Junta de Andalucía
Consejería de Educación y Deporte

UT 1
Introducción a la programación
–
Módulo de Programación
1º DAW

Autor: Fran Gómez

Objetivos

- Conocer los **conceptos básicos** relacionados con la programación y el diseño de aplicaciones
- Describir los **paradigmas** de programación más usados
- Clasificar los **lenguajes** de programación.
- Configurar el **entorno de desarrollo**
- Crear **algoritmos y programas** básicos

RA y CE implicados

RA1: Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

a) Se han identificado los bloques que componen la estructura de un programa informático
b) Se han creado proyectos de desarrollo de aplicaciones
c) Se han utilizado entornos integrados de desarrollo.
d) Se han identificado los distintos tipos de variables y la utilidad específica de cada uno.
e) Se ha modificado el código de un programa para crear y utilizar variables.
f) Se han creado y utilizado constantes y literales.
g) Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.
h) Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.
i) Se han introducido comentarios en el código.

Índice de contenidos

Introducción

Mi primer programa en bash/shell

helloWorld.bat - Notepad

File Edit Format View Help

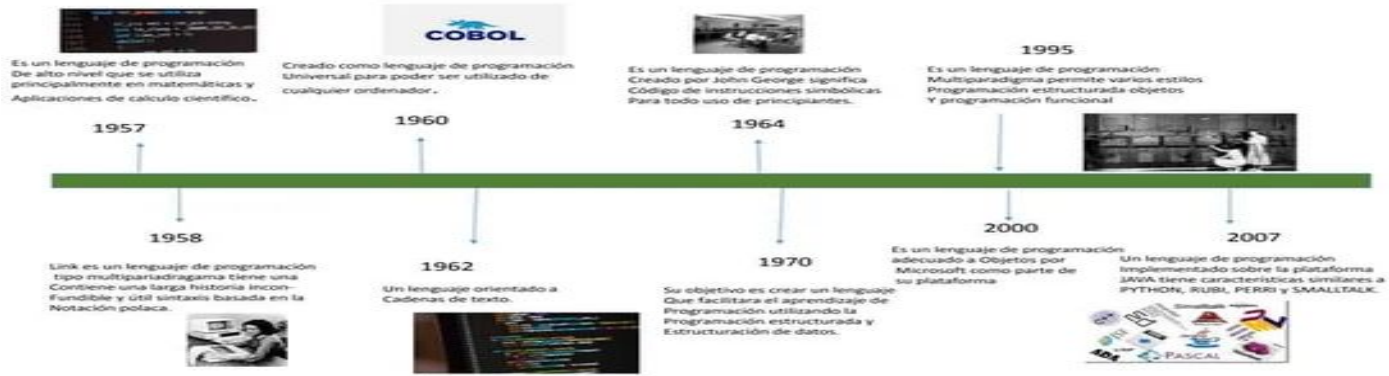
```
ECHO OFF
ECHO Hello World
PAUSE
```

Evolución de la programación

Ejercicio 1

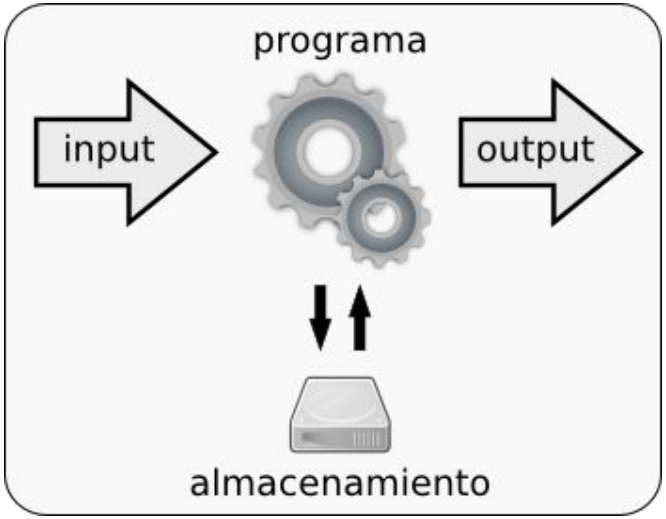
Crea una línea del tiempo con los items que más destacarías en la historia de la programación.

1. Año
2. Nombre (logotipo)
3. Creador
4. Alguna característica



Conceptos básicos

- Programación
- Algoritmo
- Programar
- Código
- Lenguaje binario
- Lenguaje ensamblador
- Lenguajes de alto nivel



- Entrada
- Salida
- Cambio de estado
- Lógica
- Datos

Conceptos básicos

Ejercicio 2

Describe en qué consiste la programación y qué diferencia hay entre los algoritmos y Código.

Clasificación de los lenguajes

- Propósito:
  - General vs específico
- Tipo:
  - Scripting (o interpretado)
  - Compilado
  - Marcado
- Uso:
  - Desarrollo Web
  - Juegos
  - Administración de Sistemas
  - Móvil / Multiplataforma
  - IA, BigData, Machine Learning…
- Plataforma:
  - Web
    - Frontend
    - Backend
  - Desktop
  - Móvil
  - Embedded
- Paradigma:
  - Imperativo
  - Estructurado
  - Orientado a objetos
  - Declarativo

Clasificación de los lenguajes

Ejercicio 3

Crea una tabla comparativa de los lenguajes más populares, clasificándolo
parámetros ant

¿Cuál elegiría
programar? ¿po

Lenguaje	Propósito	Tipo	Uso Principal	Plataforma	Ventajas	Desventajas
Python	General	Interpretado	Ciencia de datos, ML, desarrollo web	Web, ciencia de datos, ML	Fácil de aprender, gran comunidad	Velocidad de ejecución
Java						
JavaScript						
PHP						
C#						
C						
C++						
GO						
Rust						
Swift						
Ruby						
Bash/Batch/PowerShell						

Paradigmas de programación

- **Imperativa** ⇒ Instrucciones paso a paso. “Cómo”
  - **Estructurada** o procedural => Instrucciones siguiendo estructuras. Eliminar Goto.
  - **Modular** ⇒ Divide el programa en partes. Subrutinas.
- **Declarativa** ⇒ Resultado deseado. “Qué”
  - **Funcional** ⇒ funciones matemáticas
  - **Lógico** ⇒ reglas lógicas
- **Orientada a objetos** ⇒ Agrupa Datos y Lógica

Programación Declarativa

Queremos programar la obtención de los clientes de Madrid mayores de 30 años.

```sql
SELECT nombre, apellido
FROM clientes
WHERE ciudad = 'Madrid' AND edad > 30;
```

Declaración, no instrucción: No le dices al ordenador cómo buscar a los clientes, simplemente **declaras** que quieres los nombres y apellidos de aquellos que cumplen las condiciones.

Foco en el resultado: Te centras en el **qué** quieres obtener, no en el **cómo** lo vas a obtener.

Motor de base de datos: El motor de base de datos se encarga de optimizar la consulta y encontrar los datos que cumplen con los criterios especificados.

Queremos programar la elaboración de un sándwich

```c
#include <stdio.h>

int main() {
    printf("Tomando una rebanada de pan\n");
    printf("Untando mantequilla\n");
    printf("Colocando queso\n");
    printf("Tomando otra rebanada\n");
    printf("Cerrando el sándwich\n");
    return 0;
}
```

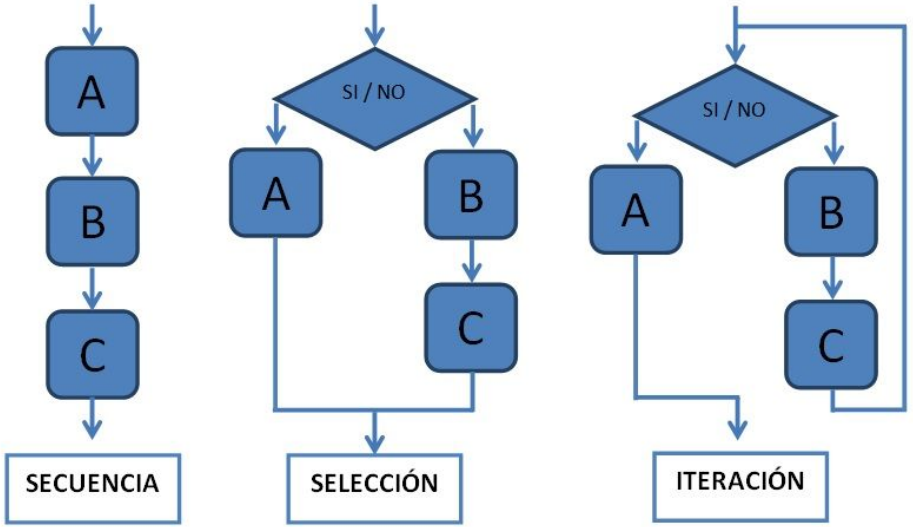
Instrucciones secuenciales: Cada línea de código representa una instrucción específica que se ejecuta una después de la otra.

Estado mutable: El estado del sándwich va cambiando a medida que se ejecutan las instrucciones.

Foco en el proceso: Nos enfocamos en describir *cómo* hacer el sándwich, paso a otro.

Programación Estructurada

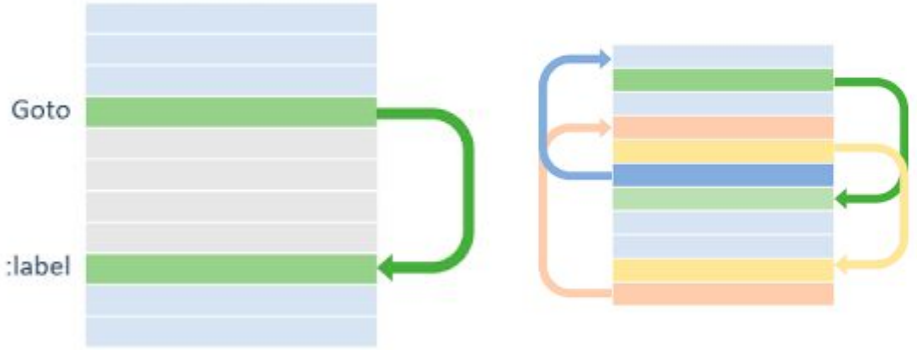
- Secuencial
- Alternativa
- Iterativa



Ideas capacitación
Programación estructurada.

Programación Estructurada

Sentencia GOTO



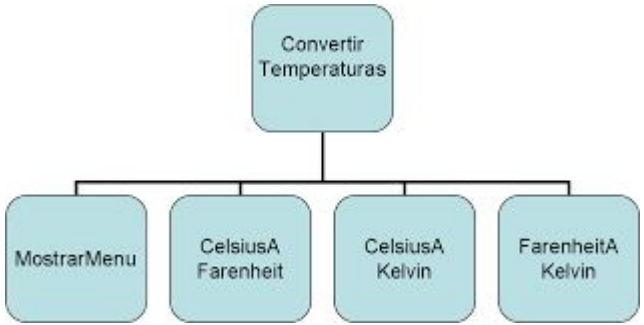

```
goto etiqueta;

...
...
...

etiqueta: sentencia;
```

Código espagueti

Java Hispano

Programación Modular

- Los programas crecen y se hacen más complejos ⇒ dividirlos en módulos
- Módulo = subprograma o rutina ⇒ funciones
- Librerías



Clasificación de los lenguajes

Ejercicio 3 (Continuación)

Completa el ejercicio 3 añadiendo una columna para indicar el paradi

¿Cambi
¿por q

Lenguaje	Propósito	Tipo	Uso Principal	Plataforma	Ventajas	Desventajas	Paradigma
Python	General	Interpretado	Ciencia de datos, ML, desarrollo web	Web, ciencia de datos, ML	Fácil de aprender, gran comunidad	Velocidad de ejecución	Multiparadigma
Java							
JavaScript							
PHP							
C#							
C							
C++							
GO							
Rust							
Swift							
Ruby							
Bash/Batch/PowerShell							

Your Wooclap question will appear here

1. Install the **Chrome** or **Firefox** extension 🧩
2. Click on “Add a Wooclap vote” and make sure you are logged into your Wooclap account 👨‍💻
3. Ensure you are in **presentation mode** on Google Slides 🖥️

wooclap

Estructura

ENCABEZADOS / I

PROGRAMA nomb

CONSTANTES

Declaración y a

VARIABLES

Declaración de

FUNCIONES / MÓD

INICIO

Instrucciones d

FIN

```java
// Importaciones
import java.util.Scanner;

public class ProgramaComple

    // Declaración de varia
    static int edad = 25;
    static String nombre =

    // Funciones / Métodos
    public static String sa
        // Devuelve un mens
        return "Hola " + pe
    }

    public static double ca
        // Calcula el área
        return Math.PI * ra
    }

    // Bloque principal / E
    public static void mair
        String mensaje = sa
        System.out.println(

        double radio = 5;
        double area = calcularAreaCirculo(radio);
        System.out.printf("El área del círculo de radio %.2f es %.2f\n", radio, area)}
}
```

```python
# Importaciones
import math

# Declaración de variables
edad = 25
nombre = "Ana"

# Funciones / Módulos
def saludar(persona, edad):
    """Devuelve un mensaje de saludo con nombre y edad"""
    return f"Hola {persona}, tienes {edad} años."

def calcular_area_circulo(radio):
    """Calcula el área de un círculo"""
    return math.pi * radio ** 2

# Bloque principal / Ejecución
if __name__ == "__main__":
    mensaje = saludar(nombre, edad)
    print(mensaje)

    radio = 5
    area = calcular_area_circulo(radio)
    print(f"El área del círculo de radio {radio} es
{area:.2f}")
```

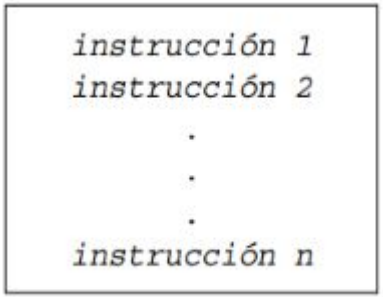
Elementos de un programa

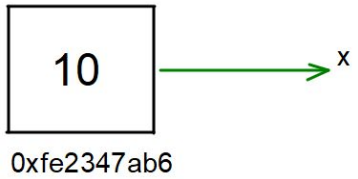
1. **Entrada**
2. Procesamiento
3. Salida
4. Palabras reservadas
5. Comentarios

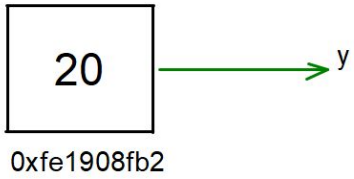
- **Datos**: La información que el programa recibe del usuario o de otros dispositivos para realizar sus cálculos o tomar decisiones.
- **Dispositivos** de **entrada**: Teclado, mouse, escáner, micrófonos, etc.

Elementos de un programa

1. Entrada
2. **Procesamiento**
3. Salida
4. Palabras reservadas
5. Comentarios







- **Instrucciones:** Órdenes o sentencias. Unidad mínima.
- **Variables**: Espacios en la memoria del ordenador donde se almacenan los datos que el programa está utilizando.
- **Constantes**: como una variable cuyo valor inicial no cambia a lo largo del programa
- **Literales**: datos “*a pelo*” en el código.

Elementos de un programa

1. Entrada
2. **Procesamiento**
3. Salida
4. Palabras reservadas
5. Comentarios

- **Operadores**: Símbolos que permiten realizar operaciones matemáticas, lógicas o de comparación sobre los datos.

  +, -, *, /, =

- **Estructuras de control**: Mecanismos que permiten controlar el flujo de ejecución del programa, como las condicionales (si, entonces, sino) y los bucles (para, mientras).

Elementos de un programa

1. Entrada
2. **Procesamiento**
3. Salida
4. Palabras reservadas
5. Comentarios

- **Expresiones:** Son combinaciones de literales, constantes, variables y operadores para ejecutar una operación.

  7 + 3 edad < 12 PI * r^2

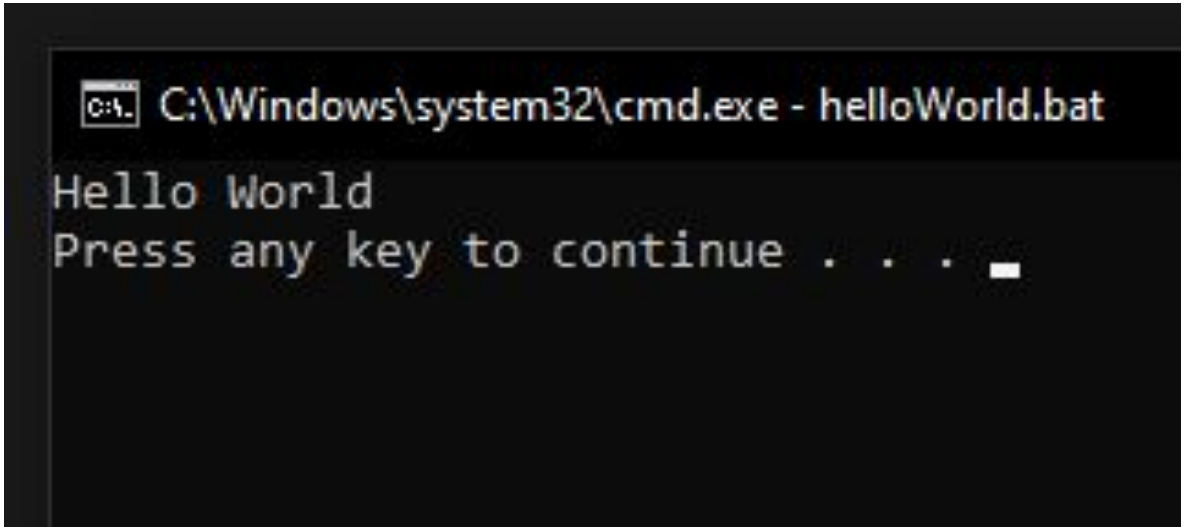
- **Asignación**: Operación que toma el valor de una expresión y lo almacena en una variable

  nombre = “Fran” suma = 2 + 3

Elementos de un programa

1. Entrada
2. Procesamiento
3. **Salida**
4. Palabras reservadas
5. Comentarios

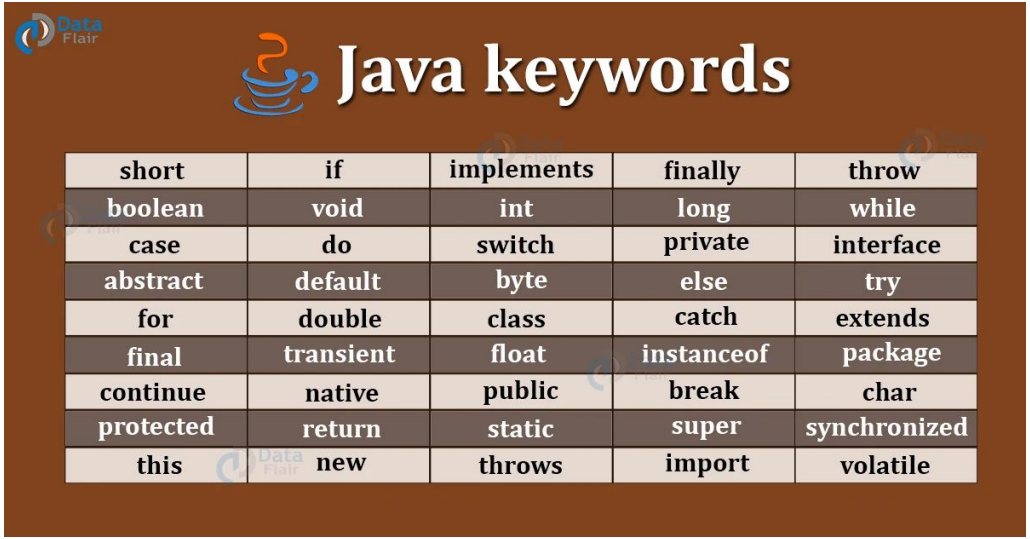
- **Resultados**: La información que el programa genera a partir del procesamiento de los datos de entrada.
- **Dispositivos** de salida: Pantalla, impresora, altavoces, etc.



Elementos de un programa

1. Entrada
2. Procesamiento
3. Salida
4. **Palabras reservadas**
5. Comentarios

Comandos que entiende el programa



No podemos usarlas para nombres de variables, algoritmos, etc.

Elementos de un programa

1. Entrada
2. Procesamiento
3. Salida
4. Palabras reservadas
5. **Comentarios**

- Texto que añade claridad al código, explicando qué hace cada parte del programa. Son muy útiles para documentar el código y facilitar su comprensión por parte de otros programadores o por uno mismo en el futuro.

```
# Imprime el resultado
print (resultado);
```

Ejemplo de Programa

Programa que calcula el área de un círculo.

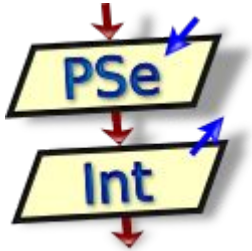
Entrada: El radio del círculo (un número ingresado por el usuario).

Procesamiento: Una variable llamada "radio" para almacenar el valor ingresado. Una constante llamada "pi" con el valor 3.14159. Una sentencia que multiplica el radio por sí mismo y luego por pi para calcular el área.

Salida: El resultado del cálculo (el área del círculo), mostrado en pantalla.

Comentarios: Explicaciones sobre cómo se realiza el cálculo y el significado de cada variable.

Pseudocódigo



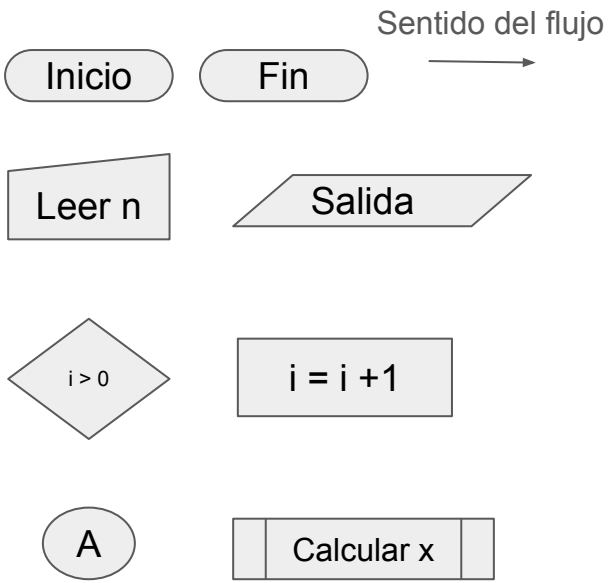

```
1  Algoritmo calcular_area_circulo
2      // Declaración de variables
3      Definir radio, area Como Real
4
5      // Entrada de datos
6      Escribir "Ingrese el radio del círculo: "
7      Leer radio
8
9      // Cálculo del área
10     area ← pi * radio * radio
11
12     // Salida de resultados
13     Escribir "El área del círculo es: ", area
14
15 FinAlgoritmo
```

Crear un algoritmo sin usar un lenguaje de programación concreto.

- Lenguaje informal → Cada uno con sus palabras
- Ayuda a ver la lógica antes de pasar a codificar en un lenguaje real
- Facilita comunicación entre programadores
- Sirve para documentar el código
- Útil para aprender a programar

Diagramas de flujo

Representación gráfica de un programa



Ejercicio

Ejercicio 4

Escribe un algoritmo y dibuja el diagrama para calcular el área de un círculo.

Debe solicitar al usuario los datos de entrada y mostrar el resultado cuando finalice.

No es necesario que uses una sintaxis de un lenguaje concreto sino tus propias palabras.

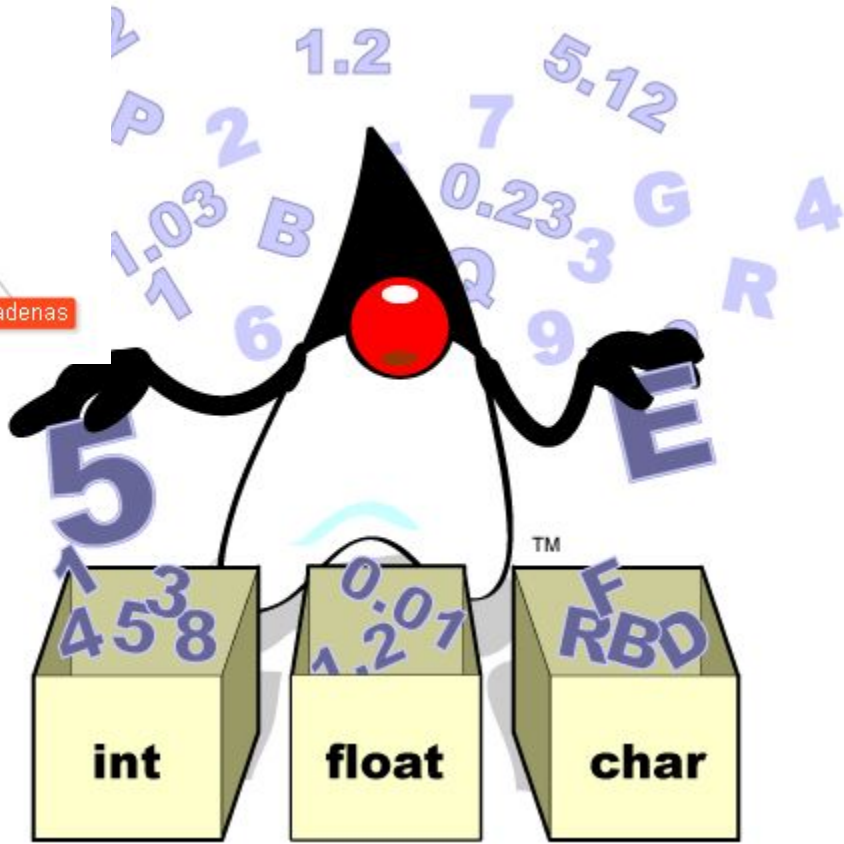
Ejemplos de Algoritmos

RECETA DE COCINA: Cómo preparar una gelatina
1. Inicio
2. Hervir en una olla un litro de agua
3. Agregar el sobre de gelatina
4. Dejarlo a fuego lento cinco minutos revolviendo constantemente
5. Dejar reposar para que tome temperatura ambiente
6. Refrigerar dos hora hasta que cuaje
7. Fin

Tipos de datos

Operadores Lógicos

P	Q	~P	~Q	P o Q	P y Q
Verdadero	Verdadero	Falso	Falso	Verdadero	Verdadero
Verdadero	Falso	Falso	Verdadero	Verdadero	Falso
Falso	Verdadero	Verdadero	Falso	Verdadero	Falso
Falso	Falso	Verdadero	Verdadero	Falso	Falso



Tipos de operadores

Tipo de operador	Símbolos python	Descripción
Aritméticos	`+`, `-`, `*`, `/`, `%`, `**`, `//`	Operaciones matemáticas: suma, resta, multiplicación, división, módulo, potencia, división entera
Relacionales / Comparación	`==`, `!=`, `>`, `<`, `>=`, `<=`	Comparan valores y devuelven `True` o `False`
Lógicos	`and`, `or`, `not`	Combina expresiones booleanas
De asignación	`=`, `+=`, `-=`, `*=`, `/=`, `%=`	Asignan valores a variables, con operaciones combinadas
Concatenación	`+`	Pega dos cadenas de caracteres
De identidad	`is`, `is not`	Comprueba si dos objetos son **el mismo objeto** en memoria
De pertenencia	`in`, `not in`	Comprueba si un elemento **pertenece a una colección**
Bit a bit / a nivel de bits	`&`, `` ` ``	`, ^, ~, <<, >>` `

Precedencia de los Operadores

Descripción	Operadores
Postfijos	`i++, i--`
Unarios	`++i, --i`
Multiplicación y división	`*, /, %`
Suma y resta	`+, -`
Relacionales	`>, <, >=, <=`
Equivalencia	`==, !=`
AND lógico	`&&`
OR lógico	`||`
Asignación	`=`

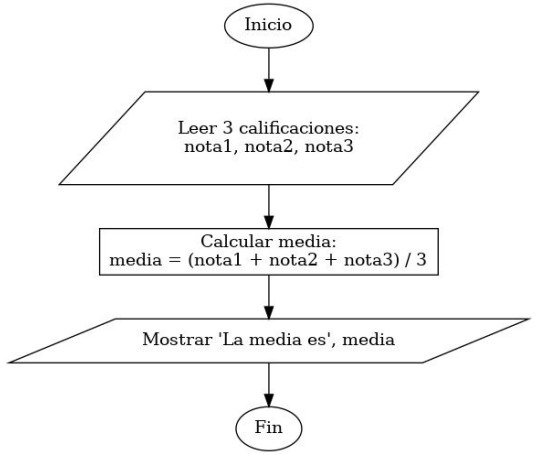
http://es.wikibooks.org/wiki/Programación_en_Java/Precedencia_de_operadores

Ejercicio

Ejercicio 5

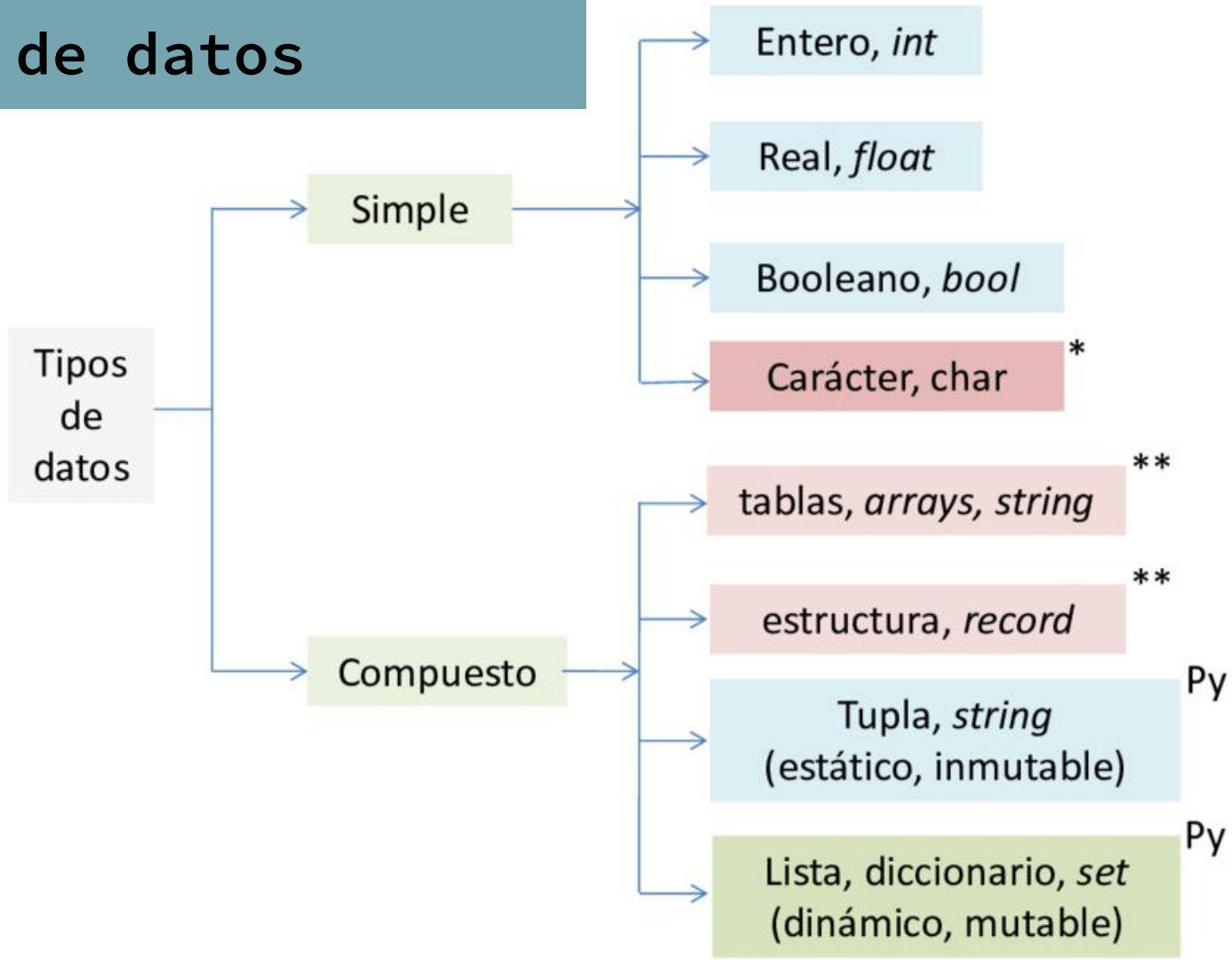
Diagrama y algoritmo que calcule la nota media de tres actividades.

Usa los paréntesis para modificar la precedencia y así realizar correctamente el cálculo



1. Inicio
2. Leer valores de: Calificación1, Calificación2, Calificación3
3. Promedio = (Calificación1 + Calificación2 + Calificación3)/3

Tipos de datos



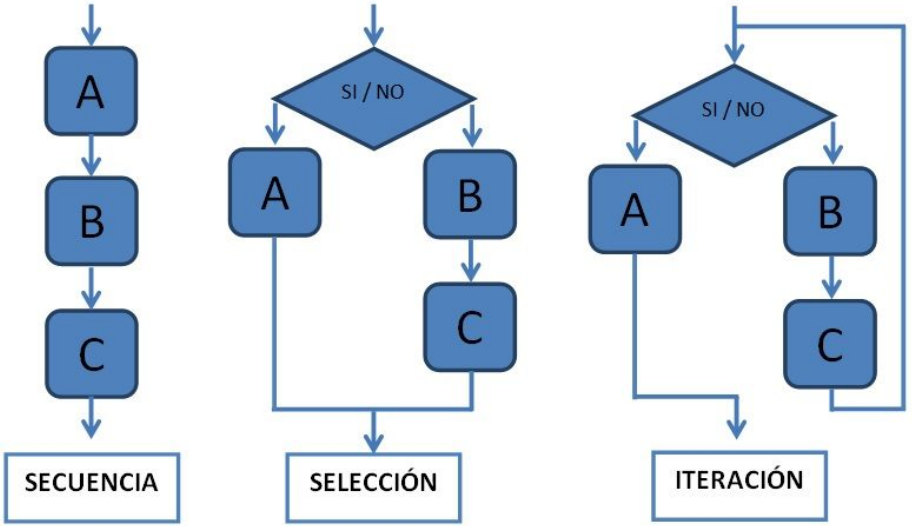
Ejercicio

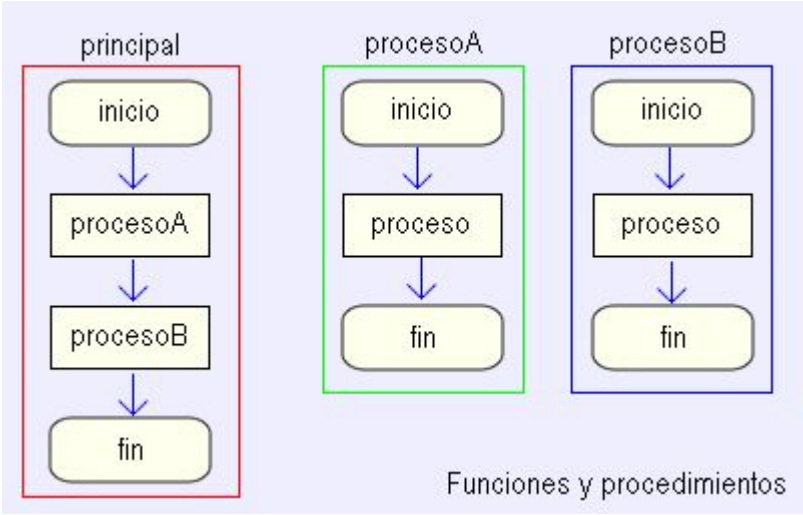
Ejercicio 5 (continuación)

Identifica y añade el tipo dato de cada variable.

Estructuras

- Secuencial
- Selectiva o Alternativa
- Iterativa o Repetitiva
- Modular: Funciones y Procedimientos





Ejercicio

Ejercicio 6

Escribe un algoritmo que a partir del año de nacimiento introducido muestre como resultado si esa persona es mayor de edad o no.

1. Inicio
2. Leer Fecha de nacimiento, Fecha Actual
3. Edad = Fecha actual – Fecha de nacimiento
4. Edad >=18
5. SI
6. "Credencial Autorizada"
7. No
8. "Credencial No Autorizada"
9. FIN

Ejercicio

Ejercicio 7

Escribe un algoritmo que a partir de tres calificaciones, calcule la media, y a partir de la media, indique si el alumno está aprobado o no.

Matemático:

Calcular el promedio de tres calificaciones; si es mayor a 6, mostrar aprobado, si no mostrar reprobado.

1. Inicio
2. Leer valores de: Calificación1, Calificación2, Calificación3
3. Promedio = (Calificación1 + Calificación2 + Calificación3)/3
4. **Si promedio** >5.9, entonces **Escribir "Aprobado"** **Si no** **Escribir "Reprobado"** **Fin Si**
5. FIN

Codificar en Python

IDLE
Integrated Development and Learning Environment

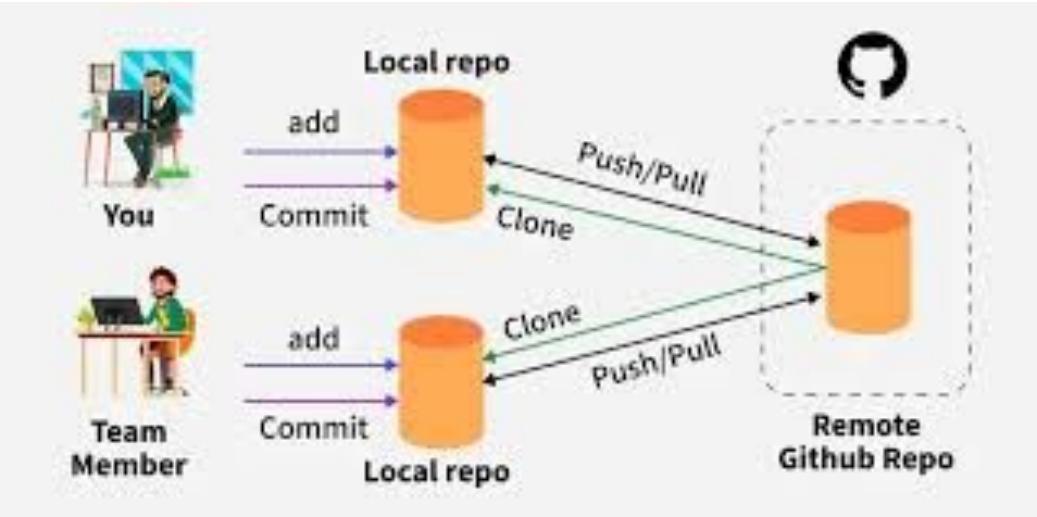
Preparación del entorno:

1. Instalamos **Python** descargando el instalador de su web: python.org
2. Instalamos el IDE **IDLE** o **VSCode**: code.visualstudio.com
   a. También podemos instalar la extensión de VSCode para Python
   b. Desactiva Copilot
3. Introducimos el código para crear un programa *Hola mundo*:

```
print("Hola mundo!");
```

Empezando con Git

1. Instalar Git
2. Configurar Git
3. Crear cuenta en GitHub
4. Crear repositorio en GitHub
5. Crear nuestro programa, iniciar repositorio local, y subirlo



Proyecto local
▼
git init
▼
git add
▼
git commit
▼
git push
▼
GitHub

Ejercicios

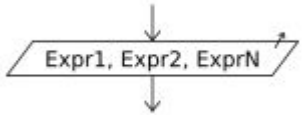
Para los siguientes ejercicios vamos a realizar los 3 pasos:

- ➔ Diagrama de flujo (opcional)
- ➔ Pseudocódigo (opcional)
- ➔ Código en Python (En blog todo en la misma entrada y en Github)

8. Hola mundo con variables: Escribe un programa que muestre por pantalla “Hola, *nombre*” donde *nombre* es una variable que debes declarar e inicializar previamente con el valor de tu nombre.

Ejercicios

Estructuras de control **secuencial**

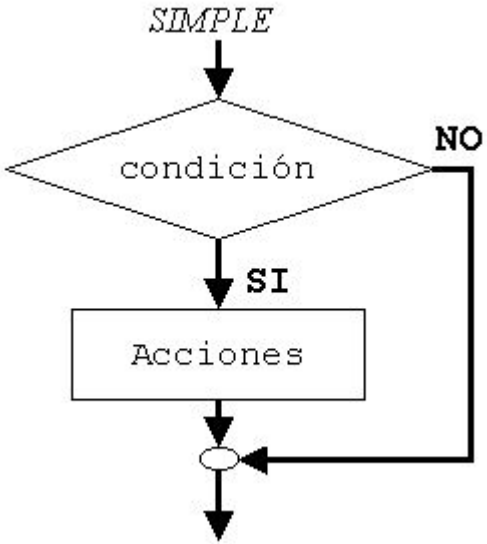


9. Hola usuario: Escribe un programa que muestre por pantalla “Hola “ y el nombre introducido por teclado. Ejemplo: si introducimos “Pepito” mostraría “Hola Pepito”. ¿Cómo harías para mostrar una admiración al final? Ej: “Hola Pepito!”

10. suma 2 números: Escribe un programa que sume dos números enteros introducidos por teclado y muestre el resultado por pantalla. Utiliza variables para cada número.

#¡NO olvides incluir comentarios en tu código!

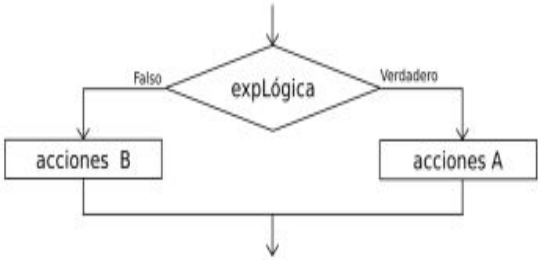
Estructuras de control alternativas simples



11. Login: Escribe un programa que pida el nombre de usuario y si es correcto muestre “¡Bienvenido <usuario>!”.

Un usuario será correcto si coincide con el valor establecido literalmente por el programador en el código.

Sentencias de control alternativas múltiples



12. Ahora añade que si el usuario no es correcto, entonces muestre "**Usuario incorrecto**"

13. Mayor o menor: Solicitar al usuario dos números y mostrar cuál es el mayor.
¿Cómo harías para evaluar además si son iguales?

14. Par o impar: Pedir un número al usuario y determinar si es par o impar.

Sentencias de control alternativas

15. Año bisiesto: Determinar si un año ingresado por el usuario es bisiesto.

Un año es bisiesto si cumple las condiciones:

a. Un año es bisiesto si es divisible entre 4.

b. Excepto los años que son múltiplos de 100, que no son bisiestos.

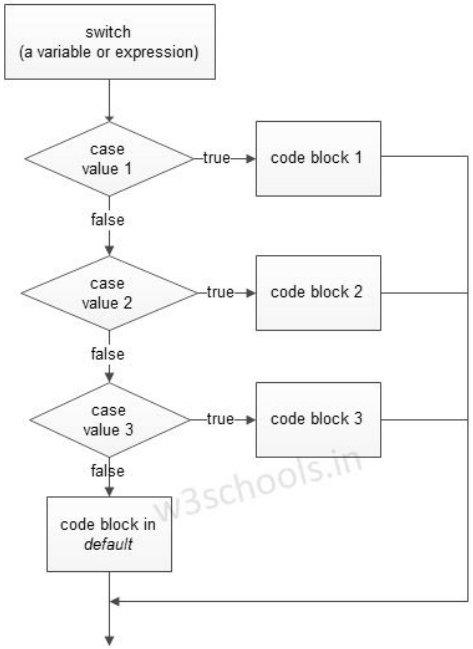
c. Salvo que además sean divisibles entre 400, en cuyo caso sí lo son.

Ejemplos:

- 2024 es bisiesto → porque es divisible entre 4
- 1900 no bisiesto → divisible entre 4 pero también entre 100
- 2000 bisiesto → divisible entre 4, también por 100 pero también por 400

Sentencias de control alternativas

Estructura de control alternativa para muchas opciones: **Según**



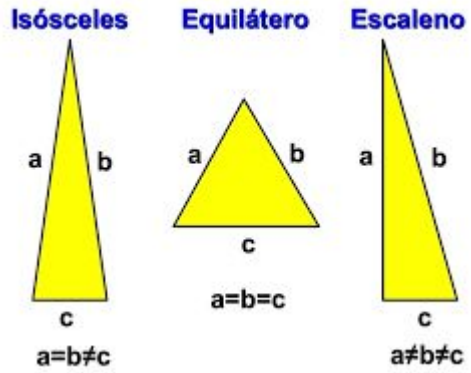
16. Calificación: Solicitar una nota numérica y mostrar la calificación correspondiente (Sobresaliente, Notable, Bien, Suficiente, Insuficiente) según una escala determinada.

0-4	Insuficiente
5	Suficiente
6	Bien
7-8	Notable
9-10	Sobresaliente

17. Calculadora básica:

Crear un programa que realice las cuatro operaciones básicas (suma, resta, multiplicación y división) según la opción que seleccione el usuario.

Sentencias de control alternativas



18. Triángulo: Dado tres lados, determinar si pueden formar un triángulo **y, en caso afirmativo, indicar si es equilátero, isósceles o escaleno.**

Tres lados forman un triángulo si la suma de las longitudes de dos cualesquiera es mayor que la longitud del tercero

Equilátero: todos los lados iguales
Isósceles: dos lados iguales
Escaleno: ningún lado igual

Sentencias de control iterativas

¿Cómo harías el programa de Hola mundo para que mostrase el mensaje dos veces?

¿Y para que lo hiciese 100 veces?

Bucles: Los bucles son estructuras de programación que nos permiten repetir un flujo sin duplicar código.

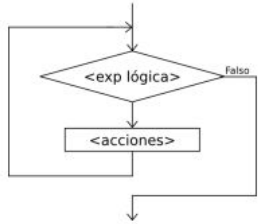
Contador: variable que registra las iteraciones

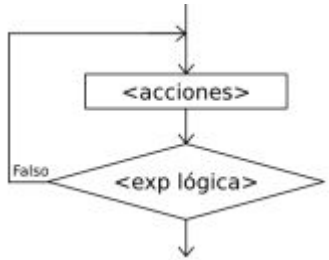
Ejercicio 19: Muestra 100 líneas con el texto "¡Hola mundo!" y enuméralas

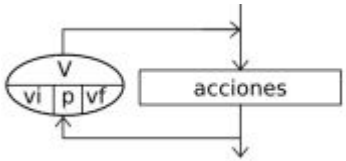
```
1.  ¡Hola mundo!
2.  ¡Hola mundo!
…
100. ¡Hola mundo!
```

Tipos de bucles o estructuras iterativas

While	MIENTRAS (condición sea verdadera) HACER <Bloque de código> FIN_MIENTRAS
While	HACER <Bloque de código> MIENTRAS (condición sea verdadera)
While	REPETIR <Bloque de código> HASTA QUE (condición sea verdadera)
for	PARA <Variable_Contadora> DESDE <Valor_Inicial> HASTA <Valor_Final> CON_PASO <Cantidad_de_Paso> HACER <Bloque de código> FIN_PARA



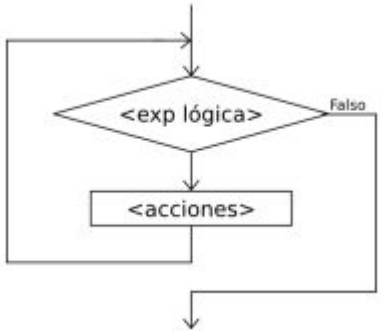




Tipos de bucles o estructuras iterativas

Tipo de Bucle	¿Cuántas veces se ejecuta?	¿Cuándo se evalúa la condición?
Mientras (While)	Número Indefinido	Al principio (Puede no ejecutarse)
Hacer Mientras	Al menos una vez	Al final
Repetir Hasta	Al menos una vez	Al final
Para (For)	Número Definido (Iteración)	N/A (Itera sobre una secuencia)

Sentencias de control iterativas



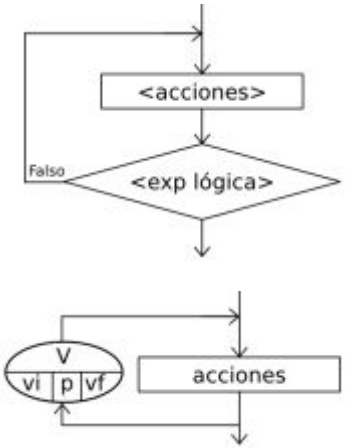
Iteración con salida al principio: **While**

20. Contar hasta un número:

Escribe un programa que pida al usuario un número entero positivo y cuente desde 1 hasta ese número.

Ejemplo: si introduzco el 3, el programa muestra: 1, 2, 3.

Sentencias de control iterativas



21. **Contar hasta un número usando “Hacer mientras”**
22. **Contar hasta un número usando “Repetir hasta”**
23. **Contar hasta un número usando “Para”**

Sentencias de control iterativas

24. **Suma hasta pulsar tecla cero:**

    Suma de números hasta que el usuario ingrese 0.

    Cada vez que el usuario introduzca un número lo sumo a lo que ya tenía y le pregunto otra vez.

    Este tipo de algoritmos se llaman acumuladores

Sentencias de control iterativas

25. **Tabla de multiplicar:** Imprimir la tabla de multiplicar de un número.
26. **Suma de números:** Calcular la suma de los números del 1 al 100.
27. **Suma inversa**: realiza la suma desde el 100 al 1
28. **Suma pares**: realiza la suma sólo de los números pares del 1 al 100
29. **Factorial:** Calcular el factorial de un número introducido por el usuario

Módulos

Dividir las funcionalidades del programa en subprogramas:

- Procedimientos
- Funciones

Objetivos:

- Reutilizar código
- Simplificar código (divide y vencerás)
- Organizar mejor el código

Procedimiento

- Escribimos cada subprograma con la siguiente sintaxis:

```
Procedimiento nombre (tipo parámetro1, ... tipo parámetroN)
      <instrucciones>
Fin Procedimiento
```

- Invocamos el procedimiento así:

```
nombre (parámetro1, . . . parámetroN)
```

- No retorna ningún valor

Función

- Escribimos cada subprograma con la siguiente sintaxis:

```
Función nombre (tipo parámetro1, ... tipo parámetroN) : tipoRetorno
      <instrucciones>
      retornar x
Fin Función
```

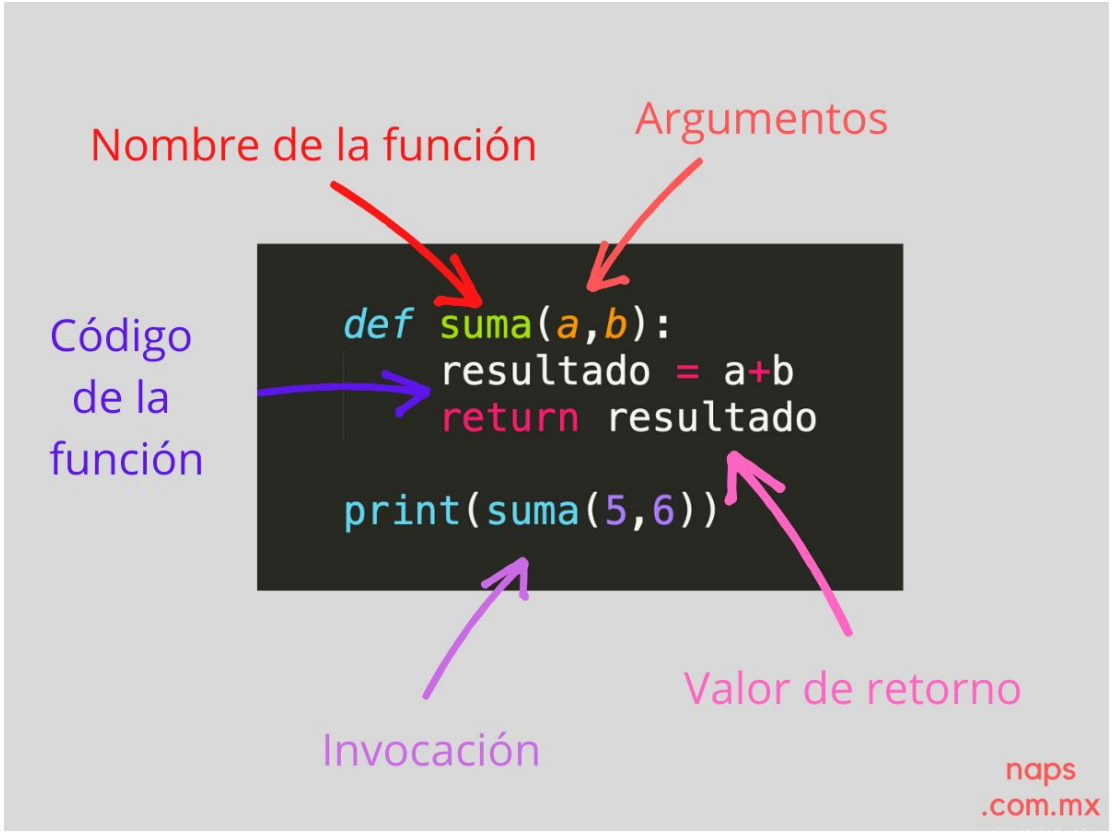
- Invocamos el procedimiento así:

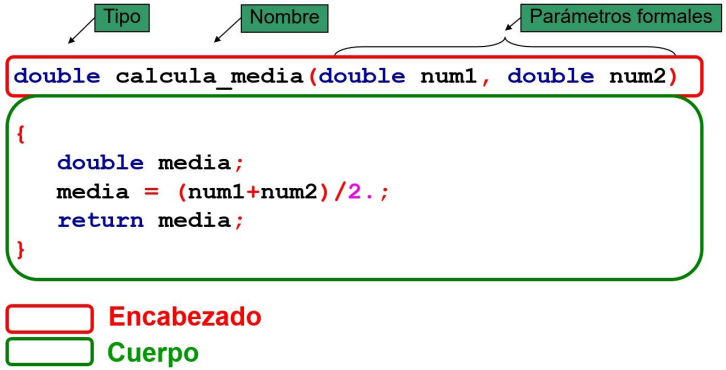
```
nombre (parámetro1, . . . parámetroN)
```

Puede usarse en una expresión

- Retorna valor de salida

Ejemplos funciones



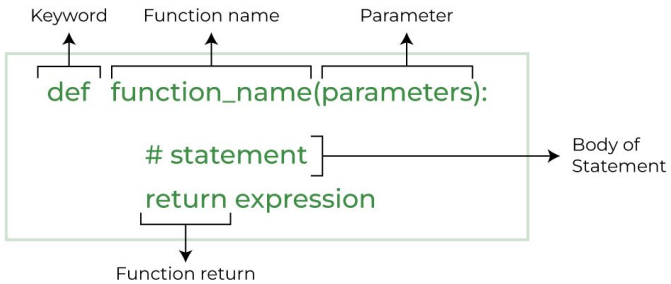


https://www.w3schools.com/python/python_functions.asp

Módulos: Ejercicios

30. Crear un programa que invoque a una función que devuelva si el número leído en el programa principal es par o impar.

31. Adaptar el ejercicio anterior para hacerlo mediante un procedimiento en lugar de una función



Funciones predefinidas

También existen funciones ya definidas en el lenguaje

Python Built-in Functions

		Built-in Functions		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	

https://www.w3schools.com/python/python_ref_functions.asp

Funciones predefinidas

32. Escribe un programa que dado un número decimal negativo lo pase a positivo y lo redondee a un número entero.

    Utiliza funciones predefinidas.

Python Built-in Functions

Funciones predefinidas para los Strings

33. Escribe un procedimiento que muestre en mayúsculas la cadena introducida por el usuario.

34. Crea una función que cuente cuántas vocales hay en una cadena dada y las reemplace por una ‘X’.

35. Modifica el programa de login para que además del nombre de usuario pida una contraseña y valide que la contraseña tenga al menos 8 caracteres.

https://www.w3schools.com/python/python_ref_string.asp