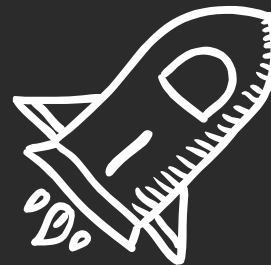




Semana 1

¡Bienvenidos!



Temas de hoy

1

Sintaxis básica:
variables, tipos de
datos, operadores

2

Pseudocódigo:
Estructura y práctica de
ejemplo mediante
Pseint.

1

Sintaxis básica

Para seguir con el aprendizaje, ahora veremos otros conceptos básicos propios de la programación.

En la primer clase estuvimos usando algunos de estos conceptos pero ahora vamos a darle nombre, además de ver sus propiedades.

Variable, tipos de datos, operadores

Variable

Esta representa un espacio en memoria donde se pueden almacenar datos de distintos tipos, realizar operaciones o cambiar de valor o tipo de dato.

Las variables tienen ciertas características como un nombre único, un valor que puede cambiar, aceptan distintos tipos de datos y pueden ser globales o locales.

Las variables son de los recursos más valiosos que usaremos en nuestra vida como programadores.

Características de una variable

Nombre único: cada variable tiene que tener un nombre descriptivo o que identifique a que se refiere el valor que se estará guardando. Por ejemplo: para una variable donde se guardará una edad, el nombre de la misma debería ser "edad".

Valor cambiante: durante la ejecución el programa, el valor almacenado dentro de la variable puede cambiar. Siguiendo con el ejemplo anterior, la variable "edad" no va a tener siempre la edad fija de 18 supongamos. Durante la ejecución del programa este valor puede ser 21 o 45 o 72, es decir, puede cambiar a lo largo de la ejecución del programa.

Variable, tipos de datos, operadores

Características de una variable

Alcance: con esto nos referimos a que las variables puede tener un alcance global o local, es decir, éstas pueden ser accesibles dentro de un bloque de código o fuera de él, refiriéndonos a estas variables como “globales” pero, si solo pueden ser accesibles dentro de un bloque de código y no fuera de él, éstas son variables “locales”.

Lo vemos en un ejemplo de código con Python:

```
2_variables.py > ...
1  variable_global = 100
2
3  def multiplicar(num):
4      variable_local = 2
5      return num * variable_local
6
7  print(multiplicar(2))
8  print(variable_global)
9  print(variable_local)
```

Esta variable fue definida por fuera de una función, por lo que se puede acceder desde cualquier parte del programa.

Sin embargo, a esta variable no se puede acceder sin llamar a la función que la contiene.

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

PS C:\Users\Pc\Documents\Info> & C:/Users/Pc/AppData/Local/Programs/Python/Python3...
ables.py

4 valor de variable local
100 valor de variable global

Traceback (most recent call last):
File "c:\Users\Pc\Documents\Info\2_variables.py", line 9, in <module>
print(variable_local)
^^^^^^^^^^^^^^^^

NameError: name 'variable_local' is not defined. Did you mean: 'variable_global'?

PS C:\Users\Pc\Documents\Info>

Este error indica que se está llamando a una variable que no "existe" o que no está definida, pero en realidad se la está intentando llamar de forma global cuando la misma es local.

Variable, tipos de datos, operadores

Tipos de datos

Una variable puede almacenar distintos tipos de datos, es decir, puede almacenar letras, números enteros o números decimales u otro tipo de datos que representan valores de verdad como Verdadero o Falso.

Estos tipos de datos comunes se pueden usar en diversos lenguajes de programación como Python, Java, JavaScript, C++, C#, entre otros.

Entero (int o integer): representan números sin la parte decimal y pueden ser positivos o negativos.

Decimal (float o double): representan números reales con su parte decimal.

Cadena de caracteres (char o string): representan letras o símbolos. En este caso char hace referencia a un solo carácter (como "a" o "A") y string hace referencia a una secuencia de caracteres (como "Hola mundo.")

Booleanos (bool): representan valores de verdad. Solo tienen estos dos valores. Verdadero (True) o Falso (False) y sirven mucho para controlar decisiones en el código.

Existen otros tipos de datos no tan comunes que veremos a lo largo del curso, como por ejemplo null o None.

Variable, tipos de datos, operadores

Operadores

Durante nuestra vida hemos visto y usado operadores, por lo que no te va a resultar algo muy raro.

A los operadores los usamos en operaciones sobre datos ya sea con símbolos o palabras reservadas. Generalmente se usan sobre operaciones matemáticas, lógicas o de asignación.

En programación veremos operadores aritméticos, operadores booleanos u operadores de asignación.

Operadores aritméticos: Entre los cuales se encuentran los operadores de suma (+), resta (-), multiplicación (*), división (/), entre otros.

Operadores booleanos: que representan valores de verdadero o falso, True o False, respectivamente.

Operadores de asignación: los cuales utilizamos para asignar un valor (=), acumular un valor (+=), entre otros más.

Todos estos operadores, entre otros, los podrás ver a detalle en el material complementario.

2

Pseudocódigo

Como ya vimos de manera introductoria en la clase anterior, el pseudocódigo es un método utilizado en la programación para describir algoritmos, o procesos, mediante un lenguaje simplificado, más cercano al lenguaje humano que al un lenguaje de programación como tal.

En esta primer etapa veremos pseudocódigo mediante el programa Pseint, el cual, nos va a servir para ejecutar código escrito en pseudocódigo.

Estructura del pseudocódigo

- ▶ **Encabezado:** el cual puede describir brevemente la función del algoritmo mediante un nombre descriptivo.
- ▶ **Declaración de variables:** esta parte es fundamental para definir las variables que utilizaremos en el pseudocódigo.
- ▶ **Cuerpo o proceso:** desde aquí comenzamos con las instrucciones paso a paso que se necesitarán para ejecutar el algoritmo. Esta parte incluye la lógica y las operaciones necesarias.

El pseudocódigo no tiene una sintaxis precisa como lo tiene que tener un lenguaje de programación, pero se debe seguir esta estructura básica para facilitar la comprensión y el análisis lógico del algoritmo.

Primeros pasos con Pseint

Gracias a este programa, podremos realizar nuestros primeros pseudocódigos y probarlos de manera funcional, pudiendo ejecutar, tal cual como en un lenguaje de programación.

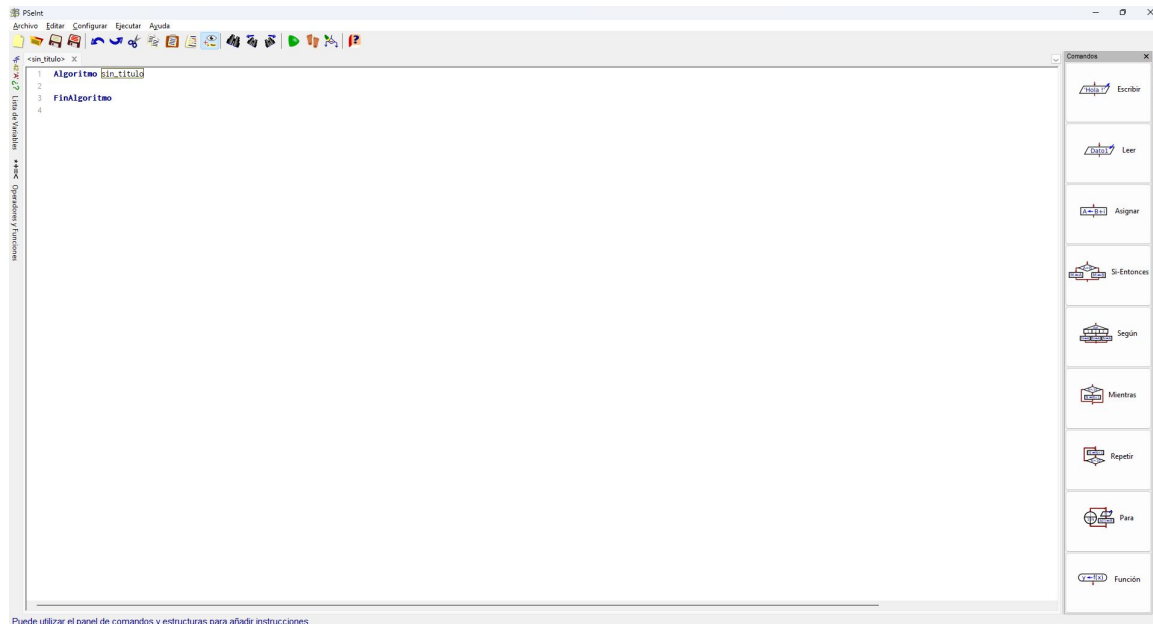
Pseint contempla la estructura básica del pseudocódigo, por lo que teniendo en cuenta esta estructura vamos a ver ejemplos en este programa.

Primero que nada haremos un repaso por el programa.



Primeros pasos con Pseint

Una vez que abrimos el programa tendremos esta interfaz inicial, la cual tu profesor o profesora te la explicará en clases.



Primeros pasos con Pseint

Lo más importante a tener en cuenta, como se mencionó anteriormente, es tener comprendida la estructura correcta del pseudocódigo, la cual nos va a permitir realizar un algoritmo legible, eficaz y listo para traducir a cualquier lenguaje.

En la imagen podrás encontrar la estructura base a seguir para cada algoritmo.

```
1 Algoritmo nombre_del_algoritmo
2
3     Definir variable_1, variable_2, aux Como Entero
4
5
6     Imprimir 'Ingreso de datos.'
7     Leer variable_1
8
9     Imprimir 'Ingreso de datos.'
10    Leer variable_1
11
12
13    aux ← variable_1 + variable_2
14
15
16    Imprimir 'Resultado: ', aux
17
18
19 FinAlgoritmo
```

Partes del algoritmo en Pseint

- ▶ Lo primero es colocar un nombre identificador a nuestro algoritmo.
- ▶ Luego hay que definir las variables y el tipo de dato a usar.
- ▶ Posterior a esto se comienzan a ingresar y almacenar datos.
- ▶ Luego de la recolección de datos, sigue la parte lógica donde se trabaja con los datos recolectados.
- ▶ Se muestra el resultado y se da fin al algoritmo.

estructura_base.psc X

```
1  Algoritmo nombre_del_algoritmo
```

```
2
```

```
3      Definir variable_1, variable_2, aux Como Entero
```

```
4
```

```
6      Imprimir 'Ingreso de datos.'
```

```
7      Leer variable_1
```

```
8
```

```
13     aux ← variable_1 + variable_2
```

```
14
```

```
16     Imprimir 'Resultado: ', aux
```

```
17
```

```
18
```

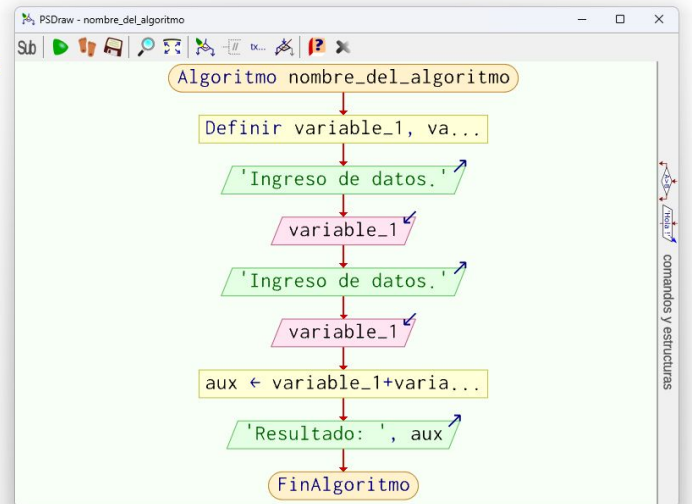
```
19 FinAlgoritmo
```

Funciones adicionales de Pseint

- ▶ Algo más que nos presenta Pseint es la posibilidad de realizar **Diagramas de Flujo** desde la escritura del algoritmo.

Incluso podríamos generar primero el Diagrama de Flujo y en base a esto que se genere el pseudocódigo.

```
estructura_base.psc" X
1 Algoritmo nombre_del_algoritmo
2
3   Definir variable_1, variable_2, aux Como Entero
4
5
6   Imprimir 'Ingreso de datos.'
7   Leer variable_1
8
9   Imprimir 'Ingreso de datos.'
10  Leer variable_1
11
12
13  aux ← variable_1 + variable_2
14
15
16  Imprimir 'Resultado: ', aux
17
18
19 FinAlgoritmo
```

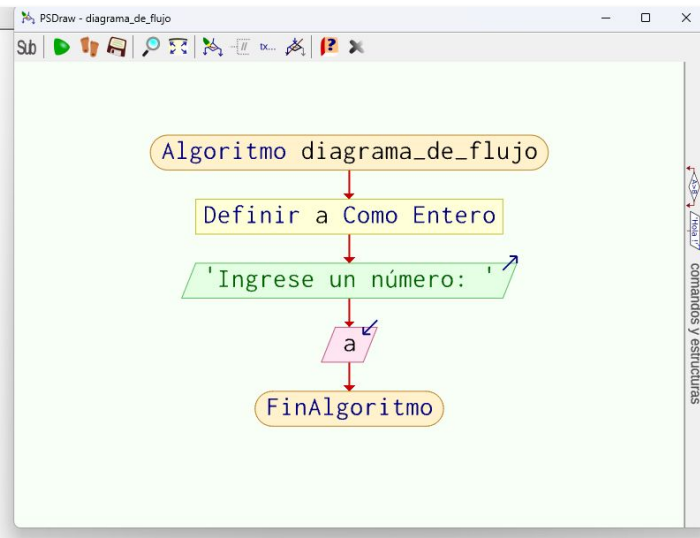


Funciones adicionales de Pseint

- ▶ En este ejemplo, el pseudocódigo fue creado a partir del Diagrama de Flujo.

Estos diagramas se utilizan para hacer un algoritmo de forma rápida y más gráfica.

```
diagrama_de_flujo.psc+ X
1 Algoritmo diagrama_de_flujo
2
3     Definir a como entero
4
5     Imprimir 'Ingrese un número: '
6     Leer a
7
8 FinAlgoritmo
9
```





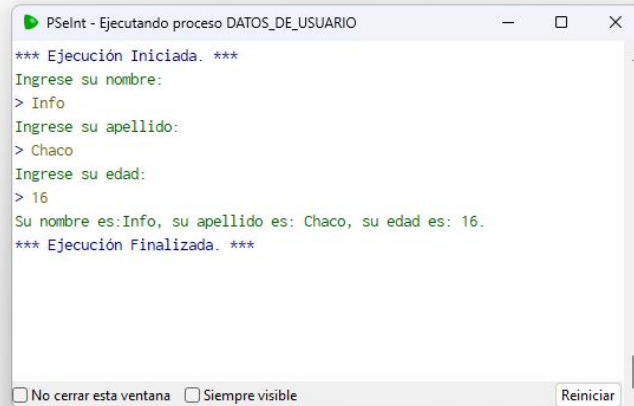
¿Listos para arrancar?

Ya conocemos el programa y su estructura, por lo que es momento de arrancar con algunos ejemplos para empezar a comprender mejor lo aprendido hasta acá. Recordá que en el material complementario vas a encontrar una profundización mayor y en clases vas a llevar a la práctica todos estos conceptos y ejemplos... ¡arranquemos!

Ejemplo 1

- ▶ Analizaremos este ejemplo de manera completa en clase y, en el material complementario vas a tener más ejemplos y su análisis para mejorar tu comprensión.

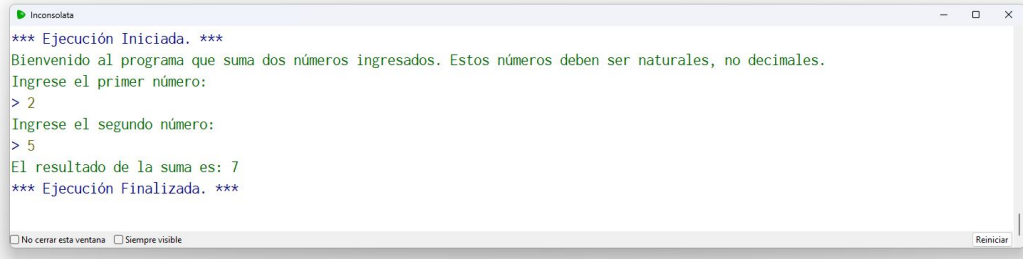
```
1  Algoritmo datos_de_usuario
2
3      Definir nombre, apellido Como Caracter
4      Definir edad Como Entero
5
6      Imprimir 'Ingrese su nombre: '
7      Leer nombre
8      Imprimir 'Ingrese su apellido: '
9      Leer apellido
10     Imprimir 'Ingrese su edad: '
11     Leer edad
12
13     Imprimir 'Su nombre es:', nombre, ', su apellido es: ', apellido, ', su edad es: ', edad, '.'
14
15 FinAlgoritmo
```



Ejemplo 2

- ▶ Analizaremos este ejemplo de manera completa en clase y, en el material complementario vas a tener más ejemplos y su análisis para mejorar tu comprensión.

```
1 Algoritmo suma_de_dos_numeros
2
3     Definir num1, num2, suma Como Entero
4
5     Imprimir 'Bienvenido al programa que suma dos números ingresados. Estos números deben ser naturales, no decimales.'
6     Imprimir 'Ingrese el primer número: '
7     Leer num1
8     Imprimir 'Ingrese el segundo número: '
9     Leer num2
10
11     suma = num1 + num2
12
13     Imprimir 'El resultado de la suma es: ', suma
14
15 FinAlgoritmo
16
```



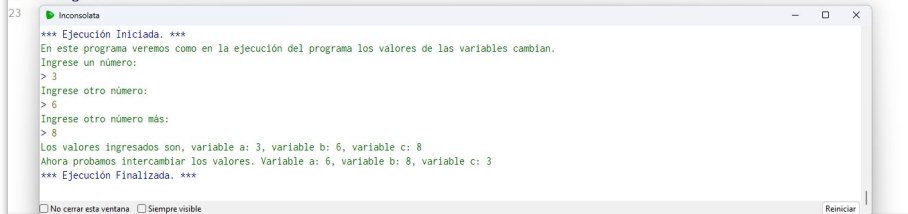
```
*** Ejecución Iniciada. ***
Bienvenido al programa que suma dos números ingresados. Estos números deben ser naturales, no decimales.
Ingrese el primer número:
> 2
Ingrese el segundo número:
> 5
El resultado de la suma es: 7
*** Ejecución Finalizada. ***
```

☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

Ejemplo 3

- ▶ Analizaremos este ejemplo de manera completa en clase y, en el material complementario vas a tener más ejemplos y su análisis para mejorar tu comprensión.

```
1 Algoritmo cambio_de_valores
2
3 Definir a, b, c, aux Como Entero
4
5 Imprimir 'En este programa veremos como en la ejecución del programa los valores de las variables cambian.'
6 Imprimir 'Ingrese un número: '
7 Leer a
8 Imprimir 'Ingrese otro número: '
9 Leer b
10 Imprimir 'Ingrese otro número más: '
11 Leer c
12
13 Imprimir 'Los valores ingresados son, variable a: ', a, ', variable b: ', b, ', variable c: ', c
14
15 aux = a //utilizamos una variable auxiliar para guardar el valor de la variable a para no perderlo en las asignaciones
16 a = b // ahora a, tiene el valor de b
17 b = c // ahora b, tiene el valor de c
18 c = aux // ahora c, tiene el valor de que tenía a al principio, pero lo guardamos en la variable auxiliar
19
20 Imprimir 'Ahora probamos intercambiar los valores. Variable a: ', a, ', variable b: ', b, ', variable c: ', c
21
22 FinAlgoritmo
```



```
*** Ejecución Iniciada. ***
En este programa veremos como en la ejecución del programa los valores de las variables cambian.
Ingrese un número:
> 3
Ingrese otro número:
> 6
Ingrese otro número más:
> 8
Los valores ingresados son, variable a: 3, variable b: 6, variable c: 8
Ahora probamos intercambiar los valores. Variable a: 6, variable b: 8, variable c: 3
*** Ejecución Finalizada. ***
```

Ejemplo 4

- ▶ Analizaremos este ejemplo de manera completa en clase y, en el material complementario vas a tener más ejemplos y su análisis para mejorar tu comprensión.

```
1 Algoritmo cambio_de_valores_letras
2
3 Definir a, b, c, aux Como Caracter
4
5 Imprimir 'En este programa veremos como en la ejecución del programa los valores de las variables cambian, pero ahora con letras!'
6 Imprimir 'Ingrese una palabra: '
7 Leer a
8 Imprimir 'Ingrese otra palabra: '
9 Leer b
10 Imprimir 'Ingrese otra palabra más: '
11 Leer c
12
13 Imprimir 'Las palabras ingresadas son, variable a: ', a, ', variable b: ', b, ', variable c: ', c
14
15 aux = a //utilizamos una variable auxiliar para guardar el valor de la variable a para no perderlo en las asignaciones
16 a = b // ahora a, tiene el valor de b
17 b = c // ahora b, tiene el valor de c
18 c = aux // ahora c, tiene el valor de que tenía a al principio, pero lo guardamos en la variable auxiliar
19
20 Imprimir 'Ahora probamos intercambiar los valores. Variable a: ', a, ', variable b: ', b, ', variable c: ', c
21
22 FinAlgoritmo
23
```

```
*** Ejecución Iniciada. ***
En este programa veremos como en la ejecución del programa los valores de las variables cambian, pero ahora con letras!
Ingrese una palabra:
> Hola
Ingrese otra palabra:
> Mundo
Ingrese otra palabra más:
> Info
Las palabras ingresadas son, variable a: Hola, variable b: Mundo, variable c: Info
Ahora probamos intercambiar los valores. Variable a: Mundo, variable b: Info, variable c: Hola
*** Ejecución Finalizada. ***
```

Lo que vimos en clase

- 1 Aprendimos de sintaxis básica: variables, tipos de datos, operadores.
- 2 Pseudocódigo y su estructura.
- 3 Conocimos la herramienta Pseint, algunas funciones y características.
- 4 Practicaste y ejercitaste con ejemplos simples en pseudocódigo.



Recordá profundizar

Aparte de estos slides, te recordamos que en el campus encontrarás material complementario donde se pueden ver estos temas con una mayor profundidad, por lo que te recomendamos que

¡no los pases por alto! 

Además, la asistencia a clases despejará tus dudas y en las mentorías vas a poder practicar para

¡consolidar los conocimientos! 



**¡Nos vemos
En la próxima
clase!**

