UADE

Introducción a la Algoritmia

Clase 2

Temas de la clase 2

- Estructuras básicas en el ámbito de la programación.
- Estructura secuencial.
- Constantes, variables y expresiones.
- Tipos de dato. Asignación.
- Operadores aritméticos.
- Orden de evaluación

Estructuras básicas en el ámbito de la programación.

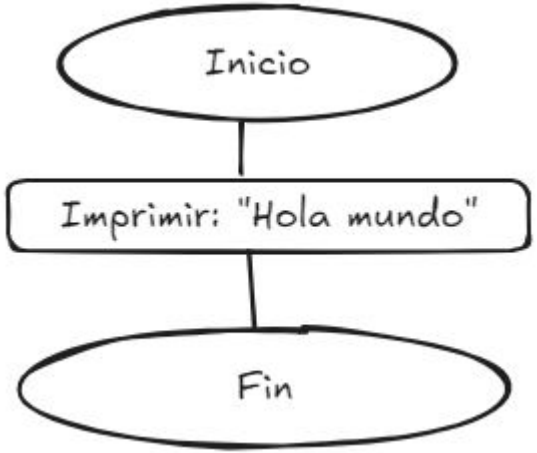
Definición: La estructura más básica donde las instrucciones se ejecutan una tras otra en el orden en que aparecen.

Ejemplo en pseudocódigo:

Inicio

Imprimir: Hola mundo

Fin



¿ Qué es Python ?

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, que se destaca por su simplicidad y legibilidad. Fue creado por Guido van Rossum y lanzado por primera vez en 1991.

Características de Python

- Sintaxis Simple y Legible: Python tiene una sintaxis que permite a los desarrolladores escribir código de manera clara y concisa. Esto hace que el lenguaje sea fácil de aprender y usar.
- Interpretado: Python es un lenguaje interpretado, lo que significa que el código se ejecuta línea por línea, facilitando la depuración y el desarrollo rápido.
- Multiparadigma: Python soporta múltiples paradigmas de programación, incluyendo programación orientada a objetos, programación imperativa y programación funcional.
- Portabilidad: El código Python puede ejecutarse en diferentes plataformas, como Windows, macOS y varias distribuciones de Linux, sin necesidad de modificar el código fuente.

Programa “Hola mundo”

1- Abrir el programa IDLE

2- Seleccionar File , luego new File

3- En la nueva ventana, escribir: print("Hola mundo")

4- Guardar como ejercicio1.py en alguna carpeta.

5- Seleccionar Run, Run Module.

Alternativa en Visual Studio code.

1- Abrir visual studio code.

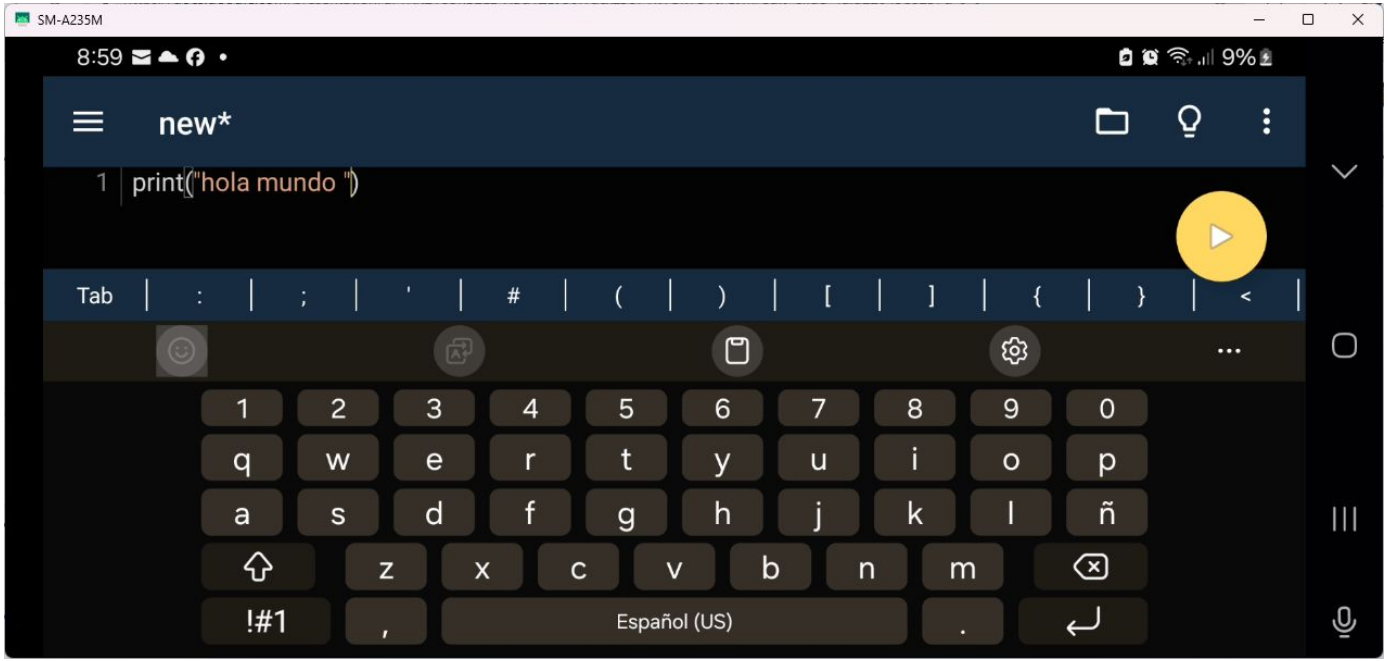
2- Seleccionar la carpeta creada en el ejercicio anterior.

3- Seleccionar el archivo a ejecutar y en el menú: Run, run without debugger

También se pueden instalar extensiones, como python de Microsoft

Alternativa en el celular

Buscar en Play Store: Pydroid 3: al ingresar, se puede escribir código python



Actividad 1

1- Realizar un programa que imprima el nombre de la materia y debajo el nombre del alumno.

2- Realizar un programa que imprima el nombre de la materia y debajo el nombre del alumno dejando una línea en blanco entre los dos.

Solución actividad 1

```
print("Introducción a la algoritmia")

print()

print("Nombre del estudiante")
```

Comentarios

En Python, los comentarios se escriben usando el símbolo #. Todo lo que sigue después de este símbolo en la misma línea es considerado un comentario y no es ejecutado por el intérprete.
Aquí tienes un ejemplo:
python
Copiar código
Este es un comentario en Python
Los comentarios son ignorados por el intérprete y son útiles para explicar el código
Los comentarios se utilizan para documentar el código y explicar lo que hace, lo que es útil tanto para el programador como para otras personas que puedan leer el código.

¿Qué son las variables ?

Una variable en programación es un contenedor que se utiliza para almacenar datos que pueden cambiar o ser referenciados a lo largo de la ejecución de un programa. Las variables permiten a los desarrolladores nombrar y manipular datos de una manera organizada y comprensible.

Características de las Variables

Nombre: Cada variable tiene un nombre que se usa para referenciarla. Este nombre debe seguir ciertas reglas de sintaxis del lenguaje de programación.

Tipo: Las variables pueden almacenar diferentes tipos de datos, como números enteros, números decimales, cadenas de texto, listas, diccionarios, entre otros.

Valor: Es el dato o información almacenada en la variable.

Declaración y Asignación de Variables en Python

En Python, declarar y asignar un valor a una variable es sencillo. No es necesario especificar el tipo de dato explícitamente, ya que Python es un lenguaje de tipado dinámico.

Características de las Variables

Nombre: Cada variable tiene un nombre que se usa para referenciarla. Este nombre debe seguir ciertas reglas de sintaxis del lenguaje de programación.

Tipo: Las variables pueden almacenar diferentes tipos de datos, como números enteros, números decimales, cadenas de texto, listas, diccionarios, entre otros.

Valor: Es el dato o información almacenada en la variable.

Ejemplos de Declaración y Asignación de Variables

```
# Declaración y asignación de variables
nombre = "Juan"         # Una cadena de texto
edad = 20               # Un número entero
altura = 1.75           # Un número decimal (float)
es_estudiante = True    # Un valor booleano

# Imprimir los valores de las variables
print(nombre)           # Output: Juan
print(edad)             # Output: 20
print(altura)           # Output: 1.75
print(es_estudiante)    # Output: True
```

Reglas para Nombrar Variables

No puede comenzar con un número: Debe comenzar con una letra o un guión bajo (_).

No puede contener espacios: Se utilizan guiones bajos (_) para separar palabras (por ejemplo, mi_variable).

No puede ser una palabra reservada: Palabras clave del lenguaje no pueden ser utilizadas como nombres de variables (por ejemplo, for, if, while).

Actividad 2

Crear tres variables para almacenar su nombre, apellido y edad.

Imprimir en pantalla los contenidos de las variables.

Solución a actividad 2

```
nombre = "Nombre del Alumno"
apellido = "Apellido del alumno"
edad = 10


print(nombre)
print(apellido)
print(edad)
```

¿ Qué es una variable auxiliar ?

Una variable auxiliar es una variable que se utiliza temporalmente para ayudar a realizar una tarea específica en un programa, a menudo para facilitar cálculos, almacenamiento temporal de valores, o intercambio de datos entre variables. Las variables auxiliares son especialmente útiles para mejorar la legibilidad y mantenimiento del código, así como para evitar errores.

Ejemplos de Uso de Variables Auxiliares

Intercambio de Valores entre Dos Variables

Acumuladores y Contadores

Almacenar Resultados Temporales

Mejorar la Legibilidad del Código

Intercambio de Valores entre Dos Variables

```
# Usando una variable auxiliar

a = 5

b = 10

aux = a

a = b

b = aux


print(a)  # Output: 10

print(b)  # Output: 5
```

Actividad 3

Dadas las variables cociente=30 y divisor=60 , realizar un intercambio de variables mediante una variable auxiliar y mostrar el resultado.

Solución actividad 3

```
cociente=30

divisor=60

aux=cociente

cociente=divisor

divisor=aux

print(cociente/divisor)
```

Constantes

En Python, no existe una palabra clave específica para definir constantes, como en otros lenguajes de programación. Sin embargo, se sigue una convención de nombres para definir constantes, que consiste en usar nombres de variables en mayúsculas.

PI = 3.14159

Tipo de dato

Python es un lenguaje de tipado dinámico, lo que significa que el tipo de una variable se determina automáticamente en tiempo de ejecución según el valor que se le asigna.

Ejemplo:

```
# No se declara el tipo, simplemente se asigna un valor

edad = 25  # Python interpreta que es un entero (int)

nombre = "Adrian"  # Python interpreta que es una cadena (str)
```

Tipo de dato

Los tipos básicos incluyen números (int, float, complex), texto (str), booleanos (bool), secuencias (list, tuple, set, dict) y el tipo None.

Se puede verificar el tipo de cualquier variable usando la función type().

Ejemplo:

```
nombre = "Alumno"

print(type(nombre))
```

Tipado fuerte en Python

En un lenguaje de tipado fuerte como Python:

Cada valor tiene un tipo de dato específico (por ejemplo, str para cadenas, int para enteros).

Las operaciones entre tipos incompatibles no están permitidas sin una conversión explícita.

Python no realiza conversiones automáticas entre tipos de datos de manera implícita para operaciones como la concatenación, suma, etc.

Tipado dinámico vs tipado estático

Python tiene tipado dinámico, ejemplo:

y = 10

print(type(y))

Salida: <class 'int'>

palabra = “hola”

print(type(palabra))

Salida: <class 'str'>

Lenguajes como java o c , son de tipado estático, porque hay que declarar el tipo de dato.

Operadores aritméticos

Suma +

Resta -

Multiplicación *

Potenciación **

División real /

División entera //

Módulo o Resto %

Expresiones aritméticas

Son combinaciones de variables y operadores que producen un valor.

Ejemplo

x = 5 + 6

Expresiones Lógicas

Solo pueden tomar dos valores, verdadero o falso.
Se forman combinando constantes lógicas, variables lógicas, utilizando los operadores not, and y or y los operadores relacionales (==,>,<,<=,>=,<>).

Ejemplo:

```
c = 10
d = 10
resultado = c > d
print(resultado)    # imprime False
```

Actividad 4

Indicar con True o False, si un número es par

Indicar con True o False, si un número es divisible por 3

Un producto tiene un precio por kilo de $8500, cuanto valen 300 gramos ?

Solución a actividad 4

Ejercicio 1

```
c = 12

resultado = c%2==0

print(resultado)
```

Solución a actividad 4

Ejercicio 2

```
c = 12

resultado = c%3==0

print(resultado)
```

Solución a actividad 4

Ejercicio 3

cantidad=1000

precio=8500

cantidadVenta=300

precioVenta=cantidadVenta*precio//cantidad

print(precioVenta)

Orden de evaluación

El orden de evaluación en algoritmia se refiere a la secuencia en la que se evalúan las subexpresiones dentro de una expresión compuesta. En otras palabras, cuando tienes una expresión que incluye múltiples operaciones o funciones, el orden de evaluación determina qué partes de la expresión se calculan primero.

Ejemplo simple: Considera la siguiente expresión en pseudocódigo:

z = (a+b) / (c-d)

Aquí, hay dos operaciones principales: la suma y la resta dentro de los paréntesis, y la multiplicación. El orden de evaluación generalmente sigue las reglas de precedencia de operadores, lo que implica que primero se evalúan las operaciones dentro de los paréntesis (suma y resta) antes de realizar la multiplicación.

Resumen del orden de evaluación en Python

Prioridad de operadores: Los operadores aritméticos y lógicos tienen diferentes niveles de prioridad. Por ejemplo, la multiplicación (*) y la división (/) tienen mayor prioridad que la suma (+) y la resta (-). Los operadores de comparación se evalúan después de los aritméticos.

Asociatividad: Cuando dos operadores tienen la misma prioridad, se evalúan de izquierda a derecha (asociatividad izquierda). Por ejemplo, en la expresión a - b + c, se evalúa primero a - b y luego se suma c.

Paréntesis: Las expresiones dentro de paréntesis se evalúan primero, lo que permite cambiar el orden de evaluación natural. Por ejemplo, en 2 * (3 + 4), se evalúa primero (3 + 4).

Ingreso de datos en Python

En Python se puede tomar datos del teclado utilizando la función input(). Esta función permite al usuario ingresar datos como texto. Ejemplo:

```
# Solicitar al usuario que ingrese su nombre
nombre = input("¿Cuál es tu nombre? ")

# Mostrar un saludo
print("Hola, " + nombre + "!")
```

Notas sobre el ingreso de datos en Python

Tipo de dato: Los datos ingresados por input() se toman como una cadena (string). Si necesitas un tipo de dato diferente, como un entero o un flotante, debes convertirlo usando int() o float(), respectivamente. Ejemplo:

```
# Solicitar al usuario que ingrese su edad y convertirla a entero

edad = int(input("¿Cuál es tu edad? "))

print("Tienes " + str(edad) + " años.")
```

Ejercicio 1: Calcular el área de un rectángulo

Descripción: El programa pedirá al usuario que ingrese la base y la altura de un rectángulo, y luego calculará y mostrará el área.

Ejercicio 2: Convertir grados Celsius a Fahrenheit

Descripción: El programa pedirá al usuario que ingrese una temperatura en grados Celsius y luego la convertirá a grados Fahrenheit.

La fórmula de conversión: F = 9/5 C + 32

Donde c es la temperatura en celcius

Ejercicio 3: Calcular el perímetro de un triángulo

Descripción: El programa pedirá al usuario que ingrese las longitudes de los tres lados de un triángulo y luego calculará y mostrará el perímetro del triángulo.

Ejercicio 4: Calcular la distancia entre dos puntos en un plano

Descripción: El programa pedirá al usuario que ingrese las coordenadas x1, y1 del primer punto y x2 y2 del segundo punto, y luego calculará y mostrará la distancia entre los dos puntos utilizando la fórmula de distancia entre dos puntos en un plano cartesiano.

Fórmula distancia: raiz cuadrada de ((x2-x1)^2 + (y2-y1)^2)

En python, la función de raíz cuadrada es: math.sqrt()