|  |  |  |
| --- | --- | --- |
| | GUÍA 2.1.3:Estructuras Base | | |
| Sigla | Asignatura | Experiencia de Aprendizaje |
| FPY1101 | Fundamentos de Programación | EA2: Programación de aplicaciones en Python |
| Tiempo | Modalidad de Trabajo | Indicadores de logro |
| 2 h | Individual/parejas o grupal | IL2.1 |

|  |
| --- |
| **Código QR con relleno sólido**  **Antecedentes generales** |

## Esta guía tiene como objetivo conocer los aspectos generales en Python

|  |
| --- |
| **Lista con relleno sólido Requerimientos para esta actividad** |

## Para el desarrollo de esta actividad deberás disponer de:

## Computador

## Visual Studio Code

|  |  |
| --- | --- |
| **Inteligencia artificial con relleno sólidoActividad** |  |

Esta actividad consiste en conocer las estructuras base utilizadas en Python, al revisar los comandos, verás que hay muchos que por ahora no se utilizarán, pero a medida que avanzas con la materia de la asignatura, comprenderás el significado.

1. Crea un archivo .py en Visual Studio y copia todas las sentencias que siguen a continuación:

# Este es un programa de ejemplo en Python.

# Para ejecutar una parte de este programa, deberás comentar las secciones que no probarás

# Variables y Tipos de Datos.

# Aquí, se han creado variables que almacenan un número entero,

# un número decimal, una cadena de texto y un valor booleano.

variable\_entera = 42 # Entero

variable\_decimal = 3.14 # Decimal

variable\_texto = "Hola, mundo!" # Cadena de texto

variable\_booleana = True # Valor booleano

# Operadores

# En este caso, resultado\_suma almacena el resultado de

# sumar variable\_entera y variable\_decimal, y comparacion

# almacena el resultado de comparar si variable\_entera es

# mayor que variable\_decimal.

resultado\_suma = variable\_entera + variable\_decimal

comparacion = (variable\_entera > variable\_decimal)

# Entrada y Salida

nombre\_usuario = input("Por favor, ingresa tu nombre: ")

print("Hola, " + nombre\_usuario + "! Este es tu primer programa en Python.")

# Estructuras de Control de decisiones

# Se utiliza una estructura if-elif-else para tomar decisiones basadas en condiciones

if variable\_booleana:

print("La variable booleana es verdadera.")

elif resultado\_suma < 10:

print("La suma es menor que 10.")

else:

print("Ninguna de las condiciones anteriores se cumple.")

# Colecciones de Datos

# Se crean listas, tuplas, diccionarios y conjuntos para almacenar datos

lista\_numeros = [1, 2, 3, 4, 5]

tupla\_colores = ("rojo", "verde", "azul")

diccionario\_edades = {"Juan": 25, "María": 30, "Carlos": 22}

conjunto\_elementos = {1, 2, 3, 4, 5}

# Funciones

# Se define una función saludar() y se la utiliza para crear un

# mensaje de saludo personalizado

def saludar(nombre):

return "¡Hola, " + nombre + "!"

mensaje\_saludo = saludar("Estudiante")

# Manejo de Errores

try:

resultado\_division = 0 / 0

except ZeroDivisionError:

print("¡Error! No se puede dividir por cero.")

finally:

print("Bloque 'finally': Este código se ejecuta siempre, haya o no haya errores.")

# Trabajo con Archivos

with open("archivo.txt", "w") as archivo:

archivo.write("¡Hola desde un archivo!")

# Módulos y Bibliotecas

import math

raiz\_cuadrada = math.sqrt(16)

# Programación Orientada a Objetos

class MiClase:

def \_\_init\_\_(self, atributo):

self.atributo = atributo

def mostrar\_atributo(self):

print("El valor del atributo es:", self.atributo)

# Manejo de Cadenas (Strings)

longitud\_cadena = len("Python")

mayusculas = "hola".upper()

minusculas = "HOLA".lower()

reemplazo = "python es divertido".replace("divertido", "increíble")