
*PSeInt* (<http://pseint.sourceforge.net>)



# Lenguaje de programación



# Idioma Inglés



Es un lenguaje de programación de propósito general. Es multiparadigma (POO, funcional, concurrente, reflectivo, etc)

- Es multiplataforma, portable.
- Es libre y de fuente abierta.
- Es interpretado. De sintaxis sencilla y código compacto.

# C/C++

## Features of Python Programming



# Programa

Una instancia de un algoritmo. Es la representación concreta de un algoritmo en algún lenguaje de programación.

## Lenguaje de programación

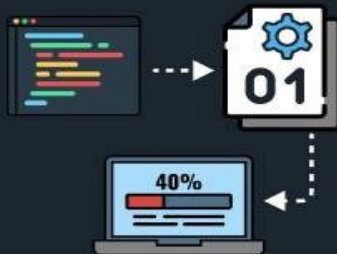
- ♦ Medio para comunicar el algoritmo a la computadora. Es un conjunto de símbolos y las reglas para combinarlos.
- ♦ Existen tres estrategias de traslación de los lenguajes de programación a ejecución en máquina:
  - **Intérpretado:** basado en un software que “comprende” el LP, lee una instrucción a la vez y lo ejecuta. Siempre se necesita el código original. (Por ejemplo Matlab, Python, R, Javascript, etc)
  - **Compilados:** convierte todo el programa escrito en el LP en otro equivalente entendible por la máquina. El resultado es usualmente independiente al programa original (Por ejemplo C, C++, Fortran, Pascal, etc).
  - **Híbridos** ( generadores de código intermedio): traslada el programa original a un programa equivalente que luego es interpretado (Por ejemplo Java)

# TIPOS DE LENGUAJE

## COMPILADO



Convierte el código a binarios que lee el sistema operativo.



## INTERPRETADO



Requiere de un programa que lea la instrucción del código en tiempo real, y la ejecute.



## INTERMEDIO



Se compila el código fuente a un lenguaje intermedio y este último se ejecuta en una máquina virtual.



# Lenguaje de programación

<https://pypl.github.io/PYPL.html>

Popularity of Programming Language

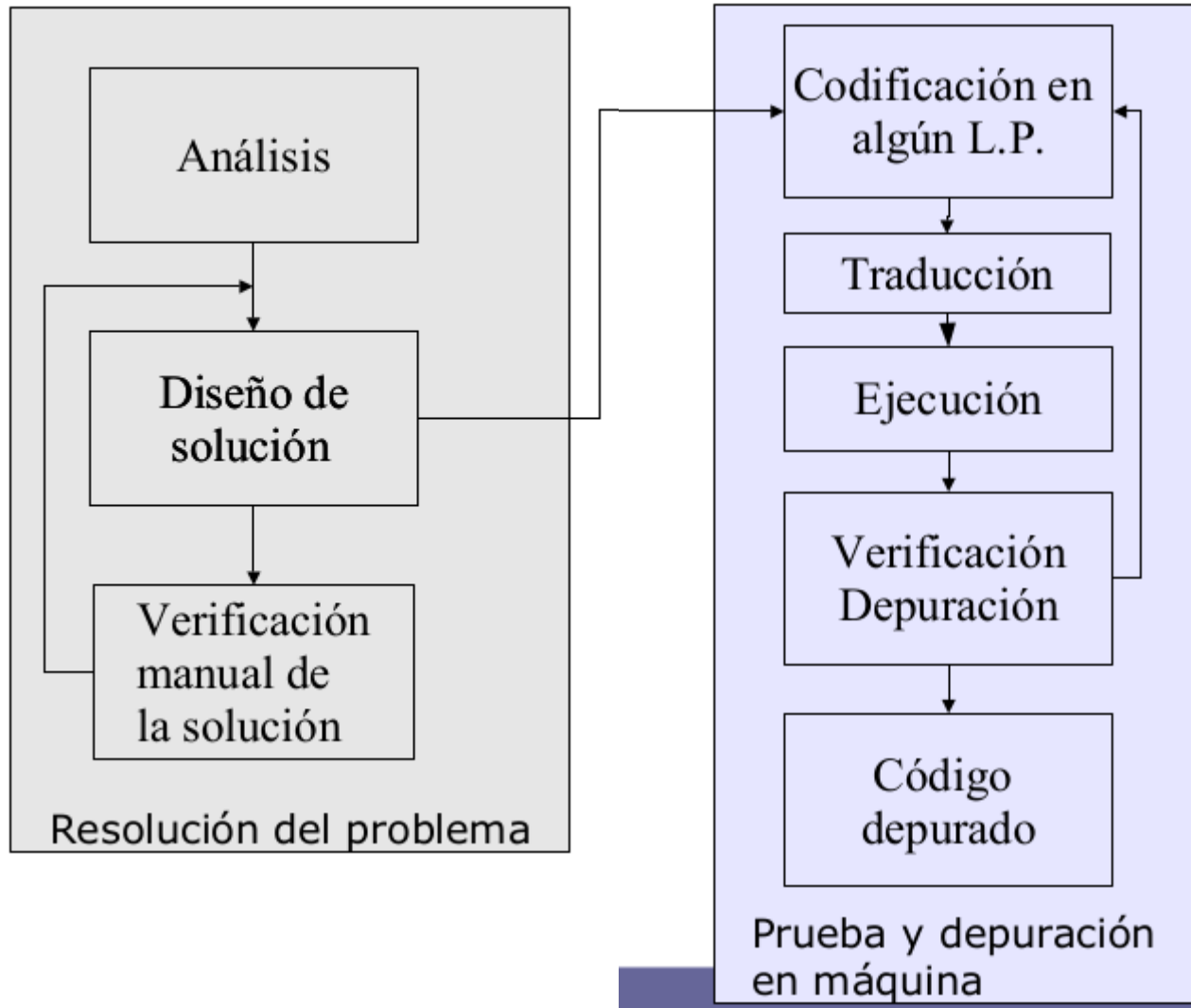
Worldwide, Dec 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.34 %	-1.0 %
2		Java	16.93 %	-0.8%
3		Javascript	9.28 %	+0.3%
4		C#	6.89 %	-0.3%
5		C/C++	6.64 %	-0.3 %
6		PHP	5.19 %	-1.0 %
7		R	3.98 %	-0.1%
8	↑	TypeScript	2.79 %	+1.1%
9	↑	Swift	2.23 %	+0.6%
10	↓	Objective-C	2.22%	+0.1%

TIOBE Programming Community index

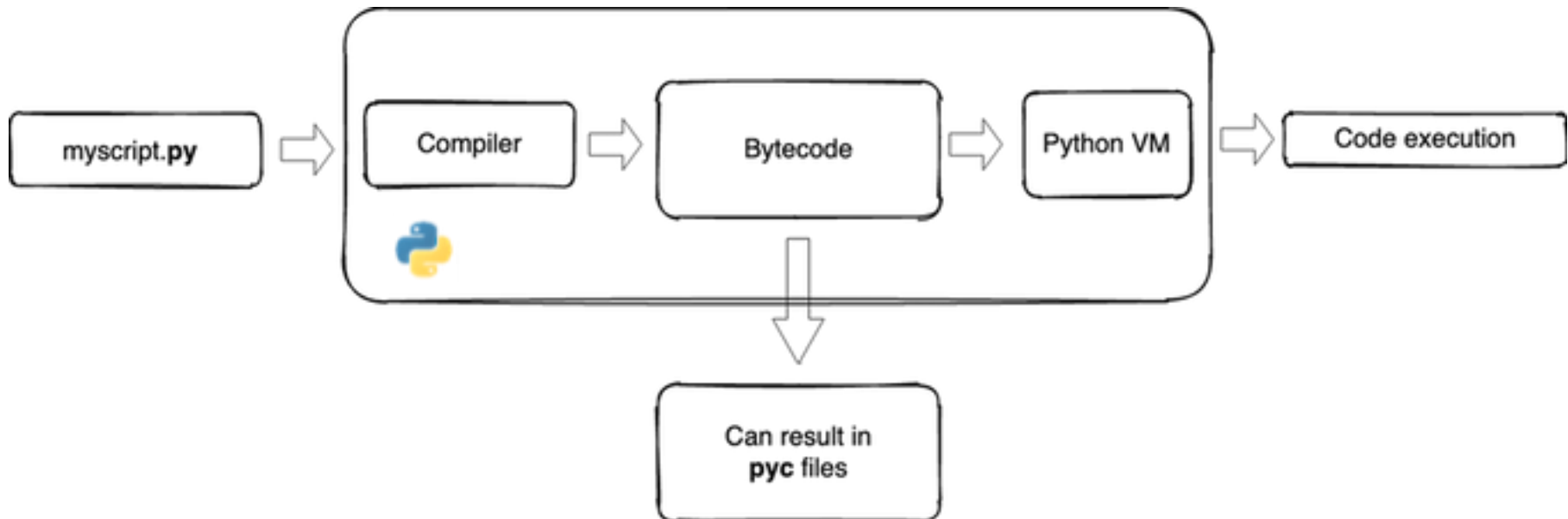
Jan 2023	Jan 2022	Change	Programming Language	Ratings	Change
1	1		 Python	16.36%	+2.78%
2	2		 C	16.26%	+3.82%
3	4	↑	 C++	12.91%	+4.62%
4	3	↓	 Java	12.21%	+1.55%
5	5		 C#	5.73%	+0.05%
6	6		 Visual Basic	4.64%	-0.10%
7	7		 JavaScript	2.87%	+0.78%
8	9	↑	 SQL	2.50%	+0.70%

# Solución de un problema algorítmico





# Modelo de ejecución (interpretación)



- El código fuente es compilado in Byte Code.
- El Byte Code ejecutado en la Máquina Virtual Python.
- El Byte Code siempre regenerado cada vez que el código cambia.

# Aspectos del Lenguaje

**sintaxis** del lenguaje español:

"gato perro niño" no tiene la sintaxis de una oración

"El gato abrazo al niño" es válido sintácticamente

sujeto+ verbo+predicado

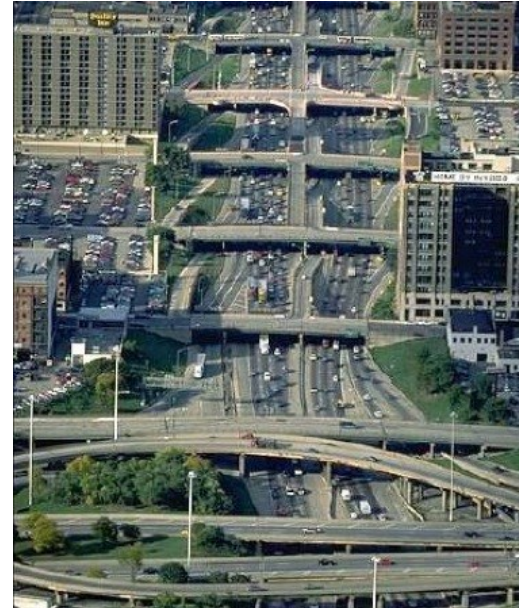
lenguaje de programación:

"hola"5 no es sintácticamente válido

3.2\*5 es sintácticamente válido

variable+ operador+variable

# Que es la Programación Estructurada



# Programación Estructurada

La programación estructurada surgió a finales de los años 70 como un paradigma de programación orientado a mejorar **la claridad, calidad y tiempo de desarrollo** de un programa.

- Boehm, Corrado, & Jacopini, Giuseppe (1966), "[Flow Diagrams, Turing Machines, and Languages with only Two Formation Rules](#)", *Communications of the ACM* 9(5): 366-371.
- Cooper, David C. (1967), "Böhm and Jacopini's Reduction of Flow Charts" (Letter to the Editor), *Communications of the ACM* 10(8) (August): 463,473.
- \* Dijkstra, Edsger W. (1968), "[Go To Statement Considered Harmful](#)" (Letter to the Editor), *Communications of the ACM* 11(3) (March): 147-148.

<https://cse.buffalo.edu/~rapaport/111F04/greatidea3.html>



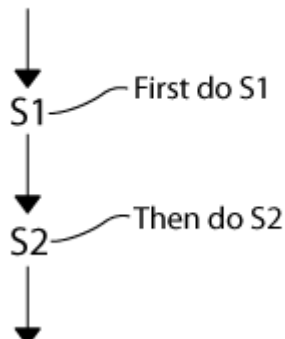
# Programación Estructurada

La programación estructurada es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa, utilizando únicamente subrutinas y tres estructuras: secuencia, selección (if y switch) e iteración (bucles for y while).

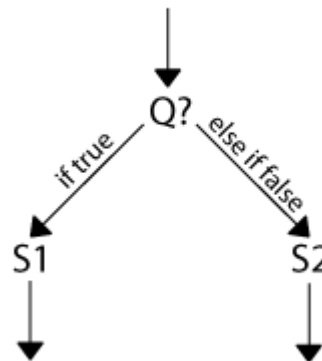
## Teorema del programa estructurado

**Böhm, Jacopini. "Flow diagrams, turing machines and languages with only two formation rules" Comm. ACM, 9(5):366-371, May 1966**

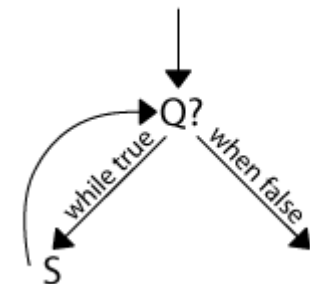
Sequence



Selection



Loop



# Estructura de Selección (PseudoCódigo)

**Si (condición) Entonces**

**..**

**Sino**

**..**

**Fin Si**

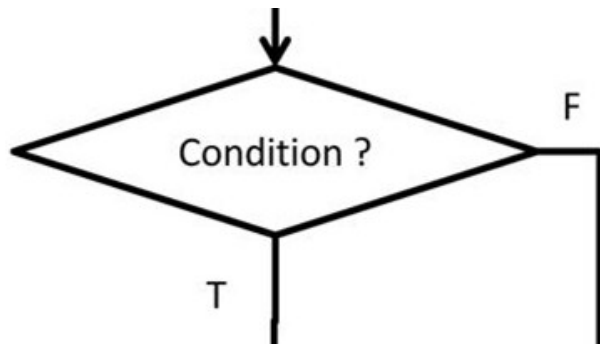
**Según (condición) Hacer**  
**Opcion\_1:**

**Opcion\_2:**

**Opcion\_3:**

**De Otro modo:**

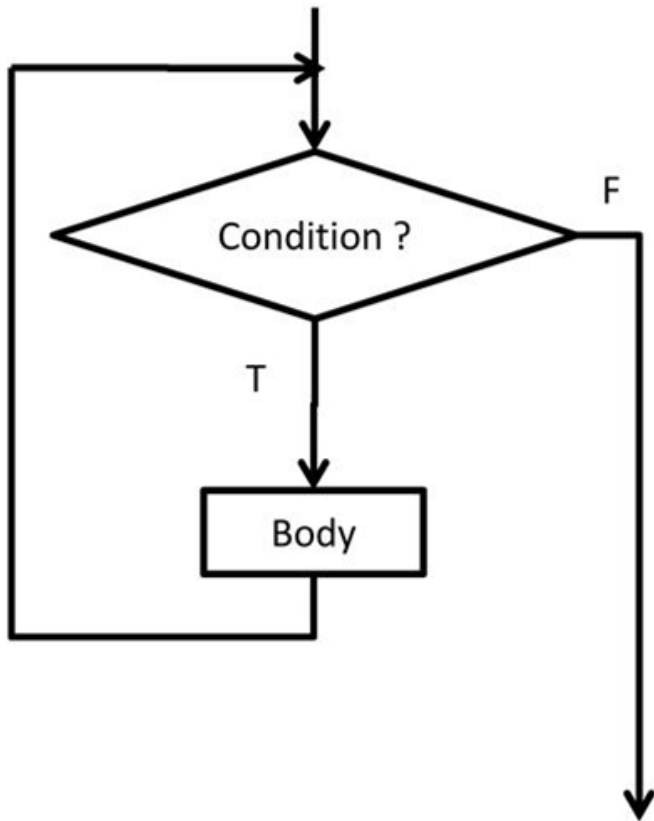
**Fin Según**



# Estructura de Bucles (PseudoCódigo)

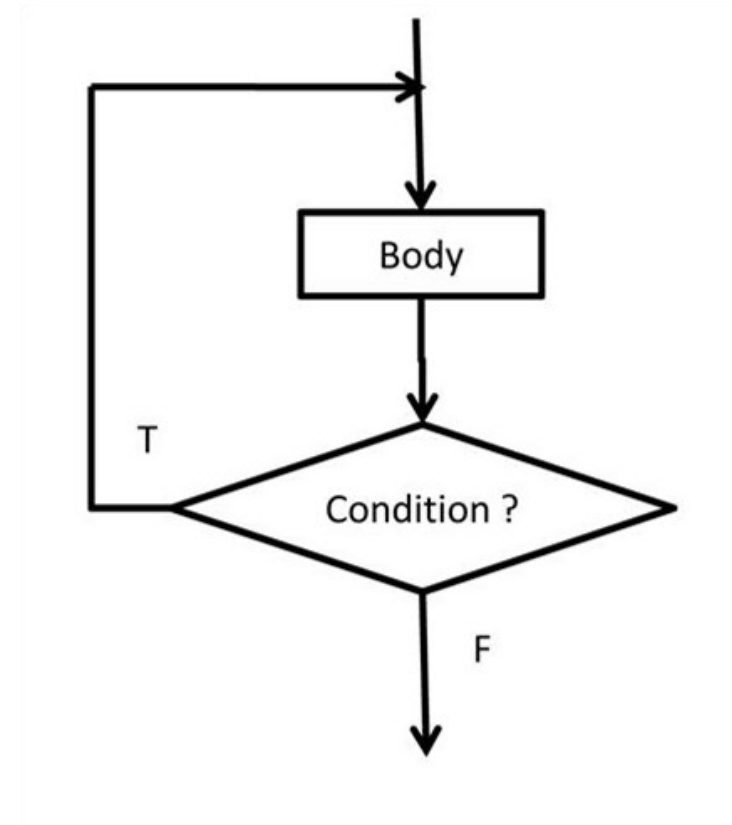
**Mientras (condición) Hacer**

**Fin Mientras**



**Repetir**

**Hasta Que (condición)**





# Conceptos Básicos

## ■ Variable

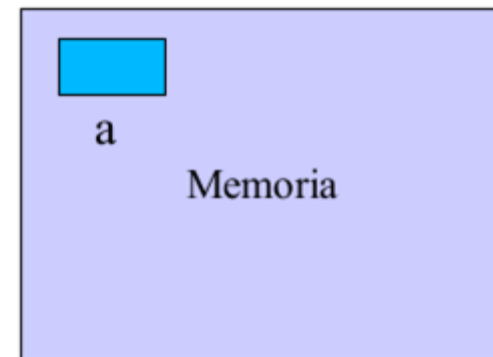
♦ Región de la memoria (principal) que puede ser accedida mediante un nombre ( nombre de variable ). El contenido de esta región puede ser cambiado.

## ■ Constante literal

♦ Cualquier número o cadena de caracteres que forma parte del texto del programa. Es un valor en sí mismo y no puede cambiar. Es almacenado (generalmente ) como parte del programa.

## ■ Identificador

♦ Nombre que es asignado a las variables o subprogramas. Existen identificadores que se denominan “reservados” y que tienen un significado especial dentro de un programa.



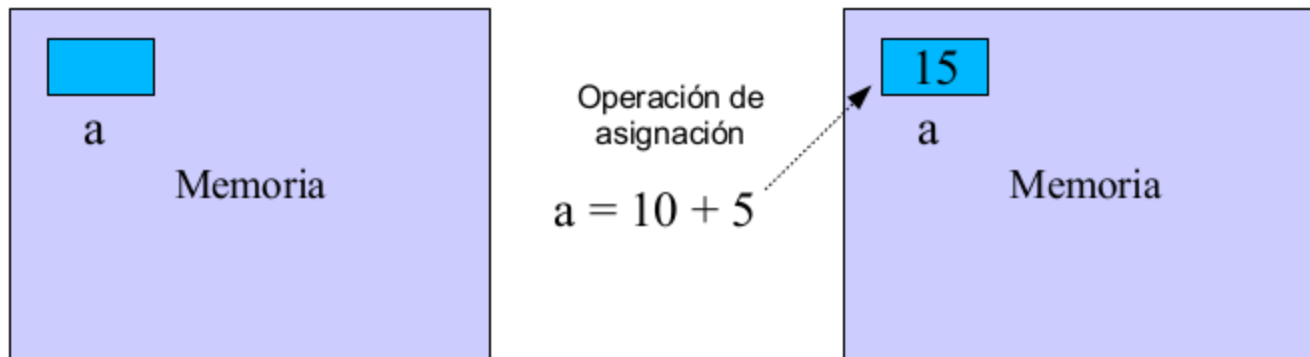
# Conceptos Básicos

## Asignación

- ◆ Es la operación que me permite acceder a cierta posición de memoria y cambiar el valor allí guardado.
- ◆ El operador de asignación está definido por el símbolo “=”. A la derecha debe aparecer una variable y a la izquierda una expresión.

### ◆ Ejemplo:

$a = 10 + 5$  . Indica que a la variable de nombre “a” se le asignará el resultado de evaluar  $10 + 5$ .



# Conceptos Básicos

## ■ Expresión

- ◆ Combinación de constantes, variables y operadores.
  - Ejemplo:  $a + b$
- ◆ Permite “indicar” un cálculo específico así como usualmente lo hacemos en matemáticas.
- ◆ Existen varios tipos de expresiones que utilizamos: aritméticas, relacionales, lógicas y otras.

## ■ Operadores

- ◆ Indica un cálculo específico en una expresión (puede pensar en una función). Por ejemplo: sumar, multiplicar, dividir, restar, etc dos operandos. Se representan mediante “símbolos especiales” :  $+$  ,  $-$  ,  $*$  ,  $/$  , etc.
- ◆ Existen varios tipos: aritméticos, lógicos, relaciones, especiales.

# Conceptos Básicos

## Operadores

- ♦ Aritméticos: +, -, /, \*, % (módulo).

Requieren operandos numéricos.

- ♦ Lógicos: and(&&), or(||), not(!).

Permiten implementar operaciones del álgebra de Boole.

- ♦ Relacionales: ==, <, >, <=, >=, <>.

Permiten comparar contenido de variables. (mayor, igual, menor, distinto, etc).

## Evaluación de operandos

- ♦ Para que una expresión pueda evaluarse (determinar el valor final de la misma) debe tenerse en cuenta qué pasa cuando existe más de un “operador” en la expresión, en qué orden se evaluará?

- Reglas de precedencia: cada operador tiene un peso de precedencia.

Por ejemplo el \* (multiplicación) tiene precedencia sobre el + (suma).

- Reglas de asociatividad: cuando un operador tiene el mismo peso de precedencia me permite determinar cual evaluaré primero.

Generalmente es de izquierda a derecha.

- Puedo “forzar” las reglas de arriba mediante la utilización de paréntesis.

Ejemplo (10+20) \* 5 (Primero se evaluaría la suma y luego la multiplicación)

Asumiendo que  $a$  y  $b$  son variables enteras  $a, b$ ;  
¿Qué valor quería luego de evaluar las siguientes expresiones?

$$a = 5$$

$$b = a + 6$$

$$a = a + 1$$

$$b = a - 5$$

Qué pasa si tenemos la siguiente expresión:

$$a = 1 + 5 * 10 + 4 / 2$$

# Ejercicios

a)  $\frac{M}{N} + 4$

b)  $M + \frac{N}{P - Q}$

c)  $\frac{\text{sen } x + \cos y}{\tan x}$

d)  $\frac{M + N}{P - Q}$

e)  $\frac{P + \frac{N}{P}}{Q - \frac{r}{5}}$

f)  $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$

# Ejercicio

- Leer una lista de 3 números reales y calcular el máximo y el promedio.



# Ejercicio

Escribir un pseudo-código que calcule las raíces, bien reales o imaginarias, de una ecuación de segundo grado.

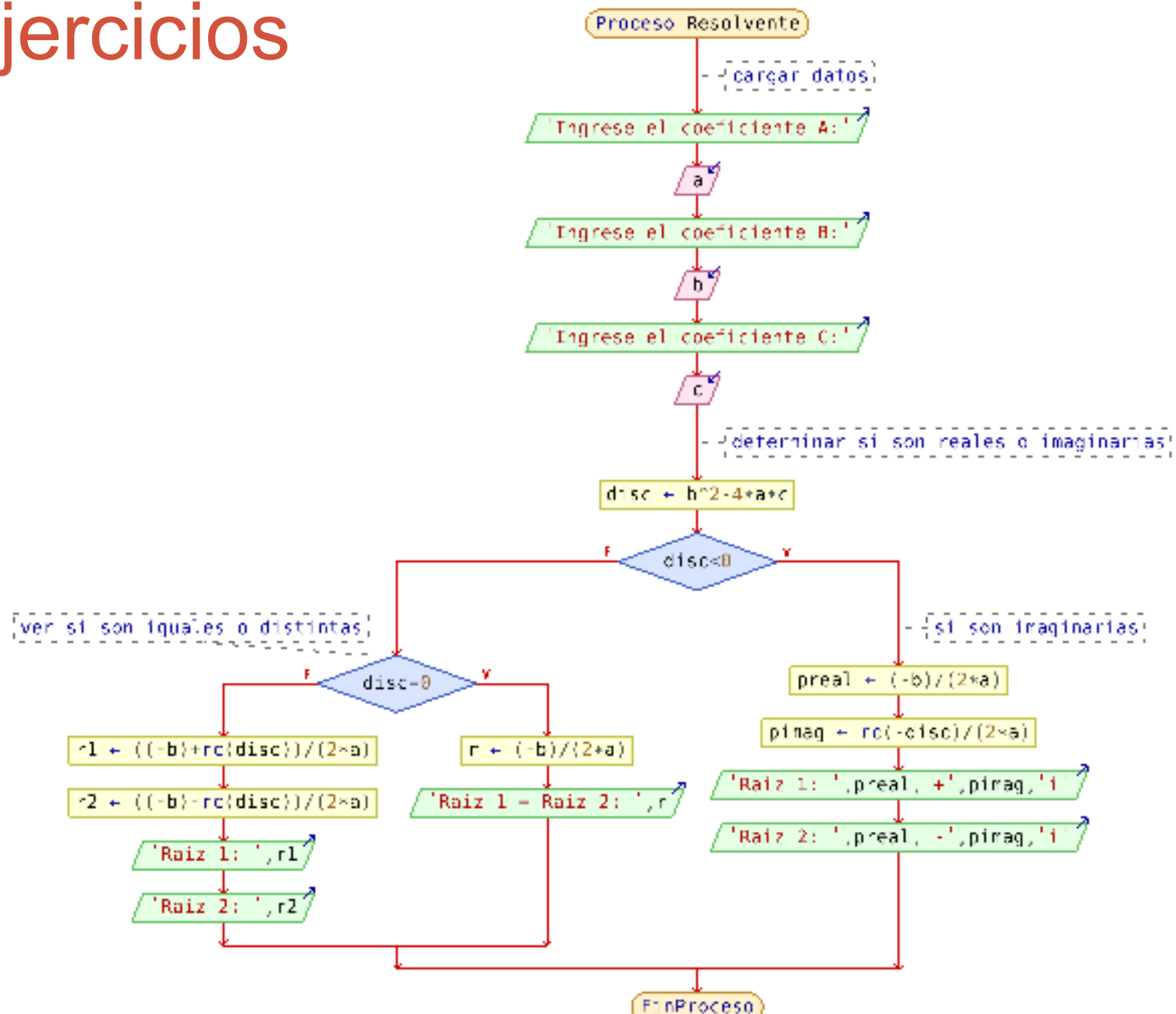
El programa también debe ser capaz de operar con valores nulos para el coeficiente de orden dos (es decir, deberá de ser capaz de resolver ecuaciones de primer grado)

# Ejercicios

---

```
Proceso Resolvente
  // cargar datos
  Escribir 'Ingrese el coeficiente A:'
  Leer a
  Escribir 'Ingrese el coeficiente B:'
  Leer b
  Escribir 'Ingrese el coeficiente C:'
  Leer c
  // determinar si son reales o imaginarias
  disc <- b^2-4*a*c
  Si disc<0 Entonces
    // si son imaginarias
    preal <- (-b)/(2*a)
    pimag <- rc(-disc)/(2*a)
    Escribir 'Raiz 1: ',preal,'+',pimag,'i'
    Escribir 'Raiz 2: ',preal,'-',pimag,'i'
  SiNo
    Si disc=0 Entonces // ver si son iguales o distintas
      r <- (-b)/(2*a)
      Escribir 'Raiz 1 = Raiz 2: ',r
    SiNo
      r1 <- ((-b)+rc(disc))/(2*a)
      r2 <- ((-b)-rc(disc))/(2*a)
      Escribir 'Raiz 1: ',r1
      Escribir 'Raiz 2: ',r2
    FinSi
  FinSi
FinProceso
```

# Ejercicios



# Primer programa en Python

Utilizando Python, escriba un programa que despliegue en pantalla el mensaje “Hola Mundo”

¿Dónde escribimos el programa?

Para las clases usaremos Visual Studio Code

<https://code.visualstudio.com/Download> descargar e instalar

Instalar los plugins para Python

<https://code.visualstudio.com/docs/python/python-tutorial>

<https://marketplace.visualstudio.com/items?itemName=ms-python.python>

Instalar un intérprete de Python

<https://www.python.org/downloads/>

# Primer programa en Python

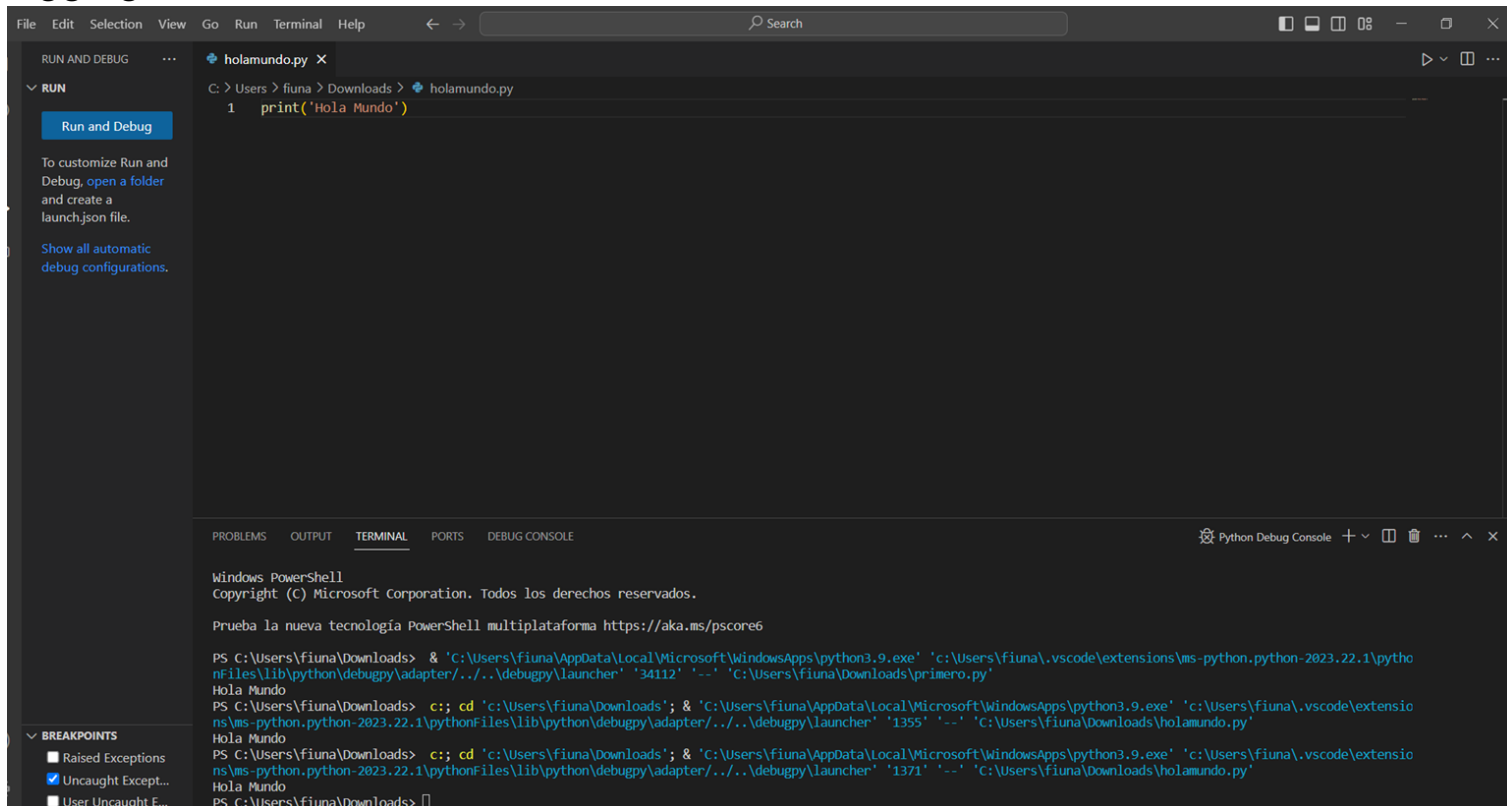
Se crea un programa nuevo: File->New File

Se escribe el programa: Según se muestra en la impresión de pantalla.

Print: Orden de Python que indica que se debe imprimir algo en pantalla (mensaje o variable, cuando es mensaje se coloca este entre comillas “)

Una vez escrito el programa, se guarda: File-> Save as...

Para verificar errores y correr el programa si no hay errores: Run->Start debugging



The screenshot shows the Visual Studio Code interface. The editor window displays a file named `holamundo.py` with the following code:

```
1 print('Hola Mundo')
```

The left sidebar shows the **RUN AND DEBUG** view with a **Run and Debug** button. Below it, there is a message: "To customize Run and Debug, open a folder and create a launch.json file. Show all automatic debug configurations."

The bottom panel shows the **TERMINAL** view with the following output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Fiuna\Downloads> & 'C:\Users\Fiuna\AppData\Local\Microsoft\WindowsApps\python3.9.exe' 'c:\Users\Fiuna\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '34112' '--' 'c:\Users\Fiuna\Downloads\primero.py'
Hola Mundo
PS C:\Users\Fiuna\Downloads> c:: cd 'c:\Users\Fiuna\Downloads'; & 'C:\Users\Fiuna\AppData\Local\Microsoft\WindowsApps\python3.9.exe' 'c:\Users\Fiuna\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '1355' '--' 'c:\Users\Fiuna\Downloads\holamundo.py'
Hola Mundo
PS C:\Users\Fiuna\Downloads> c:: cd 'c:\Users\Fiuna\Downloads'; & 'C:\Users\Fiuna\AppData\Local\Microsoft\WindowsApps\python3.9.exe' 'c:\Users\Fiuna\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '1371' '--' 'c:\Users\Fiuna\Downloads\holamundo.py'
Hola Mundo
PS C:\Users\Fiuna\Downloads>
```

The left sidebar also shows the **BREAKPOINTS** view with the following options:

- ☐ Raised Exceptions
- ☒ Uncaught Except...
- ☐ User Uncaught E...