

cÓNQUER



Pseudocódigo en *PseInt* (II)

Módulo 2: IDE, Pseudocódigo y Python para principiantes



Contenidos

1. Repasando conceptos anteriores
 - a. Variables: declaración, inicialización y modificación (op. algebraicos +, -, *, /)
 - b. Instrucciones lectura/escritura
 - c. Estructura condicional SI-ENTONCES (op. relacionales =, ≠, <, >, ≤, ≥)
 - d. Esqueleto básico de un algoritmo en *PseInt*
2. Aprendiendo nuevos conceptos
 - a. Op. algebraicos: ↑, MOD
 - b. Constantes: número PI
 - c. Estructura condicional SI-ENTONCES (op. lógicos Y, O, NO)
 - d. Estructura selectiva SEGÚN
3. Algunos ejemplos
4. Ejercicios de la semana

1. Repasando conceptos anteriores (i)

Variables

1. Repasando conceptos anteriores (i)

Variables

¿Qué es una variable?

- Es un espacio reservado en memoria que tiene asignado un identificador (ejemplo, *sumaTotal*)
- Al proceso de asignar un identificador a un espacio de memoria se le conoce como “**definir una variable**”
- Se utiliza para guardar datos que, posteriormente, puedan ser utilizados por nuestro algoritmo/ programa

1. Repasando conceptos anteriores (i)

Variables

¿Qué es una variable?

- Es un espacio reservado en memoria que tiene asignado un identificador (ejemplo, *sumaTotal*)
- Al proceso de asignar un identificador a un espacio de memoria se le conoce como “**definir una variable**”
- Se utiliza para guardar datos que, posteriormente, puedan ser utilizados por nuestro algoritmo/ programa

Memoria

1. Repasando conceptos anteriores (i)

Variables

¿Qué es una variable?

- Es un espacio reservado en memoria que tiene asignado un identificador (ejemplo, *sumaTotal*)
- Al proceso de asignar un identificador a un espacio de memoria se le conoce como “**definir una variable**”
- Se utiliza para guardar datos que, posteriormente, puedan ser utilizados por nuestro algoritmo/ programa

Memoria

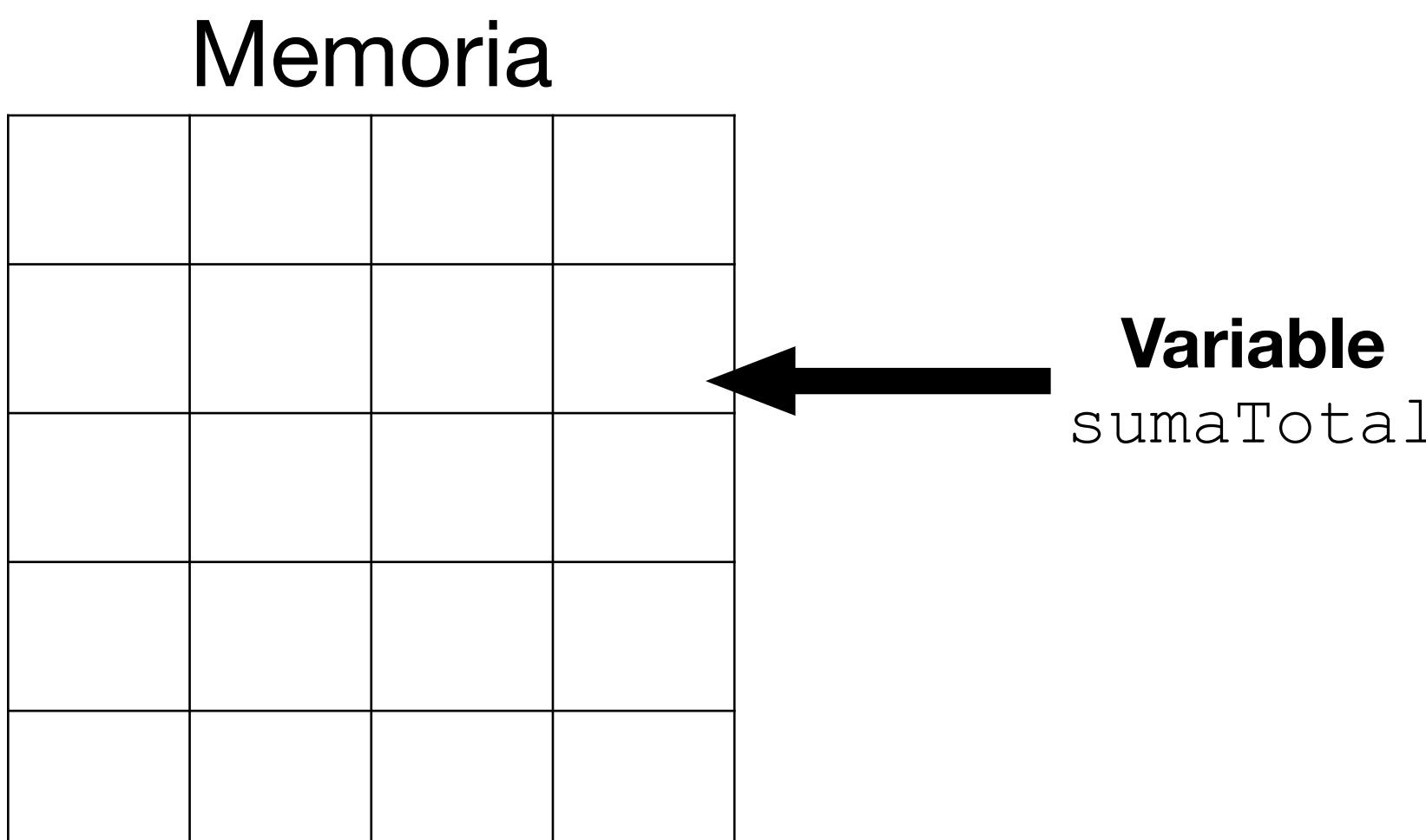
Variable
sumaTotal

1. Repasando conceptos anteriores (i)

Variables

¿Qué es una variable?

- Es un espacio reservado en memoria que tiene asignado un identificador (ejemplo, `sumaTotal`)
- Al proceso de asignar un identificador a un espacio de memoria se le conoce como “**definir una variable**”
- Se utiliza para guardar datos que, posteriormente, puedan ser utilizados por nuestro algoritmo/programa

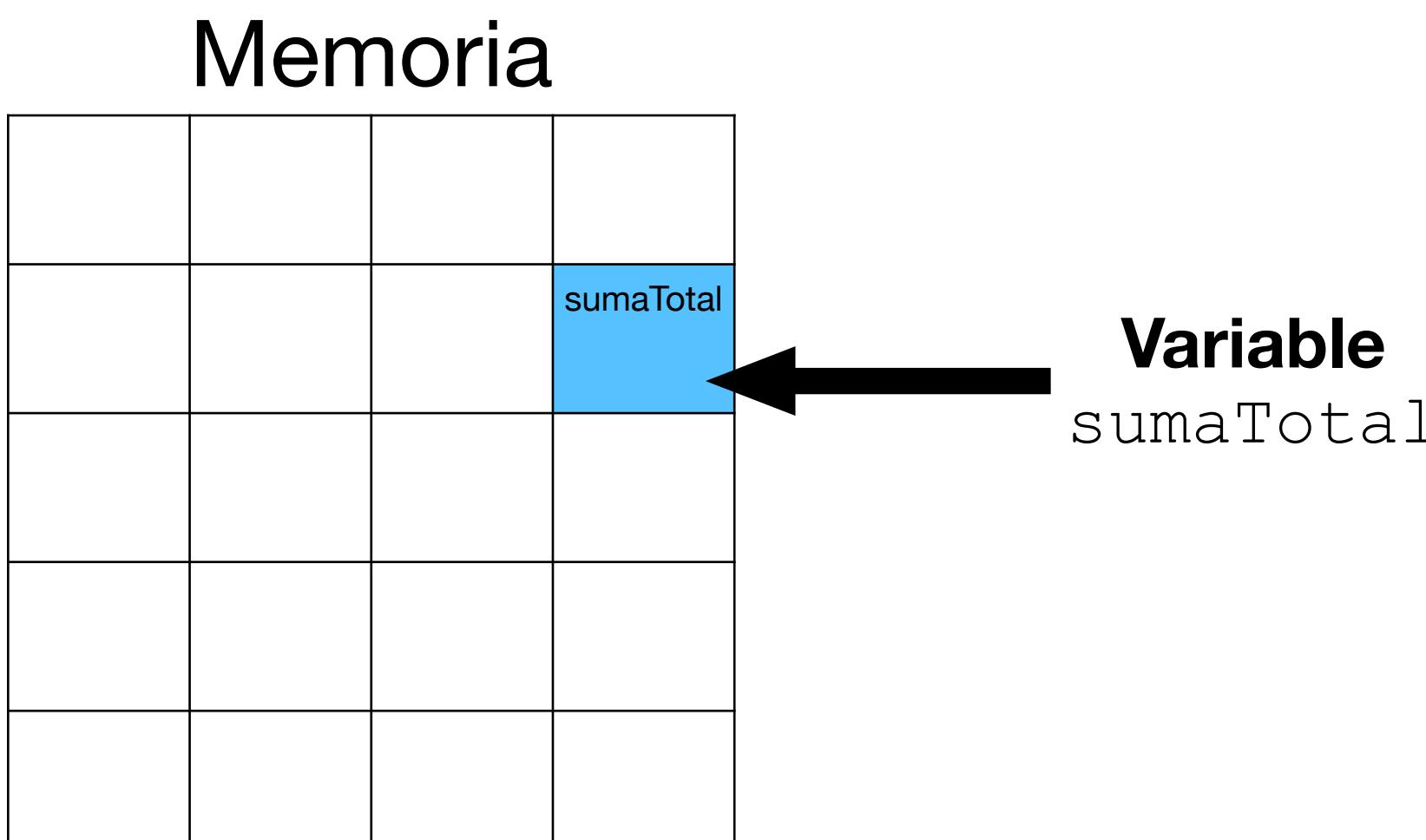


1. Repasando conceptos anteriores (i)

Variables

¿Qué es una variable?

- Es un espacio reservado en memoria que tiene asignado un identificador (ejemplo, `sumaTotal`)
- Al proceso de asignar un identificador a un espacio de memoria se le conoce como “**definir una variable**”
- Se utiliza para guardar datos que, posteriormente, puedan ser utilizados por nuestro algoritmo/programa



1. Repasando conceptos anteriores (i)

Variables

¿Qué valores puede guardar una variable?

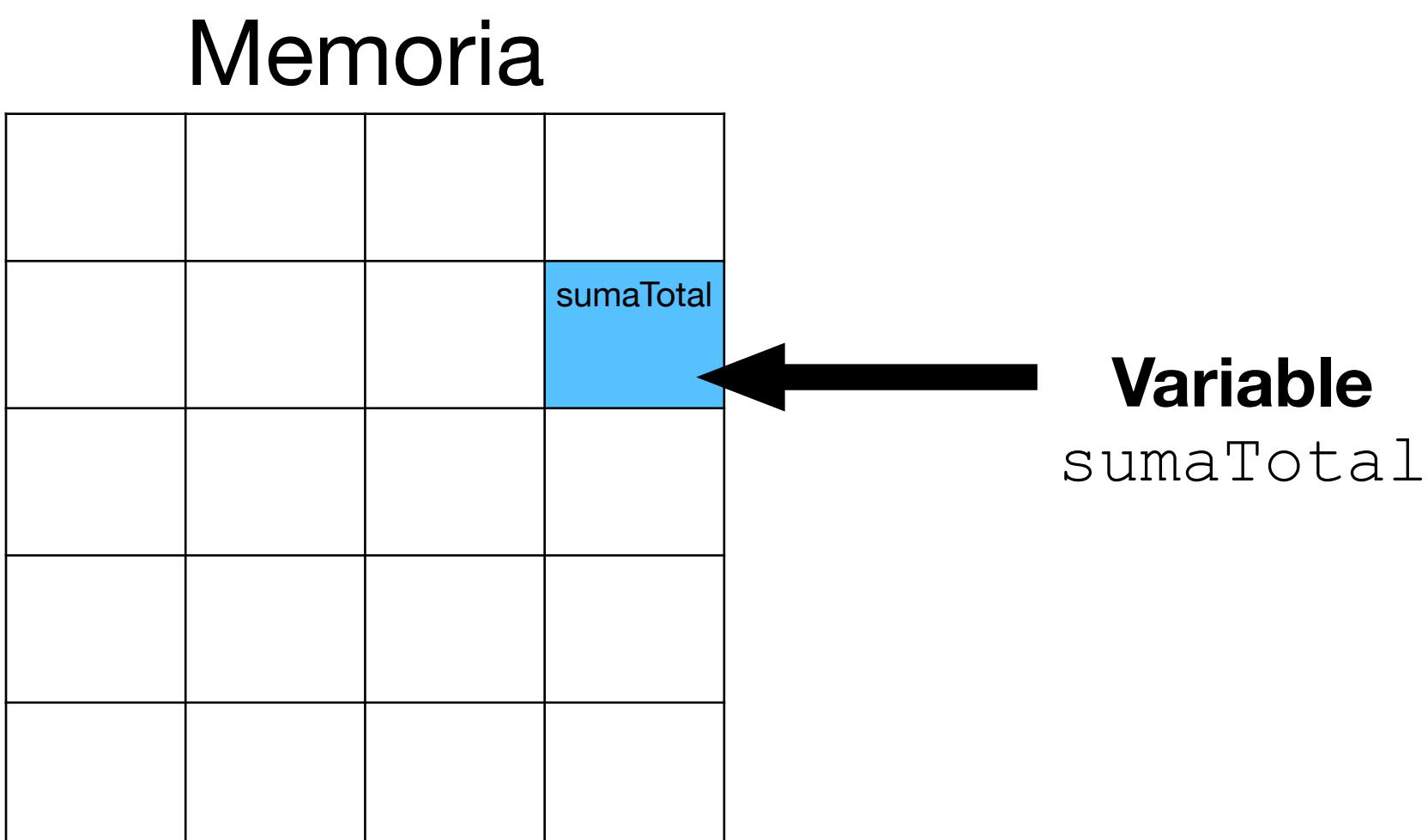
Depende del tipo de dato que le asignamos al definirla: **entero** y **real** (numérico), **texto**, **lógica** (V/F)

1. Repasando conceptos anteriores (i)

Variables

¿Qué valores puede guardar una variable?

Depende del tipo de dato que le asignamos al definirla: **entero** y **real** (numérico), **texto**, **lógica** (V/F)

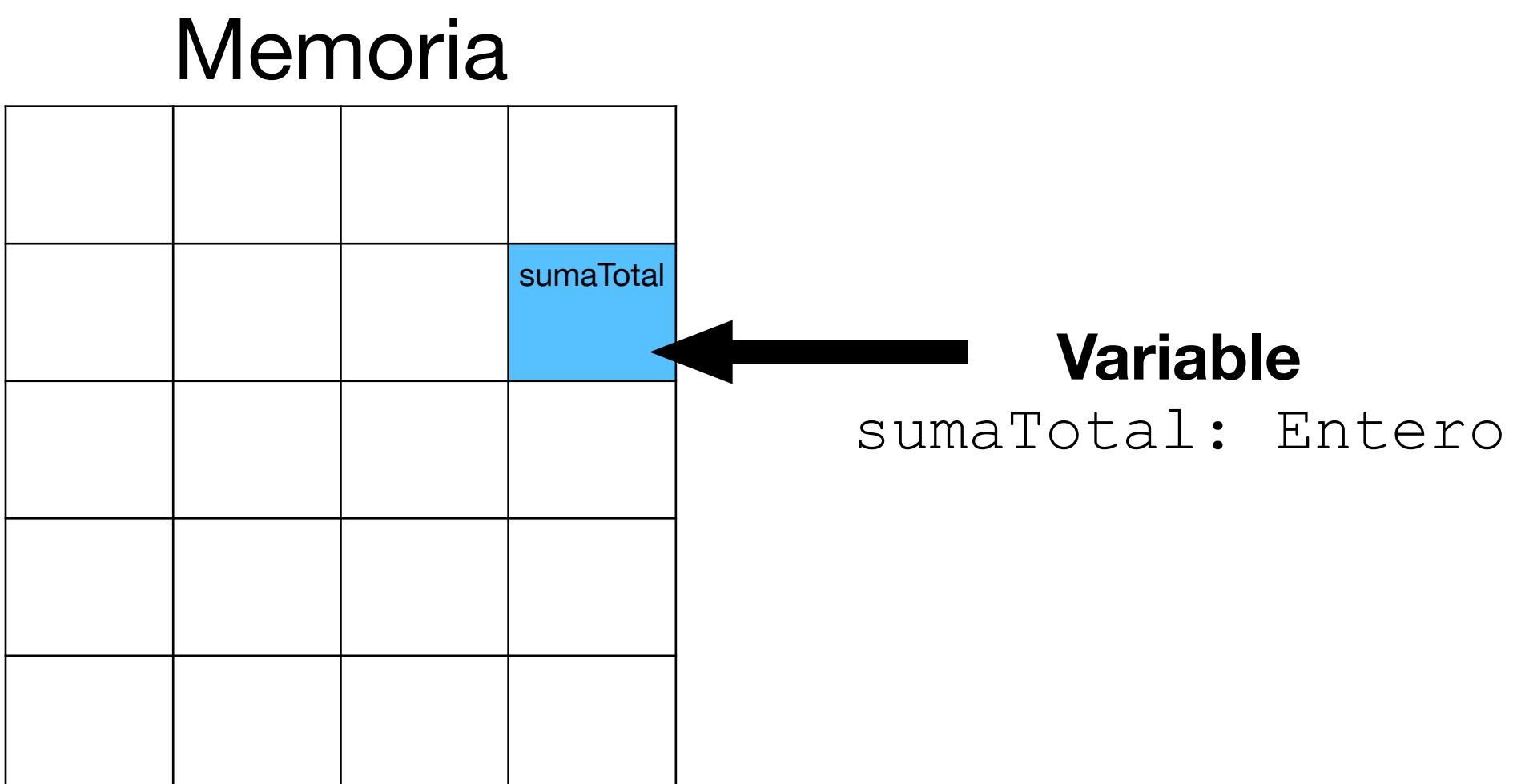


1. Repasando conceptos anteriores (i)

Variables

¿Qué valores puede guardar una variable?

Depende del tipo de dato que le asignamos al definirla: **entero** y **real** (numérico), **texto**, **lógica** (V/F)

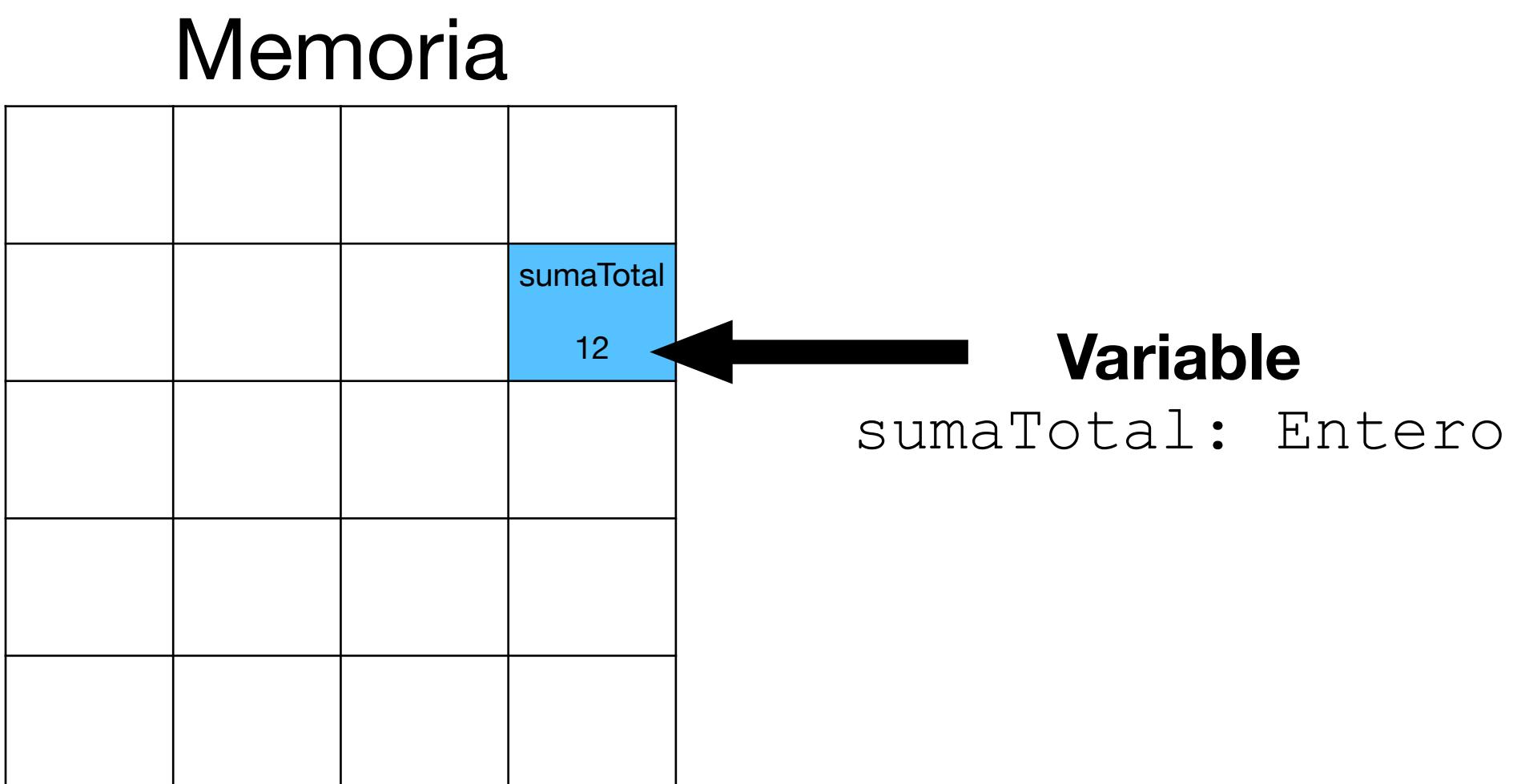


1. Repasando conceptos anteriores (i)

Variables

¿Qué valores puede guardar una variable?

Depende del tipo de dato que le asignamos al definirla: **entero** y **real** (numérico), **texto**, **lógica** (V/F)

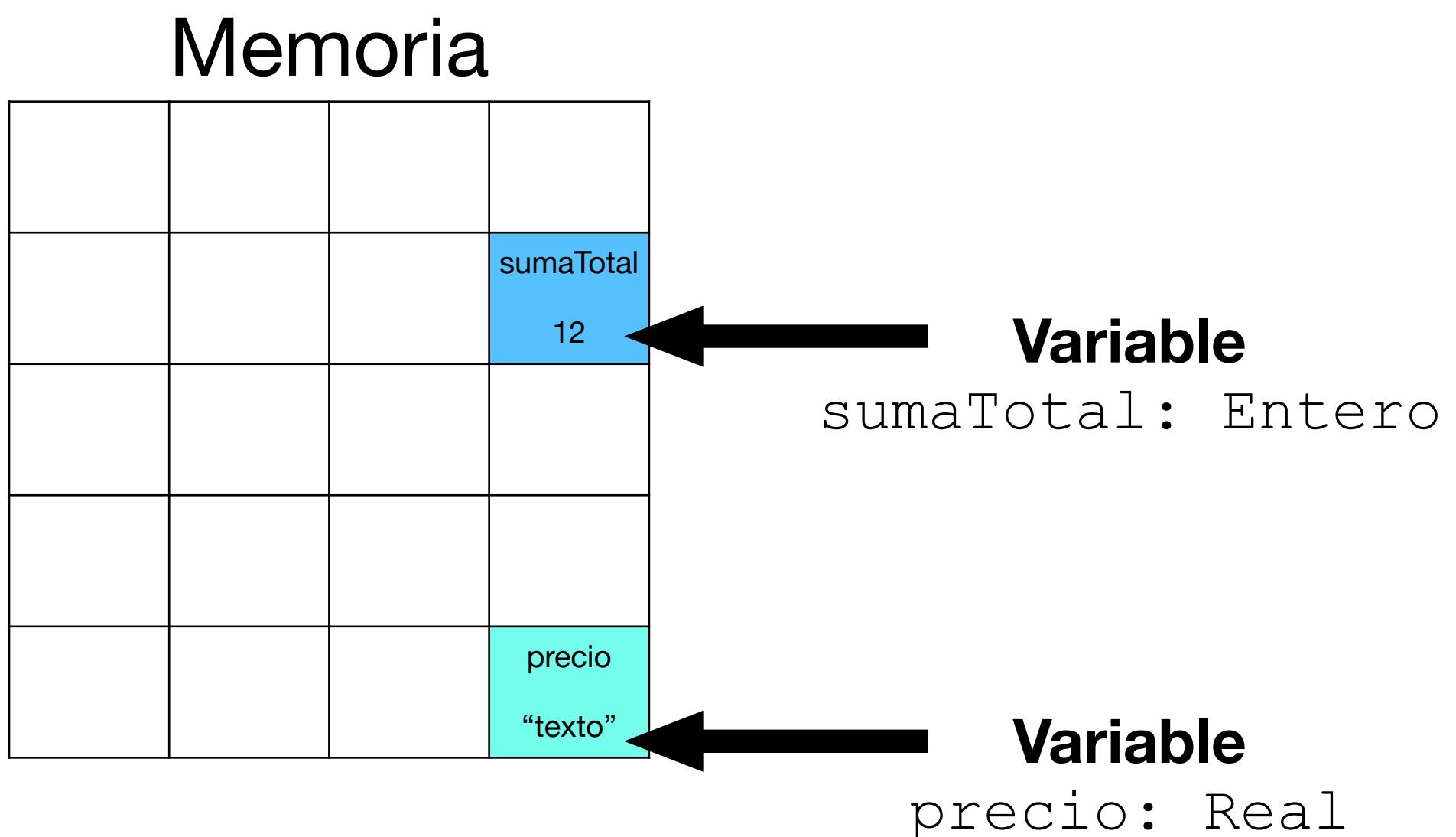


1. Repasando conceptos anteriores (i)

Variables

¿Qué valores puede guardar una variable?

Depende del tipo de dato que le asignamos al definirla: **entero** y **real** (numérico), **texto**, **lógica** (V/F)

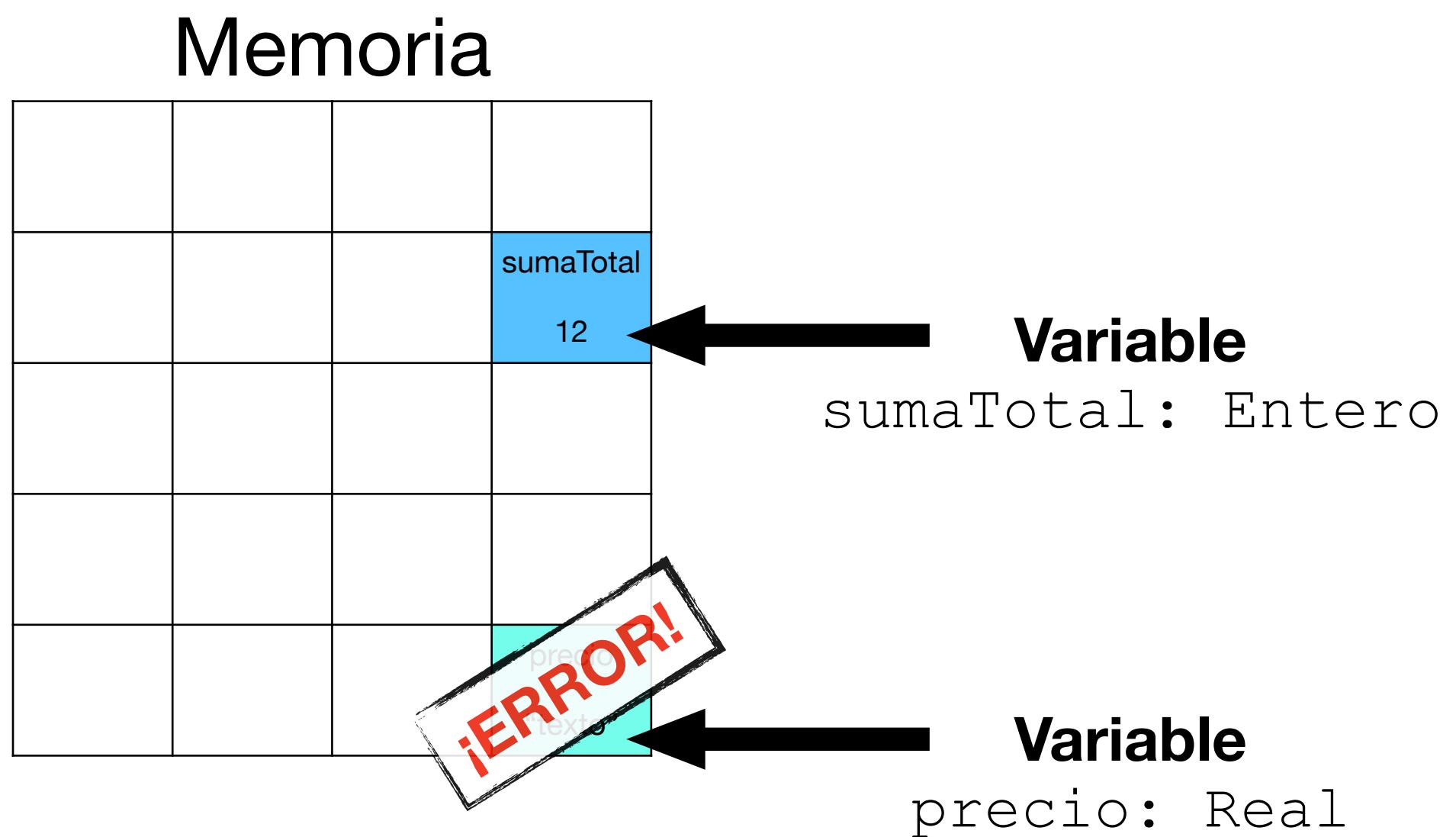


1. Repasando conceptos anteriores (i)

Variables

¿Qué valores puede guardar una variable?

Depende del tipo de dato que le asignamos al definirla: **entero** y **real** (numérico), **texto**, **lógica** (V/F)



1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **define** una variable en *PseInt*?

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **define** una variable en *PseInt*?

```
Definir sumaTotal Como Entero
```

```
Definir precio Como Real
```

```
Definir nota Como Texto
```

```
Definir terminado Como Logica
```

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **define** una variable en *PseInt*?

```
Definir sumaTotal Como Entero  
Definir precio Como Real  
Definir nota Como Texto  
Definir terminado Como Logica
```

Memoria			
sumaTotal			
	precio		terminado
	nota		

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **inicializa** una variable en *PseInt*?

```
Definir sumaTotal Como Entero
```

```
Definir precio Como Real
```

```
Definir nota Como Texto
```

```
Definir terminado Como Logica
```

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **inicializa** una variable en *PseInt*?

```
Definir sumaTotal Como Entero
```

```
Definir precio Como Real
```

```
Definir nota Como Texto
```

```
Definir terminado Como Logica
```

```
sumaTotal = 12
```

```
precio = 20.5
```

```
nota = "Hola"
```

```
terminado = Falso
```

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **inicializa** una variable en *PseInt*?

```
Definir sumaTotal Como Entero  
Definir precio Como Real  
Definir nota Como Texto  
Definir terminado Como Logica
```

```
sumaTotal = 12  
precio = 20.5  
nota = "Hola"  
terminado = Falso
```

Memoria			
sumaTotal			
12			
	precio		terminado
	20.5		Falso
	nota		
	"Hola"		

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **inicializa** una variable en *PseInt*?

```
Definir sumaTotal Como Entero  
Definir precio Como Real  
Definir nota Como Texto  
Definir terminado Como Logica
```

```
sumaTotal = 12  
precio = 20.5  
nota = "Hola"  
terminado = Falso
```

Memoria

sumaTotal			
12			
	precio		terminado
	20.5		Falso
	nota		
	"Hola"		

RECORDATORIO!!

Para **inicializar** una variable tiene que haber sido **previamente definida**

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **inicializa** una variable en *PseInt*?

```
Definir sumaTotal Como Entero  
Definir precio Como Real  
Definir nota Como Texto  
Definir terminado Como Logica
```

```
sumaTotal = 12  
precio = 20.5  
nota = "Hola"  
terminado = Falso
```

Memoria			
sumaTotal			
12			
	precio		terminado
	20.5		Falso
	nota		
	"Hola"		

RECORDATORIO!!

Para **inicializar** una variable tiene que haber sido **previamente definida**

RECORDATORIO 2!!

La inicialización no siempre es necesaria pero nos evitará errores

BUENAS PRÁCTICAS

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **modifica** una variable en *PseInt*?

```
sumaTotal = 12  
  
precio = 20.5  
  
nota = "Hola"  
  
terminado = Falso
```

Memoria			
sumaTotal			
12			
	precio		terminado
	20.5		Falso
	nota		
	"Hola"		

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **modifica** una variable en *PseInt*?

```
sumaTotal = 12  
  
precio = 20.5  
  
nota = "Hola"  
  
terminado = Falso
```

```
sumaTotal = sumaTotal + 4  
  
precio = 3.4 + 4.6  
  
nota = "Adios"  
  
terminado = Verdadero
```

Memoria			
sumaTotal			
12			
	precio		terminado
	20.5		Falso
	nota		
	"Hola"		

1. Repasando conceptos anteriores (ii)

Variables

¿Cómo se **modifica** una variable en *PseInt*?

```
sumaTotal = 12  
  
precio = 20.5  
  
nota = "Hola"  
  
terminado = Falso
```

```
sumaTotal = sumaTotal + 4  
  
precio = 3.4 + 4.6  
  
nota = "Adios"  
  
terminado = Verdadero
```

Memoria			
sumaTotal			
16			
	precio		terminado
	8.0		Verdadero
	nota		
	"Adios"		

1. Repasando conceptos anteriores (ii)

Lectura - Escritura

¿Qué hace la instrucción de **lectura**?

Lee un dato introducido por **pantalla/consola/terminal** y lo almacena en una variable previamente definida

1. Repasando conceptos anteriores (ii)

Lectura - Escritura

¿Qué hace la instrucción de **lectura**?

Lee un dato introducido por **pantalla/consola/terminal** y lo almacena en una variable previamente definida

```
//Definición e inicialización  
//de variable precio  
Definir precio Como Real  
precio = 0  
  
Leer precio
```

1. Repasando conceptos anteriores (ii)

Lectura - Escritura

¿Qué hace la instrucción de **lectura**?

Lee un dato introducido por **pantalla/consola/terminal** y lo almacena en una variable previamente definida

```
//Definición e inicialización  
//de variable precio  
Definir precio Como Real  
precio = 0  
  
Leer precio
```

Memoria

precio			
0			

1. Repasando conceptos anteriores (ii)

Lectura - Escritura

¿Qué hace la instrucción de **lectura**?

Lee un dato introducido por **pantalla/consola/terminal** y lo almacena en una variable previamente definida

```
//Definición e inicialización  
//de variable precio  
Definir precio Como Real  
precio = 0  
  
Leer precio
```

```
*** Ejecución Iniciada. ***  
> 20.3
```

Memoria

precio			
0			

1. Repasando conceptos anteriores (ii)

Lectura - Escritura

¿Qué hace la instrucción de **lectura**?

Lee un dato introducido por **pantalla/consola/terminal** y lo almacena en una variable previamente definida

```
//Definición e inicialización  
//de variable precio  
Definir precio Como Real  
precio = 0  
  
Leer precio
```

```
*** Ejecución Iniciada. ***  
> 20.3
```

Memoria

precio	20.3		

1. Repasando conceptos anteriores (ii)

Lectura - Escritura

¿Qué hace la instrucción de **escritura**?

Escribe por pantalla/consola/terminal el valor de una o varias variables, separadas por comas

1. Repasando conceptos anteriores (ii)

Lectura - Escritura

¿Qué hace la instrucción de **escritura**?

Escribe por **pantalla/consola/terminal** el valor de una o varias variables, separadas por comas

```
//Definición e inicialización  
//de variable precio  
Definir precio Como Real  
precio = 20.3  
  
Escribir "El precio es: ", precio
```

Memoria

precio			
20.3			

1. Repasando conceptos anteriores (ii)

Lectura - Escritura

¿Qué hace la instrucción de **escritura**?

Escribe por **pantalla/consola/terminal** el valor de una o varias variables, separadas por comas

```
//Definición e inicialización  
//de variable precio  
Definir precio Como Real  
precio = 20.3  
  
Escribir "El precio es: ", precio
```

```
*** Ejecución Iniciada. ***  
El precio es: 20.3  
*** Ejecución Finalizada. ***
```

Memoria

precio	20.3		

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES

¿Para qué sirve la estructura condicional SI-ENTONCES?

Permite ejecutar un código diferente en función del resultado de una **expresión lógica** (V/F)

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES

¿Para qué sirve la estructura condicional SI-ENTONCES?

Permite ejecutar un código diferente en función del resultado de una **expresión lógica** (V/F)

¿Cuál es su sintaxis?

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES

¿Para qué sirve la estructura condicional SI-ENTONCES?

Permite ejecutar un código diferente en función del resultado de una **expresión lógica** (V/F)

¿Cuál es su sintaxis?

```
Si expresion_logica Entonces  
|   ejecucion_codigo_1  
SiNo  
|   ejecucion_codigo_2  
FinSi
```

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES

¿Para qué sirve la estructura condicional SI-ENTONCES?

Permite ejecutar un código diferente en función del resultado de una **expresión lógica** (V/F)

¿Cuál es su sintaxis?

```
Si expresion_logica Entonces
|   ejecucion_codigo_1
SiNo
|   ejecucion_codigo_2
FinSi
```

La parte **SiNo** es **OPCIONAL**

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES

¿Para qué sirve la estructura condicional SI-ENTONCES?

Permite ejecutar un código diferente en función del resultado de una **expresión lógica** (V/F)

¿Cuál es su sintaxis?

¿Qué expresiones lógicas podemos utilizar?

```
Si expresion_logica Entonces  
|   ejecucion_codigo_1  
SiNo  
|   ejecucion_codigo_2  
FinSi
```

La parte **SiNo** es **OPCIONAL**

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES

¿Para qué sirve la estructura condicional SI-ENTONCES?

Permite ejecutar un código diferente en función del resultado de una **expresión lógica** (V/F)

¿Cuál es su sintaxis?

```
Si expresion_logica Entonces  
| ejecucion_codigo_1  
SiNo  
| ejecucion_codigo_2  
FinSi
```

¿Qué expresiones lógicas podemos utilizar?

```
precio = 10  
precio ≤ 10  
precio ≠ 10  
terminado = Verdadero  
nota = "Hola"
```

La parte **SiNo** es **OPCIONAL**

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES

Ejemplo

```
//Definición e inicialización
//de variable precio
Definir precio Como Entero
precio = 0
//Leemos el nuevo dato y se
//almacena en "precio"
Leer precio
//Condicional SI-ENTONCES
Si precio < 20 Entonces
    Escribir "El precio es menor a 20. Es: ", precio
SiNo
    Escribir "El precio es mayor o igual a 20. Es: ", precio
FinSi
```

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES

Ejemplo

```
//Definición e inicialización
//de variable precio
Definir precio Como Entero
precio = 0
//Leemos el nuevo dato y se
//almacena en "precio"
Leer precio
//Condicional SI-ENTONCES
Si precio < 20 Entonces
    Escribir "El precio es menor a 20. Es: ", precio
SiNo
    Escribir "El precio es mayor o igual a 20. Es: ", precio
FinSi
```

```
*** Ejecución Iniciada. ***
> 13
El precio es menor a 20. Es: 13
*** Ejecución Finalizada. ***
```

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES anidada

¿Para qué sirve la estructura condicional **SI-ENTONCES anidada?**

Permite ejecutar un código diferente en función del resultado de **varias expresiones lógicas** (existen más de dos posibilidades)

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES anidada

¿Para qué sirve la estructura condicional **SI-ENTONCES anidada?**

Permite ejecutar un código diferente en función del resultado de **varias expresiones lógicas** (existen más de dos posibilidades)

¿Cuál es su sintaxis?

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES anidada

¿Para qué sirve la estructura condicional **SI-ENTONCES** anidada?

Permite ejecutar un código diferente en función del resultado de **varias expresiones lógicas** (existen **más de dos posibilidades**)

¿Cuál es su sintaxis?

```
Si expresion_logica_1 Entonces
    Si expresion_logica_2 Entonces
        ejecucion_codigo_1
    SiNo
        ejecucion_codigo_2
    FinSi
    SiNo
        Si expresion_logica_3 Entonces
            ejecucion_codigo_3
        SiNo
            ejecucion_codigo_4
        FinSi
    FinSi
```

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES anidada

Ejemplo

```
//Definición e inicialización
Definir hasPagado Como Logica
Definir precio Como Entero
hasPagado = Falso
precio = 0
//Leemos los datos
Leer hasPagado
Leer precio
//Condicional SI-ENTONCES anidado
Si hasPagado = Falso Entonces
    Escribir "No has pagado aún"
    Si precio ≤ 20 Entonces
        Escribir "Tienes que pagar menos de 20 euros"
    SiNo
        Escribir "Tienes que pagar más de 20 euros"
    FinSi
SiNo
    Escribir "Ya has pagado"
FinSi
```

1. Repasando conceptos anteriores (iii)

Condicional SI-ENTONCES anidada

Ejemplo

```
//Definición e inicialización
Definir hasPagado Como Logica
Definir precio Como Entero
hasPagado = Falso
precio = 0
//Leemos los datos
Leer hasPagado
Leer precio
//Condicional SI-ENTONCES anidado
Si hasPagado = Falso Entonces
    Escribir "No has pagado aún"
    Si precio ≤ 20 Entonces
        Escribir "Tienes que pagar menos de 20 euros"
    SiNo
        Escribir "Tienes que pagar más de 20 euros"
    FinSi
SiNo
    Escribir "Ya has pagado"
FinSi
```

```
*** Ejecución Iniciada. ***
> falso
> 14
No has pagado aún
Tienes que pagar menos de 20 euros
*** Ejecución Finalizada. ***
```

1. Repasando conceptos anteriores (iv)

Esqueleto básico algoritmo *PselInt*

1. Repasando conceptos anteriores (iv)

Esqueleto básico algoritmo *PselInt*

Algoritmo ejemplo

```
//1. Definición de variables  
//2. Inicialización de variables  
//3. Operaciones de lectura  
//4. Operaciones lógico-matemáticas  
//4.1. Estructuras condicionales, selectivas, bucles...  
//5. Operaciones de escritura
```

FinAlgoritmo

1. Repasando conceptos anteriores (iv)

Esqueleto básico algoritmo *PselInt*

Algoritmo ejemplo

//1. Definición de variables

//2. Inicialización de variables

//3. Operaciones de lectura

//4. Operaciones lógico-matemáticas

//4.1. Estructuras condicionales, selectivas, bucles...

//5. Operaciones de escritura

FinAlgoritmo

Algoritmo ejemplo

//1. Definición de variables

Definir hasPagado como Logico

Definir precio Como Entero

Definir mensajeHasPagado, mensajePrecio Como Texto

//2. Inicialización de variables

hasPagado = Falso

precio = 0

mensajeHasPagado = ""

mensajePrecio = ""

//3. Operaciones de lectura

Leer hasPagado

Leer precio

//4. Operaciones lógico-matemáticas

//4.1. Estructuras condicionales, selectivas, bucles...

Si hasPagado = Falso Entonces

 mensajeHasPagado = "No has pagado aún"

 Si precio ≤ 20 Entonces

 mensajePrecio = "Tienes que pagar menos de 20 euros"

 SiNo

 mensajePrecio = "Tienes que pagar más de 20 euros"

 FinSi

 SiNo

 mensajeHasPagado = "Ya has pagado"

 FinSi

//5. Operaciones de escritura

Escribir mensajeHasPagado, ".", ", ", mensajePrecio

FinAlgoritmo

1. Repasando conceptos anteriores (iv)

Esqueleto básico algoritmo *PselInt*

Algoritmo ejemplo

//1. Definición de variables

//2. Inicialización de variables

//3. Operaciones de lectura

//4. Operaciones lógico-matemáticas

//4.1. Estructuras condicionales, selectivas, bucles...

//5. Operaciones de escritura

FinAlgoritmo

*** Ejecución Iniciada. ***

> falso

> 14

No has pagado aún. Tienes que pagar menos de 20 euros

*** Ejecución Finalizada. ***

Algoritmo ejemplo

//1. Definición de variables

Definir hasPagado como Logico

Definir precio Como Entero

Definir mensajeHasPagado, mensajePrecio Como Texto

//2. Inicialización de variables

hasPagado = Falso

precio = 0

mensajeHasPagado = ""

mensajePrecio = ""

//3. Operaciones de lectura

Leer hasPagado

Leer precio

//4. Operaciones lógico-matemáticas

//4.1. Estructuras condicionales, selectivas, bucles...

Si hasPagado = Falso Entonces

 mensajeHasPagado = "No has pagado aún"

 Si precio ≤ 20 Entonces

 mensajePrecio = "Tienes que pagar menos de 20 euros"

 SiNo

 mensajePrecio = "Tienes que pagar más de 20 euros"

 FinSi

 SiNo

 mensajeHasPagado = "Ya has pagado"

 FinSi

//5. Operaciones de escritura

Escribir mensajeHasPagado, ".", ", mensajePrecio

FinAlgoritmo

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

¿Para qué se utilizan los operadores algebraicos?

Permiten realizar ciertas operaciones matemáticas

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

¿Para qué se utilizan los operadores algebraicos?

Permiten realizar ciertas operaciones matemáticas

suma (+), resta (-), multiplicación (*), división (*)

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

¿Para qué se utilizan los **operadores algebraicos**?

Permiten realizar ciertas operaciones matemáticas

suma (+), resta (-), multiplicación (*), división (/)

```
precio = 2 + 4 //precio = 6  
precio = 6 - 1 //precio = 5  
precio = 2 * 3 //precio = 6  
precio = 4 / 2 //precio = 2
```

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia

Recordatorio

The diagram shows the expression $3^2 = 9$. A yellow arrow points from the word "exponente" to the superscript "2". A green arrow points from the word "base" to the number "3". To the right of the equals sign, a blue arrow points from the word "potencia" to the result "9".

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia

Recordatorio

The diagram shows the expression $3^2 = 9$. A yellow arrow points from the number 2 to the equals sign, labeled "exponente". A green arrow points from the number 3 to the equals sign, labeled "base". To the right of the equals sign, the word "potencia" is written in blue.

$$3 * 3 = 9$$

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia

Recordatorio

A diagram showing the components of a power expression $3^2 = 9$. A yellow arrow points from the number 2 to the equals sign, labeled "exponente". A green arrow points from the equals sign to the number 9, labeled "base". The entire expression is labeled "potencia" at the bottom right.

$$3^2 = 9$$

En *PseInt*...

$$3 * 3 = 9$$

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

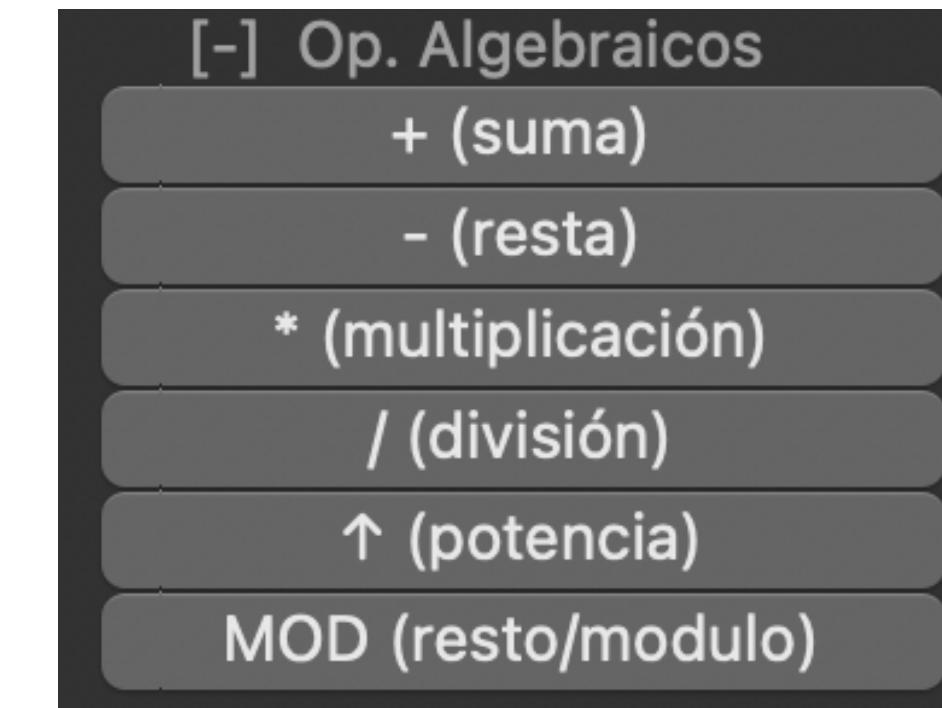
1. Potencia

Recordatorio

A diagram illustrating the components of a power expression. It shows $3^2 = 9$. An orange arrow points from the number 3 to the text "base". An orange arrow points from the number 2 to the text "exponente". A blue arrow points from the equals sign to the text "potencia".

$$3 * 3 = 9$$

En *PseInt*...



2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia

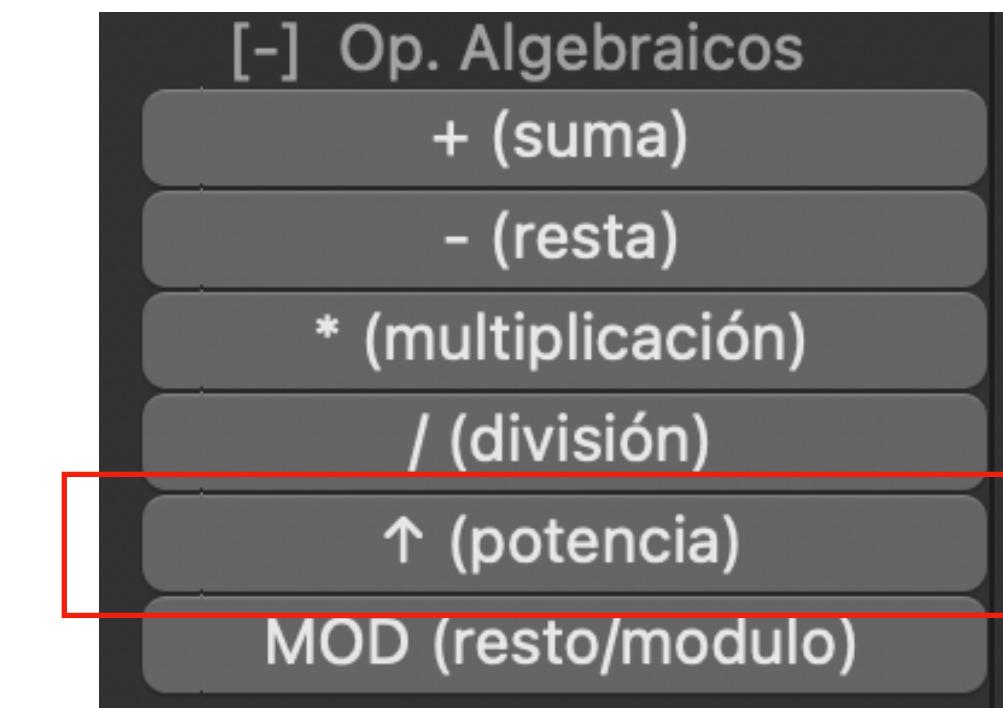
Recordatorio

Diagram illustrating the components of a power expression:

- base**: The number being multiplied (3).
- exponent**: The number of times the base is multiplied by itself (2).
- potencia**: The result of the multiplication (9).

$$3 * 3 = 9$$

En *PseInt*...



2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia

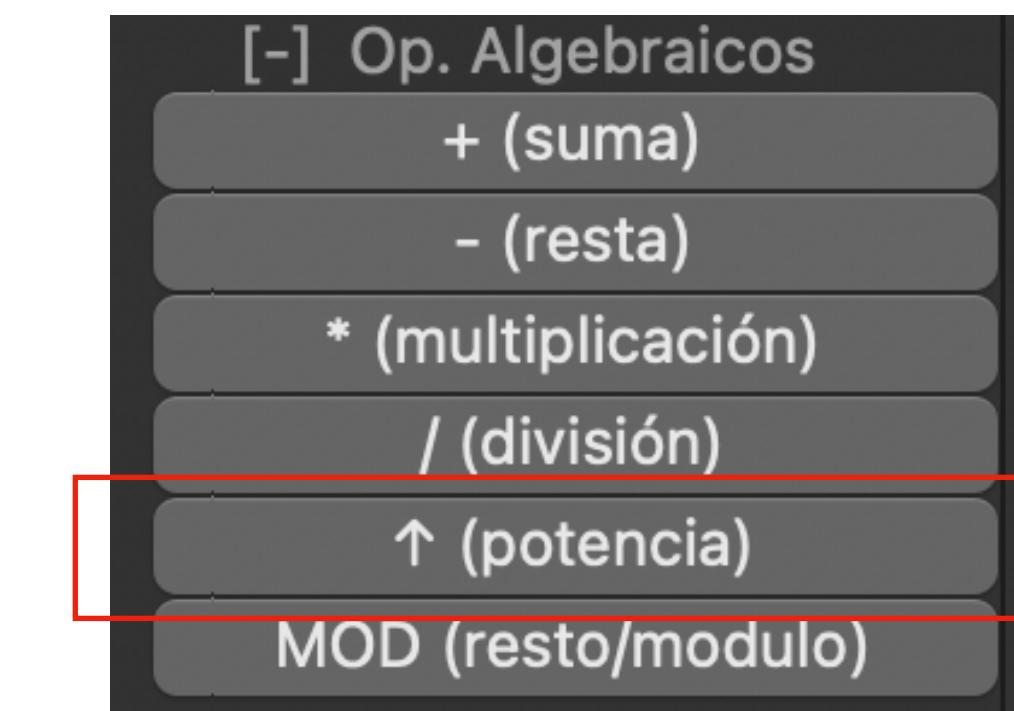
Recordatorio

$$3^2 = 9$$

exponente
↑
base
↓

$$3 * 3 = 9$$

En *PseInt*...



```
precio = 3 ↑ 2
```

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia

Recordatorio

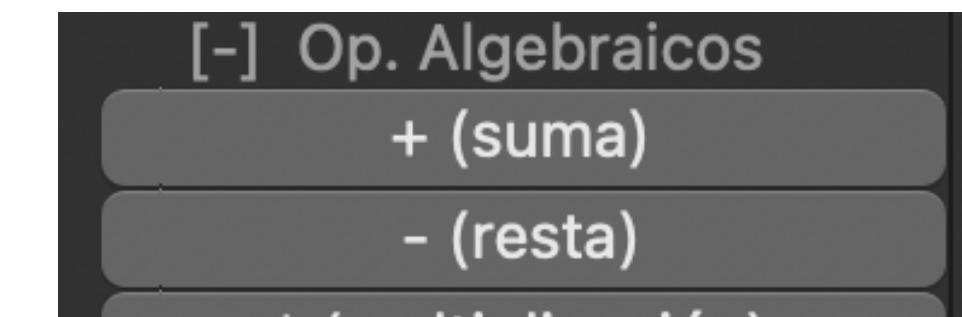
$$3^2 = 9$$

exponente
↑
base ↓

potencia

← 3 * 3 = 9

En *PseInt*...



precio = 3 ↑ 2



2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia (ejemplo)

A diagram showing the components of the power expression $3^2 = 9$. A yellow arrow points from the word "exponente" to the superscript "2". A green arrow points from the word "base" to the number "3". A blue arrow points from the expression to the word "potencia".

$$3^2 = 9 \rightarrow \text{potencia}$$

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia (ejemplo)

$$3^2 = 9$$

Diagram illustrating the components of the power operation:

- exponente** (Exponent) points to the superscript 2.
- base** (Base) points to the number 3.
- potencia** (Power) points to the result 9.

```
//Definición e inicialización
Definir base, exponente, potencia Como Entero
base = 0
exponente = 0
potencia = 0
//Leemos los datos
Escribir "Introduce la base de la potencia"
Leer base
Escribir "Introduce el exponente de la potencia"
Leer exponente
//Calculamos la potencia
potencia = base ↑ exponente
//Escribimos el resultado de la potencia
Escribir "El resultado de la potencia es: ", potencia
```

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

1. Potencia (ejemplo)

$$3^2 = 9$$

Diagram illustrating the components of the power operation:

- exponente** (Exponent) points to the superscript 2.
- base** (Base) points to the number 3.
- potencia** (Power) points to the result 9.

```
//Definición e inicialización
Definir base, exponente, potencia Como Entero
base = 0
exponente = 0
potencia = 0
//Leemos los datos
Escribir "Introduce la base de la potencia"
Leer base
Escribir "Introduce el exponente de la potencia"
Leer exponente
//Calculamos la potencia
potencia = base ↑ exponente
//Escribimos el resultado de la potencia
Escribir "El resultado de la potencia es: ", potencia
```

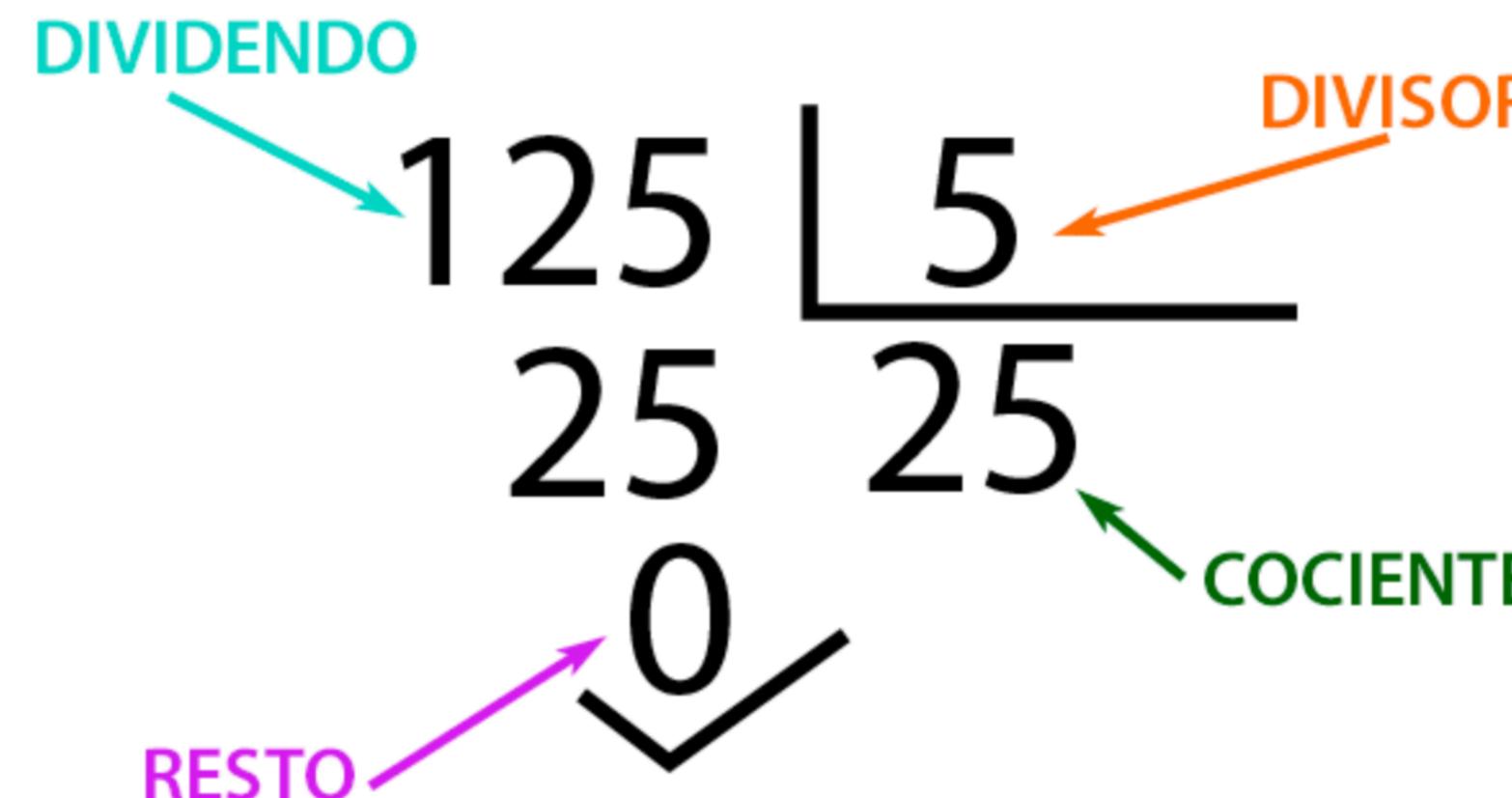
```
*** Ejecución Iniciada. ***
Introduce la base de la potencia
> 3
Introduce el exponente de la potencia
> 2
El resultado de la potencia es: 9
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

2. MOD (o resto)

Se utiliza en operaciones entre dos operandos, para obtener el módulo (o resto) del primero entre el segundo



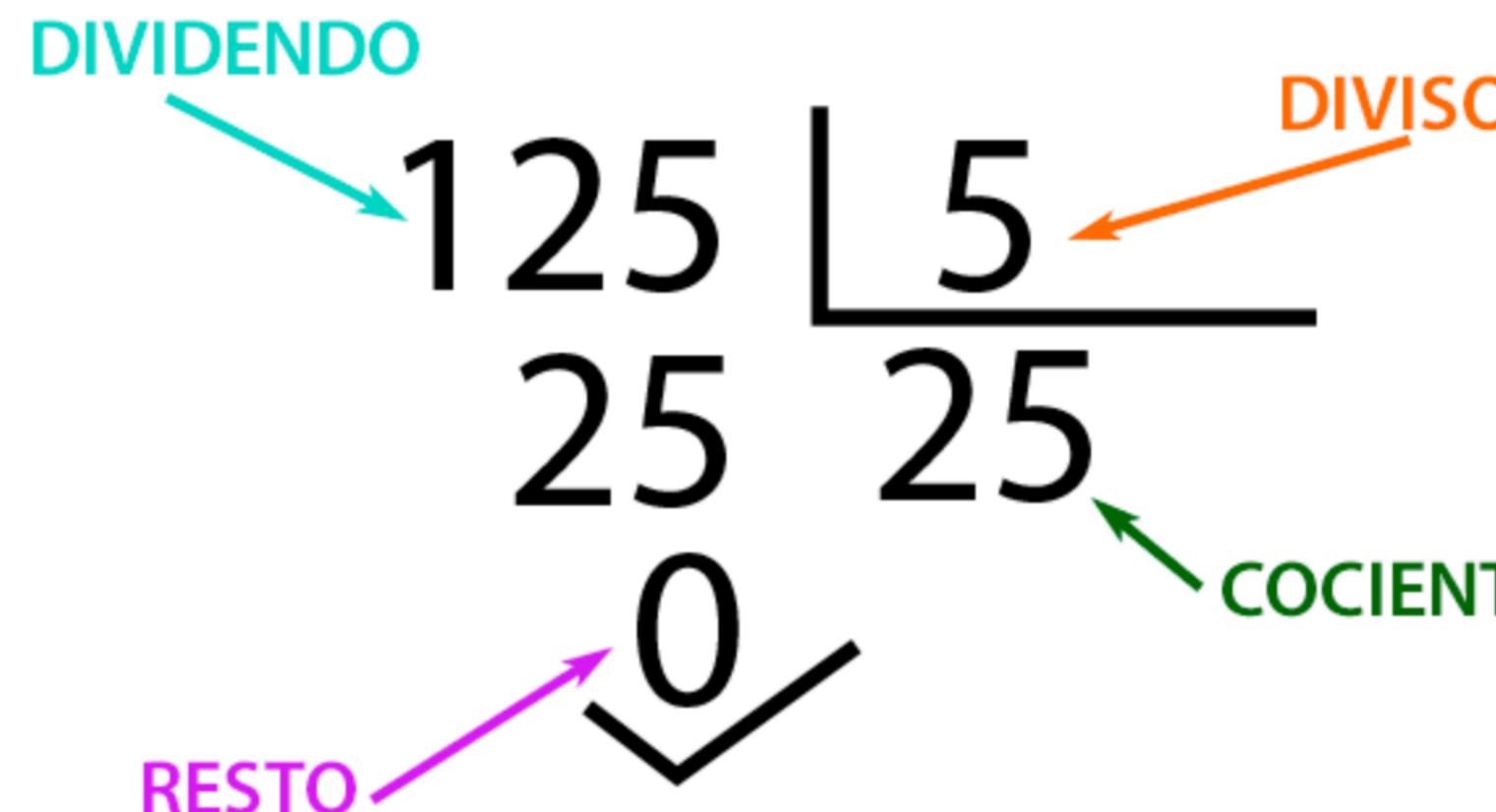
A diagram illustrating a division operation. On the left, the number 125 is labeled "DIVIDENDO" with a cyan arrow. In the center, the number 5 is labeled "DIVISOR" with an orange arrow. To the right of the divisor, the number 25 is labeled "COCIENTE" with a green arrow. Below the dividend, the number 0 is labeled "RESTO" with a magenta arrow. The entire setup is enclosed in a horizontal line, representing a long division problem where 5 goes into 125 exactly 25 times, leaving a remainder of 0.

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

2. MOD (o resto)

Se utiliza en operaciones entre dos operandos, para obtener el módulo (o resto) del primero entre el segundo



The diagram illustrates the components of a division operation:

- DIVIDENDO**: The number being divided, 125.
- DIVISOR**: The number by which the dividend is divided, 5.
- COCIENTE**: The result of the division, 25.
- RESTO**: The remainder left after division, 0.

Below the diagram, the division equation is shown: $125 / 5 = 25$.

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

2. MOD (o resto)

Se utiliza en operaciones entre dos operandos, para obtener el módulo (o resto) del primero entre el segundo

$$\begin{array}{r} \text{DIVIDENDO} \\ \text{DIVISOR} \\ \text{RESTO} \\ \text{COCIENTE} \end{array} \begin{array}{r} 125 \\ 25 \\ \hline 0 \end{array} \begin{array}{r} 5 \\ \hline 25 \end{array}$$

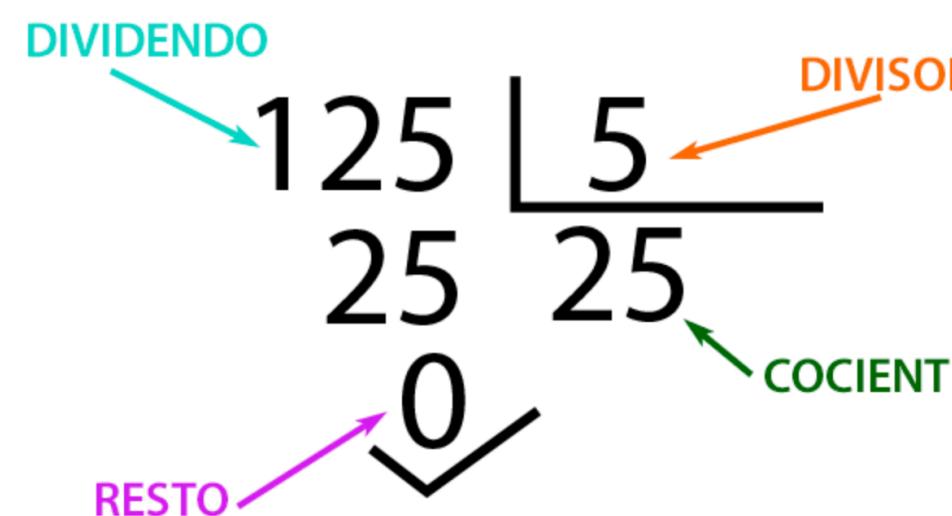
$$125 / 5 = 25$$

$$125 \text{ MOD } 5 = 0 \rightarrow 125 \% 5 = 0$$

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

2. MOD (ejemplo)

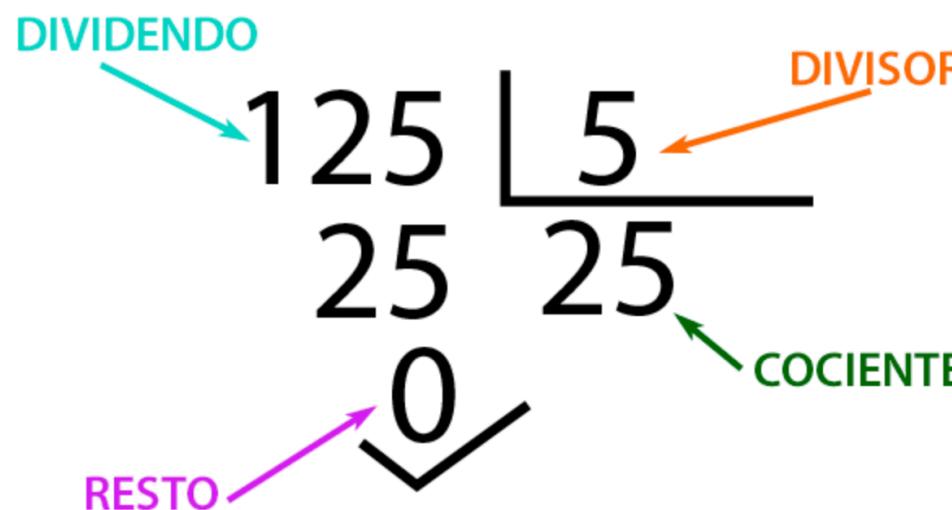


A diagram illustrating a division operation. The dividend is 125, the divisor is 5, the quotient is 25, and the remainder is 0. The diagram shows the division symbol with 125 under the bar and 5 outside. Arrows point from the labels to their respective parts: 'DIVIDENDO' points to 125, 'DIVISOR' points to 5, 'COCIENTE' points to 25, and 'RESTO' points to 0.

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

2. MOD (ejemplo)

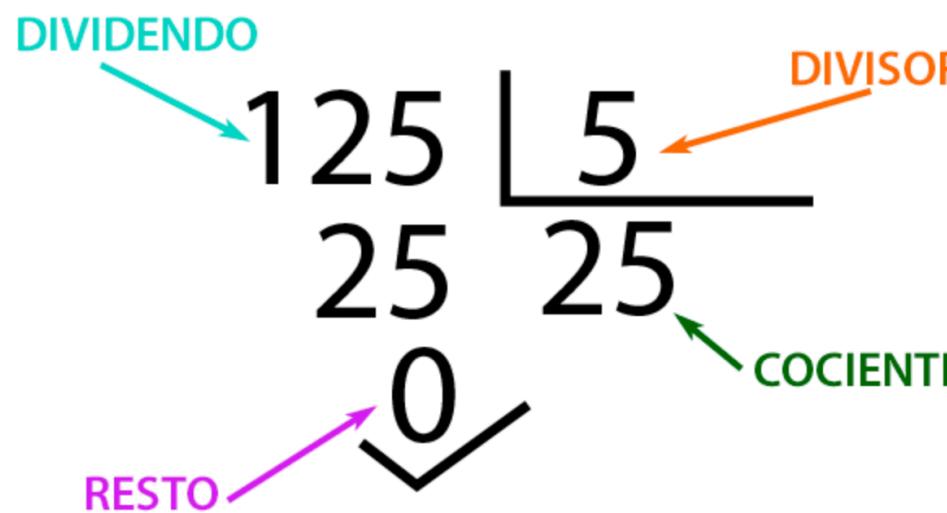


```
//Definición e inicialización
Definir dividendo, divisor, modulo Como Real
dividendo = 0
divisor = 0
modulo = 0
//Leemos los datos
Escribir "Introduce el dividendo"
Leer dividendo
Escribir "Introduce el divisor"
Leer divisor
//Calculamos el módulo (resto)
modulo = dividendo MOD divisor
//Escribimos el resultado del módulo
Escribir "El resultado del módulo es: ", modulo
```

2. Aprendiendo nuevos conceptos (i)

Operadores algebraicos

2. MOD (ejemplo)



```
//Definición e inicialización
Definir dividendo, divisor, modulo Como Real
dividendo = 0
divisor = 0
modulo = 0
//Leemos los datos
Escribir "Introduce el dividendo"
Leer dividendo
Escribir "Introduce el divisor"
Leer divisor
//Calculamos el módulo (resto)
modulo = dividendo MOD divisor
//Escribimos el resultado del módulo
Escribir "El resultado del módulo es: ", modulo
```

```
*** Ejecución Iniciada. ***
Introduce el dividendo
> 5
Introduce el divisor
> 2
El resultado del módulo es: 1
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (ii)

Constantes

2. Aprendiendo nuevos conceptos (ii)

Constantes

¿Qué es una constante?

- Al igual que una variable, es un espacio reservado en memoria que tiene asignado un identificador
- La **diferencia** con las variables es que **el valor de una constante no cambia**
- Un ejemplo es el número ***PI*** (3,14159...)

2. Aprendiendo nuevos conceptos (ii)

Constantes

¿Qué es una constante?

- Al igual que una variable, es un espacio reservado en memoria que tiene asignado un identificador
- La **diferencia** con las variables es que **el valor de una constante no cambia**
- Un ejemplo es el número ***PI*** (3,14159...)

Memoria

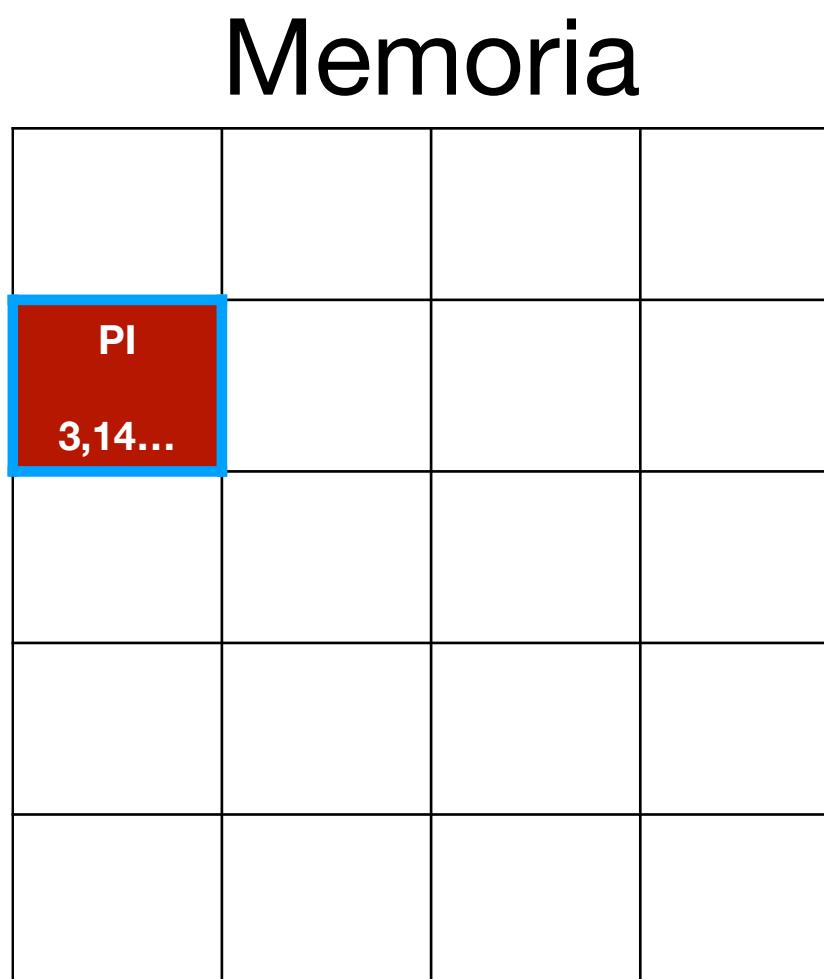
PI			
3,14...			

2. Aprendiendo nuevos conceptos (ii)

Constantes

¿Qué es una constante?

- Al igual que una variable, es un espacio reservado en memoria que tiene asignado un identificador
- La **diferencia** con las variables es que **el valor de una constante no cambia**
- Un ejemplo es el número **PI** (3,14159...)



En *PseInt...*

2. Aprendiendo nuevos conceptos (ii)

Constantes

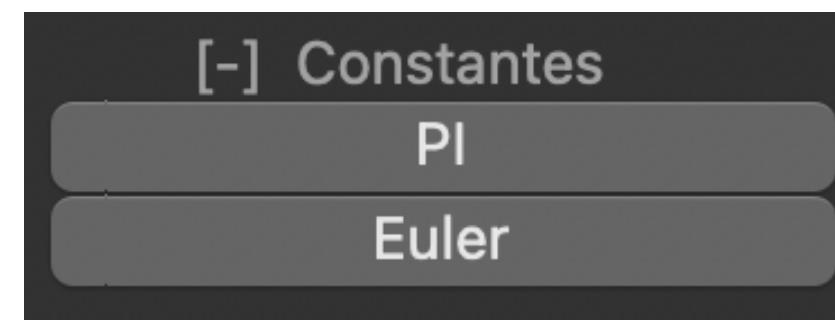
¿Qué es una constante?

- Al igual que una variable, es un espacio reservado en memoria que tiene asignado un identificador
- La **diferencia** con las variables es que **el valor de una constante no cambia**
- Un ejemplo es el número **PI** (3,14159...)

Memoria

PI			
3,14...			

En *PseInt...*

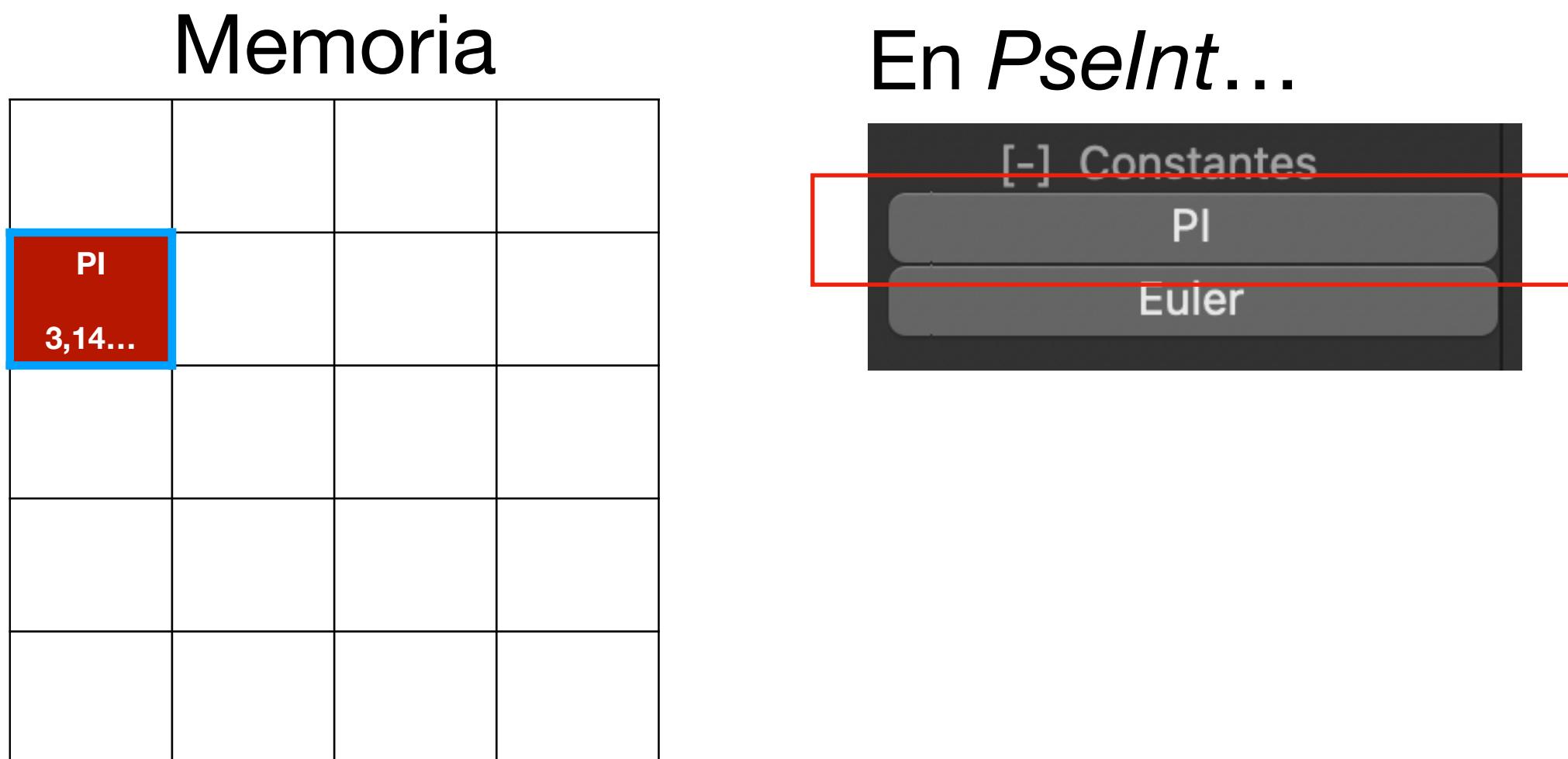


2. Aprendiendo nuevos conceptos (ii)

Constantes

¿Qué es una constante?

- Al igual que una variable, es un espacio reservado en memoria que tiene asignado un identificador
- La **diferencia** con las variables es que **el valor de una constante no cambia**
- Un ejemplo es el número **PI** (3,14159...)

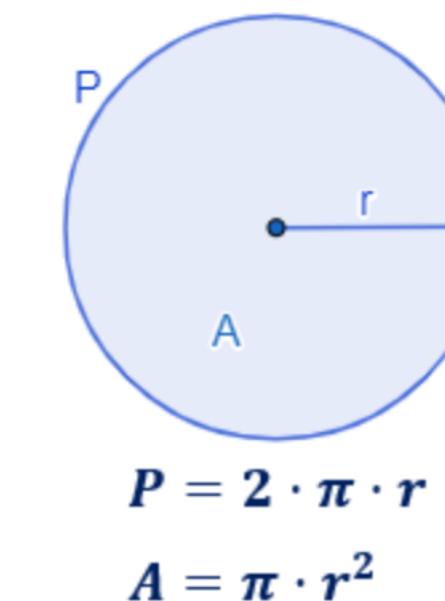
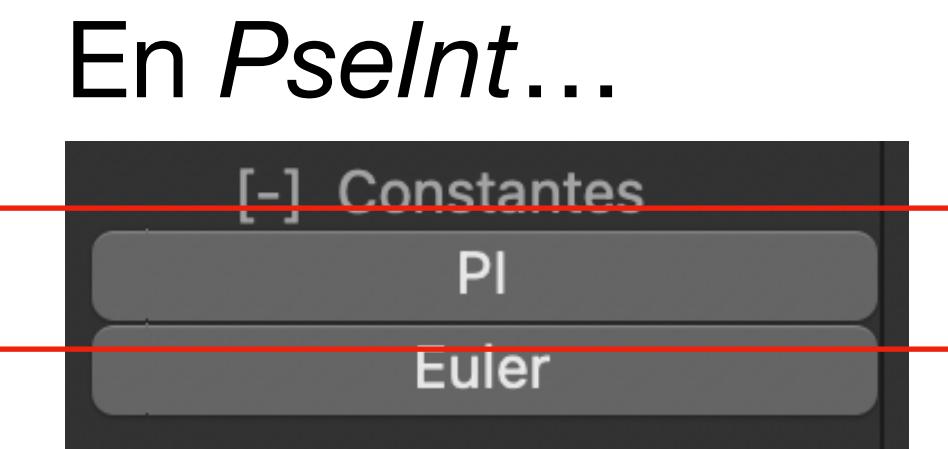
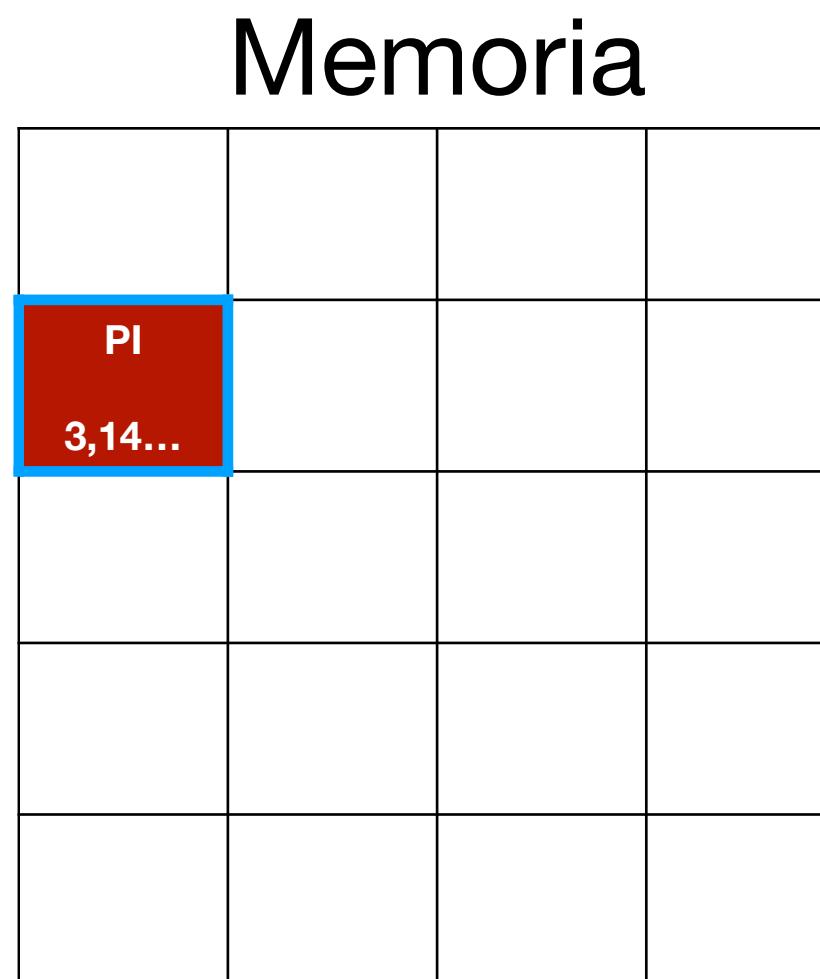


2. Aprendiendo nuevos conceptos (ii)

Constantes

¿Qué es una constante?

- Al igual que una variable, es un espacio reservado en memoria que tiene asignado un identificador
- La **diferencia** con las variables es que **el valor de una constante no cambia**
- Un ejemplo es el número **PI** (3,14159...)

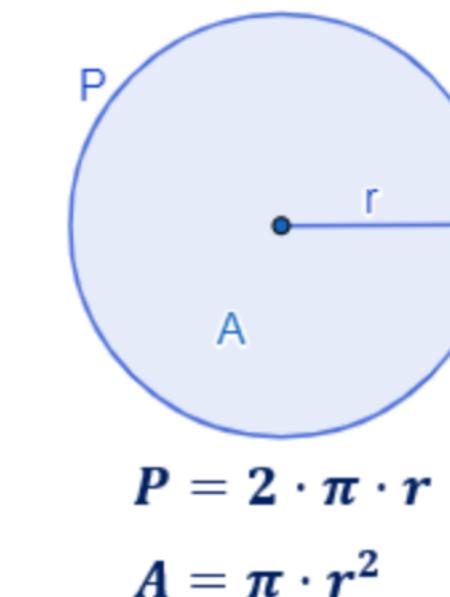
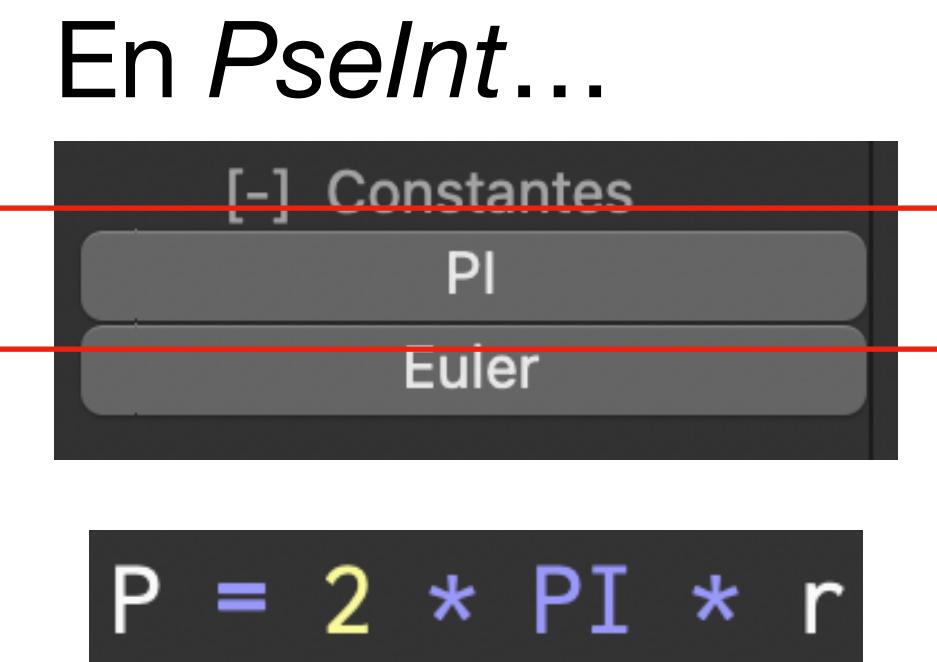
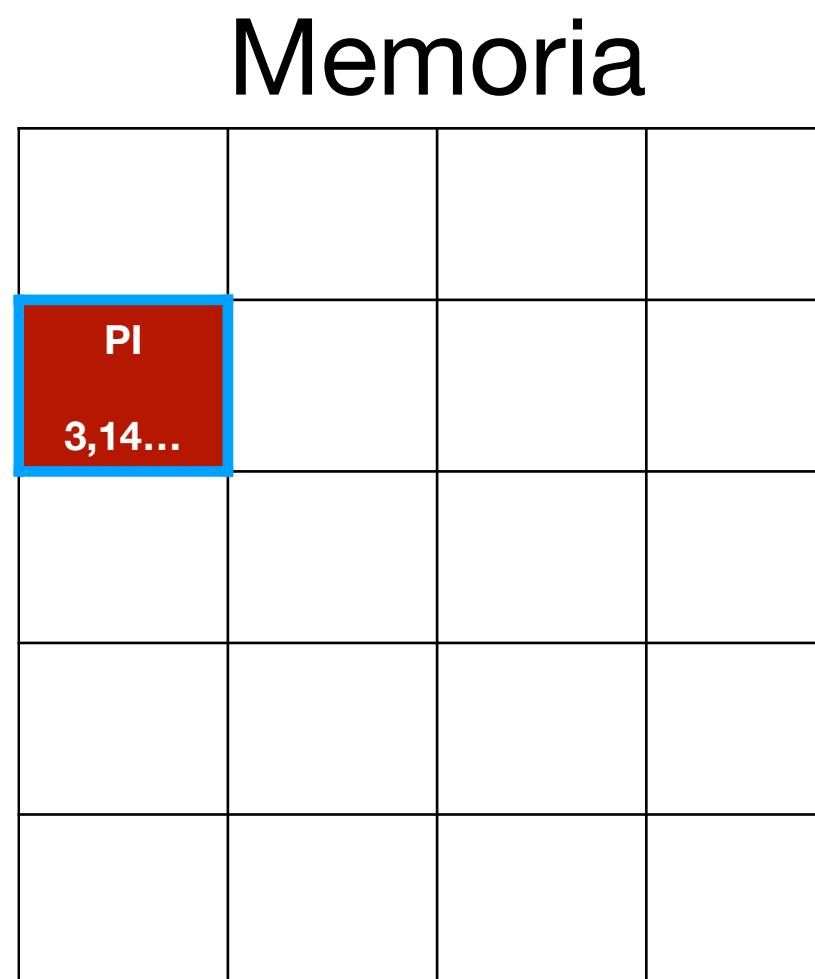


2. Aprendiendo nuevos conceptos (ii)

Constantes

¿Qué es una constante?

- Al igual que una variable, es un espacio reservado en memoria que tiene asignado un identificador
- La **diferencia** con las variables es que **el valor de una constante no cambia**
- Un ejemplo es el número **PI** (3,14159...)



2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

¿Para qué se utilizan los operadores lógicos?

Son operadores que se utilizan para combinar dos valores lógicos (V/F) y devolver un resultado lógico (V/F)

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

¿Para qué se utilizan los operadores lógicos?

Son operadores que se utilizan para combinar dos valores lógicos (V/F) y devolver un resultado lógico (V/F)

En *PseInt*...

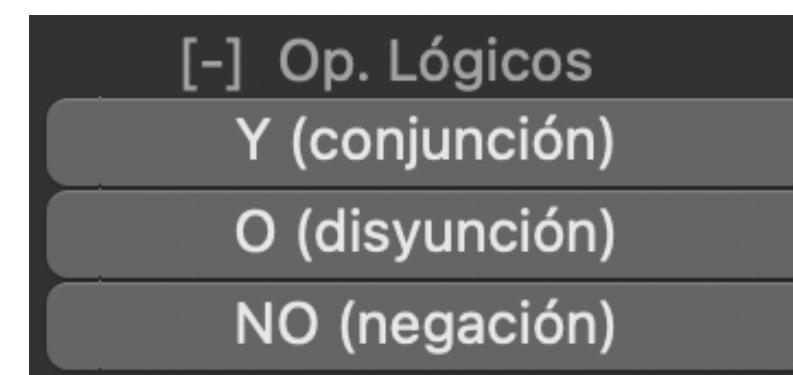
2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

¿Para qué se utilizan los operadores lógicos?

Son operadores que se utilizan para combinar dos valores lógicos (V/F) y devolver un resultado lógico (V/F)

En *PseInt*...



2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Operador lógico “Y”

- Devuelve “**Verdadero**” cuando `valor_logico_1` (`expresion_logica_1`) es “**Verdadero**” y `valor_logico_2` (`expresion_logica_2`) es “**Verdadero**”
- En otro caso, devuelve “**Falso**”
- Sintaxis

```
valor_logico_1 Y valor_logico_2  
expresion_logica_1 Y expresion_logica_2
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Operador lógico “Y”

- Devuelve “**Verdadero**” cuando `valor_logico_1` (`expresion_logica_1`) es “**Verdadero**” y `valor_logico_2` (`expresion_logica_2`) es “**Verdadero**”
- En otro caso, devuelve “**Falso**”
- Sintaxis

```
valor_logico_1 Y valor_logico_2  
expresion_logica_1 Y expresion_logica_2
```

```
Definir valorLogico Como Logico  
  
valorLogico = Verdadero Y Verdadero //Verdadero  
valorLogico = Verdadero Y Falso //Falso  
valorLogico = Falso Y Verdadero //Falso  
valorLogico = Falso Y Falso //Falso
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Operador lógico “Y”

- Devuelve “**Verdadero**” cuando `valor_logico_1` (`expresion_logica_1`) es “**Verdadero**” y `valor_logico_2` (`expresion_logica_2`) es “**Verdadero**”
- En otro caso, devuelve “**Falso**”
- Sintaxis

```
valor_logico_1 Y valor_logico_2  
expresion_logica_1 Y expresion_logica_2
```

```
Definir valorLogico Como Logico  
  
valorLogico = Verdadero Y Verdadero //Verdadero  
valorLogico = Verdadero Y Falso //Falso  
valorLogico = Falso Y Verdadero //Falso  
valorLogico = Falso Y Falso //Falso
```

```
Definir valorLogico Como Logico  
Definir num Como Entero  
num = 7  
  
valorLogico = (num < 10) Y (num > 5) //Verdadero  
valorLogico = (num < 10) Y (num > 7) //Falso  
valorLogico = (num > 10) Y (num ≤ 7) //Falso  
valorLogico = (num = 10) Y (num < 5) //Falso
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Operador lógico “O”

- Devuelve “**Verdadero**” cuando `valor_logico_1` (`expresion_logica_1`) o `valor_logico_2` (`expresion_logica_2`) es “**Verdadero**”
- En otro caso, devuelve “**Falso**”
- Sintaxis

```
valor_logico_1 O valor_logico_2  
expresion_logica_1 O expresion_logica_2
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Operador lógico “O”

- Devuelve “**Verdadero**” cuando `valor_logico_1` (`expresion_logica_1`) o `valor_logico_2` (`expresion_logica_2`) es “**Verdadero**”
- En otro caso, devuelve “**Falso**”
- Sintaxis

```
valor_logico_1 O valor_logico_2  
expresion_logica_1 O expresion_logica_2
```

```
Definir valorLogico Como Logico  
  
valorLogico = Verdadero O Verdadero //Verdadero  
valorLogico = Verdadero O Falso //Verdadero  
valorLogico = Falso O Verdadero //Verdadero  
valorLogico = Falso O Falso //Falso
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Operador lógico “O”

- Devuelve “**Verdadero**” cuando `valor_logico_1` (`expresion_logica_1`) o `valor_logico_2` (`expresion_logica_2`) es “**Verdadero**”
- En otro caso, devuelve “**Falso**”
- Sintaxis

```
valor_logico_1 O valor_logico_2  
expresion_logica_1 O expresion_logica_2
```

```
Definir valorLogico Como Logico
```

```
valorLogico = Verdadero O Verdadero //Verdadero  
valorLogico = Verdadero O Falso //Verdadero  
valorLogico = Falso O Verdadero //Verdadero  
valorLogico = Falso O Falso //Falso
```

```
Definir valorLogico Como Logico
```

```
Definir num Como Entero
```

```
num = 7
```

```
valorLogico = (num < 10) O (num > 5) //Verdadero  
valorLogico = (num < 10) O (num > 7) //Verdadero  
valorLogico = (num > 10) O (num ≤ 7) //Verdadero  
valorLogico = (num = 10) O (num < 5) //Falso
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Operador lógico “NO”

- Devuelve “**Verdadero**” cuando valor_logico (expresion_logica) es “**Falso**”
- En otro caso, devuelve “**Falso**”
- Sintaxis

```
NO valor_logico  
NO expresion_logica
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Operador lógico “NO”

- Devuelve “**Verdadero**” cuando valor_logico (expresion_logica) es “**Falso**”
- En otro caso, devuelve “**Falso**”
- Sintaxis

```
NO valor_logico  
NO expresion_logica
```

```
Definir valorLogico Como Logico  
  
valorLogico = NO Verdadero //Falso  
valorLogico = NO Falso //Verdadero
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Operador lógico “NO”

- Devuelve “**Verdadero**” cuando `valor_logico` (`expresion_logica`) es “**Falso**”
- En otro caso, devuelve “**Falso**”
- Sintaxis

```
NO valor_logico  
NO expresion_logica
```

```
Definir valorLogico Como Logico  
  
valorLogico = NO Verdadero //Falso  
valorLogico = NO Falso //Verdadero
```

```
Definir valorLogico Como Logico  
Definir num Como Entero  
num = 7  
  
valorLogico = NO (num > 10) //Verdadero (num ≤ 10)  
valorLogico = NO (num < 10) //Falso (num ≥ 10)
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

¿Cómo utilizar los **op. lógicos** en una estructura condicional **SI-ENTONCES**?

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

¿Cómo utilizar los **op. lógicos** en una estructura condicional **SI-ENTONCES?**

```
Si expresion_logica Entonces
| ejecucion_codigo_1
SiNo
| ejecucion_codigo_2
FinSi
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

¿Cómo utilizar los **op. lógicos** en una estructura condicional **SI-ENTONCES?**

```
Si expresion_logica Entonces
| ejecucion_codigo_1
SiNo
| ejecucion_codigo_2
FinSi
```

```
//Operador lógico "Y"
Si (expresion_logica_1) Y (expresion_logica_2) Entonces
| ejecucion_codigo_1
SiNo
| ejecucion_codigo_2
FinSi
//Operador lógico "O"
Si (expresion_logica_1) O (expresion_logica_2) Entonces
| ejecucion_codigo_1
SiNo
| ejecucion_codigo_2
FinSi
//Operador lógico "NO"
Si NO (expresion_logica) Entonces
| ejecucion_codigo_1
SiNo
| ejecucion_codigo_2
FinSi
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Ejemplo

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Ejemplo

```
Definir num Como Real
num = 0

Escribir "Introduce un número natural de dos cifras"
Leer num

Si NO (num > 0) Entonces
    Escribir "No has introducido un número natural"
SiNo
    Si (num ≥ 10) Y (num ≤ 99) Entonces
        Escribir "El número está entre el 10 y el 99"
    SiNo
        Escribir "El número es menor a 10"
    FinSi
FinSi
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Ejemplo

```
Definir num Como Real  
num = 0  
  
Escribir "Introduce un número natural de dos cifras"  
Leer num  
  
Si NO (num > 0) Entonces  
    Escribir "No has introducido un número natural"  
SiNo  
    Si (num ≥ 10) Y (num ≤ 99) Entonces  
        Escribir "El número está entre el 10 y el 99"  
    SiNo  
        Escribir "El número es menor a 10"  
    FinSi  
FinSi
```

```
*** Ejecución Iniciada. ***  
Introduce un número natural de dos cifras  
> 23  
El número está entre el 10 y el 99  
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Ejemplo

```
Definir num Como Real  
num = 0  
  
Escribir "Introduce un número natural de dos cifras"  
Leer num  
  
Si NO (num > 0) Entonces  
    Escribir "No has introducido un número natural"  
SiNo  
    Si (num ≥ 10) Y (num ≤ 99) Entonces  
        Escribir "El número está entre el 10 y el 99"  
    SiNo  
        Escribir "El número es menor a 10"  
    FinSi  
FinSi
```

*** Ejecución Iniciada. ***
Introduce un número natural de dos cifras
> 23
El número está entre el 10 y el 99
*** Ejecución Finalizada. ***

*** Ejecución Iniciada. ***
Introduce un número natural de dos cifras
> 6
El número es menor a 10
*** Ejecución Finalizada. ***

2. Aprendiendo nuevos conceptos (iii)

Operadores lógicos

Ejemplo

```
Definir num Como Real  
num = 0  
  
Escribir "Introduce un número natural de dos cifras"  
Leer num  
  
Si NO (num > 0) Entonces  
    Escribir "No has introducido un número natural"  
SiNo  
    Si (num ≥ 10) Y (num ≤ 99) Entonces  
        Escribir "El número está entre el 10 y el 99"  
    SiNo  
        Escribir "El número es menor a 10"  
    FinSi  
FinSi
```

```
*** Ejecución Iniciada. ***  
Introduce un número natural de dos cifras  
> 23  
El número está entre el 10 y el 99  
*** Ejecución Finalizada. ***
```

```
*** Ejecución Iniciada. ***  
Introduce un número natural de dos cifras  
> 6  
El número es menor a 10  
*** Ejecución Finalizada. ***
```

```
*** Ejecución Iniciada. ***  
Introduce un número natural de dos cifras  
> -1  
No has introducido un número natural  
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (iv)

Selectiva SEGUN

¿Para qué sirve la estructura selectiva SEGUN?

Permite ejecutar un código diferente en función del **valor de una variable**

2. Aprendiendo nuevos conceptos (iv)

Selectiva SEGUN

¿Para qué sirve la estructura selectiva SEGUN?

Permite ejecutar un código diferente en función del **valor de una variable**

¿Cuál es su sintaxis?

2. Aprendiendo nuevos conceptos (iv)

Selectiva SEGUN

¿Para qué sirve la estructura selectiva SEGUN?

Permite ejecutar un código diferente en función del **valor de una variable**

¿Cuál es su sintaxis?

```
Segun variable Hacer
    opcion_1:
        ejecucion_codigo_1
    opcion_2:
        ejecucion_codigo_2
    ...
De otro Modo:
    ejecucion_codigo_otro_modo
FinSegun
```

2. Aprendiendo nuevos conceptos (iv)

Selectiva SEGUN

¿Para qué sirve la estructura selectiva SEGUN?

Permite ejecutar un código diferente en función del **valor de una variable**

¿Cuál es su sintaxis?

```
Segun variable Hacer
    opcion_1:
        ejecucion_codigo_1
    opcion_2:
        ejecucion_codigo_2
    ...
De otro Modo:
    ejecucion_codigo_otro_modo
FinSegun
```

La parte **De otro Modo** es
OPCIONAL

2. Aprendiendo nuevos conceptos (iv)

Selectiva SEGUN

Ejemplo

2. Aprendiendo nuevos conceptos (iv)

Selectiva SEGUN

Ejemplo

```
Definir nombre Como Texto
nombre = ""

Escribir "Introduce tu nombre"
Leer nombre

Segun nombre Hacer
    "Juan":
        Escribir "Bienvenido Juan!"
    "Maria":
        Escribir "Bienvenida Maria!"
    "Pepa":
        Escribir "Bienvenida Pepa!"
De otro Modo:
    Escribir "Bienvenido, seas quien seas!"
FinSegun
```

2. Aprendiendo nuevos conceptos (iv)

Selectiva SEGUN

Ejemplo

```
Definir nombre Como Texto
nombre = ""

Escribir "Introduce tu nombre"
Leer nombre

Segun nombre Hacer
    "Juan":
        Escribir "Bienvenido Juan!"
    "Maria":
        Escribir "Bienvenida Maria!"
    "Pepa":
        Escribir "Bienvenida Pepa!"
De otro Modo:
    Escribir "Bienvenido, seas quien seas!"
FinSegun
```

```
*** Ejecución Iniciada. ***
Introduce tu nombre
> Juan
Bienvenido Juan!
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (iv)

Selectiva SEGUN

Ejemplo

```
Definir nombre Como Texto
nombre = ""

Escribir "Introduce tu nombre"
Leer nombre

Segun nombre Hacer
    "Juan":
        Escribir "Bienvenido Juan!"
    "Maria":
        Escribir "Bienvenida Maria!"
    "Pepa":
        Escribir "Bienvenida Pepa!"
De otro Modo:
    Escribir "Bienvenido, seas quien seas!"
FinSegun
```

```
*** Ejecución Iniciada. ***
Introduce tu nombre
> Juan
Bienvenido Juan!
*** Ejecución Finalizada. ***
```

```
*** Ejecución Iniciada. ***
Introduce tu nombre
> Pepa
Bienvenida Pepa!
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (iv)

Selectiva SEGUN

Ejemplo

```
Definir nombre Como Texto
nombre = ""

Escribir "Introduce tu nombre"
Leer nombre

Segun nombre Hacer
    "Juan":
        Escribir "Bienvenido Juan!"
    "Maria":
        Escribir "Bienvenida Maria!"
    "Pepa":
        Escribir "Bienvenida Pepa!"
De otro Modo:
    Escribir "Bienvenido, seas quien seas!"
FinSegun
```

```
*** Ejecución Iniciada. ***
Introduce tu nombre
> Juan
Bienvenido Juan!
*** Ejecución Finalizada. ***
```

```
*** Ejecución Iniciada. ***
Introduce tu nombre
> Pepa
Bienvenida Pepa!
*** Ejecución Finalizada. ***
```

```
*** Ejecución Iniciada. ***
Introduce tu nombre
> Antonio
Bienvenido, seas quien seas!
*** Ejecución Finalizada. ***
```

**Y ahora...
vamos a ver unos ejemplo sobre lo que
hemos visto**

:-)