INGENIERIA EN COMPUTACION

Materia: Ingeniería en computación

Alumno: Ayon Dimas Andres

Matricula: 366845

Profesor: Pedro Nunes Yepiz

Tema: Ciclos, try, random

Ensenada B.C 18 de septiembre del 2023

INTRODUCCION

Aquí podremos observar la manera de utilizar los ciclos en Python, ya que se utilizan algo distinto de cómo son en C#. En Python los ciclos son un tanto más cortos y este mismo hace casi todo de manera automática, también veremos lo que es la sentencia try que sirve para evitar que un programa de error al momento de insertar un dato que no sea del tipo que están solicitando al usuario y evitar que este de error. Así mismo veremos lo que es en rondom, que nos permite generar cosas de manera aleatoria, al igual veremos funciones en Python de una manera breve.

COMPETENCIA

Dominar las distintas sentencias presentadas en esta actividad, así mismo el uso que se les debe de dar, como utilizarlas y así mismo el cuándo es adecuado utilizar este tipo de sentencias, en que momento y en que programa nos podría facilitar el trabajo a la hora de realizar nuestro programa en Python.

FUNDAMENTOS

Ciclo for

Junto a la rama if else, el bucle for es probablemente la estructura de programación más conocida. Generaciones de estudiantes se han exprimido el cerebro con el concepto informático del bucle. Esto se debe a que, para empezar, **los bucles no son muy intuitivos**. Sin embargo, los problemas de interpretación pueden surgir más bien de la forma en la que se presenta el material. Porque, si los analizamos desde un punto de vista general, los bucles no son nada fuera de lo común.

Random

El módulo **random** de la librería estándar de Python incluye un conjunto de funciones que permiten obtener de distintos modos números aleatorios o, para ser rigurosos, pseudoaleatorios.

¿Por qué números pseudoaleatorios y no aleatorios?

Las funciones a las que hacemos referencia, cada vez que son invocadas, devuelven un valor de una secuencia de números predeterminada. Esta secuencia tiene un periodo

bastante largo, es decir, es necesario obtener muchos números antes de que se vuelva a reproducir la misma secuencia. De ahí, el tratamiento de pseudo (o falso), aunque la utilidad que nos ofrezca sea equivalente al de los números aleatorios.

Try

En Python, puedes usar los bloques try y except para manejar estos errores como *excepciones.*
En este tutorial aprenderás la sintaxis básica de try y except. Después escribirás ejemplos simples, detectaras que puede salir mal y brindaras soluciones correctivas usando los bloques try y except.

PROCEDIMIENTO

```
import random


# Definimos una función para generar un número aleatorio entre 1 y 100.
def generar_numero():
    return random.randint(1, 100)


# Definimos una función que recibe un número y permite al usuario adivinar en un número limitado de intentos.
def adivinar_numero(numero_secreto):
    intentos_maximos = 7  # Número máximo de intentos permitidos
    for intento in range(1, intentos_maximos + 1):
        try:
            guess = int(input(f"Intento {intento}/{intentos_maximos}: Adivina el número entre 1 y 100: "))


            if guess < 1 or guess > 100:
                raise ValueError("El número debe estar entre 1 y 100.")
```

```
            if guess < numero_secreto:
                print("El número es mayor.")
            elif guess > numero_secreto:
                print("El número es menor.")
            else:
                print(f"¡Correcto! Adivinaste el número en {intento} intentos.")
                return

        except ValueError as e:
            print(f"Error: {e}. Por favor, ingresa un número válido.")

    print(f"¡Lo siento! Te quedaste sin intentos. El número secreto era {numero_secreto}.")

# Función principal del juego
def main():
    print("¡Bienvenido al juego de adivinanza de números!")
    numero_secreto = generar_numero()
    adivinar_numero(numero_secreto)

if __name__ == "__main__":
    main()
```

RESULTADOS Y CONCLUSIONES

Como pudimos apreciar, el utilizar los ciclos en Python requiere de menos trabajo que lo requiere C# ya que Python hace muchas cosas por nosotros al momento de agregar estas sentencias como lo son el for, try except, random y también las funciones en python.

Esto nos facilita mucho nuestro programa ya que no necesitamos escribir muchas veces una cosa para preguntar a l usuario ya que con la sentencia **for** podremos repetirlo las veces que deseemos y de esta manera tener un flujo controlado por el usuario, donde el puede llegar a determinar lo que quiere hacer con este ciclo **for**. Al igual las funciones nos son muy útiles, ya que podemos mandar llamarlas para que se ejecute determinada parte del programa y tener un control sobre este mismo. Son herramientas fundamentales en python que no podemos pasar por alto ya que engloban las sentencias principales de este mismo y con ellas podremos realiza programas útiles y eficaces.