Objetivos

- Preparar el entorno de trabajo que utilizarán a lo largo de la materia instalando las herramientas básicas.
- Aprender a escribir nuestro primer programa con **Python** y ejecutarlo.
- Crear nuestro primer repositorio local de código y sincronizar el mismo a un repositorio remoto (GitHub).

Introducción

Python

Parte principal de esta actividad es tener su entorno de trabajo preparado para trabajar durante la cursada. Por este motivo es fundamental que comenzemos por instalar `python` en su máquina. Este año utilizaremos la versión de **Python 3.12.9**. Para instalar **Python** en su máquina les proveemos esta guía de instalación.

Git

Como ya vimos **Git** es una herramienta muy buena que nos permite manejar versiones de nuestro código de manera distribuida con nuestro equipo de trabajo.

Para poder realizar esto es necesario contar con un **servidor** de **Git** además de tener la herramienta instalada en su máquina **local**. En esta oportunidad vamos a analizar un poco GitHub que es el **servidor** de **Git** más popular actualmente.

Recuerde que se encuentra publicada la guía básica de Git donde se realizan varias de las tareas que solicitamos para realizar esta entrega.

Actividades introductorias

1. Desarrolla un programa que solicite al usuario que ingrese su edad y luego calcule y muestre cuántos años le faltan para alcanzar los 100 años.
2. Haz un programa que pida al usuario que ingrese una temperatura en grados Celsius y luego convierta esa temperatura a grados Fahrenheit, mostrando el resultado.
3. Crea un programa que calcule la suma de los primeros 100 números naturales utilizando un bucle for.
4. Cree un programa que dada una lista de números imprima sólo los que son pares. Nota: utilice la sentencia `continue` donde haga falta.
5. Implementa un programa que solicite al usuario que ingrese una lista de números. Luego, imprime la lista pero detén la impresión si encuentras un número negativo. Nota: utilice la sentencia `break` cuando haga falta.

6. Modifique el ejercicio 4 para que dada la lista de número genere dos nuevas listas, una con los número pares y otras con los que son impares. Imprima las listas al terminar de procesarlas.
7. Escribe un programa que tome una lista de números enteros como entrada del usuario. Luego, convierte cada número en la lista a string y únelos en una sola cadena, separados por guiones ('-'). Sin embargo, excluye cualquier número que sea múltiplo de 3 de la cadena final.

Entrega

1. Realizar la instalación de Python **3.12.X**.
2. Instale **git** en su sistema operativo.
3. Escriba en un archivo llamado `questions.py` el siguiente programa en Python.

```python
import random

# Preguntas para el juego
questions = [
    "¿Qué función se usa para obtener la longitud de una
cadena en Python?",
    "¿Cuál de las siguientes opciones es un número entero en
Python?",
    "¿Cómo se solicita entrada del usuario en Python?",
    "¿Cuál de las siguientes expresiones es un comentario
válido en Python?",
    "¿Cuál es el operador de comparación para verificar si
dos valores son iguales?",
]
# Respuestas posibles para cada pregunta, en el mismo orden
que las preguntas
answers = [
    ("size()", "len()", "length()", "count()"),
    ("3.14", "'42'", "10", "True"),
    ("input()", "scan()", "read()", "ask()"),
    (
        "// Esto es un comentario",
        "/* Esto es un comentario */",
        "-- Esto es un comentario",
        "# Esto es un comentario",
    ),
    ("=", "==", "!=", "==="),
]
# Índice de la respuesta correcta para cada pregunta, el el
mismo orden que las preguntas
correct_answers_index = [1, 2, 0, 3, 1]
```

```
# El usuario deberá contestar 3 preguntas
for _ in range(3):
    # Se selecciona una pregunta aleatoria
    question_index = random.randint(0, len(questions) - 1)

    # Se muestra la pregunta y las respuestas posibles
    print(questions[question_index])
    for i, answer in enumerate(answers[question_index]):
        print(f"{i + 1}. {answer}")

    # El usuario tiene 2 intentos para responder
correctamente
    for intento in range(2):
        user_answer = int(input("Respuesta: ")) - 1
        # Se verifica si la respuesta es correcta
        if user_answer ==
correct_answers_index[question_index]:
            print("¡Correcto!")
            break
    else:
        # Si el usuario no responde correctamente después de
2 intentos,
        # se muestra la respuesta correcta
        print("Incorrecto. La respuesta correcta es:")
        print(answers[question_index]
[correct_answers_index[question_index]])

    # Se imprime un blanco al final de la pregunta
    print()
```

4. Comience un repositorio **local** y agregue el archivo recientemente creado.
5. Crea tu propio repositorio **remoto** en Github y suba el archivo al repositorio remoto.
6. Agrega el `README.md` con tu nombre y número de estudiante.
7. Modifique el programa anterior con las siguientes funcionalidades:
    - El juego tiene un bug. Si el usuario ingresa un número de respuesta no válido por ejemplo 42 o "huevos con spam" el programa falla con un error. Modifica el mismo para que si la respuesta no es un número o bien no está en el rango de las respuestas posibles muestre un mensaje diciendo: "Respuesta no válida" y termine de inmediato con exit status igual a 1.
    - Modifique el juego para que al final de la partida se muestre el puntaje del jugador o jugadora. El puntaje se calcula de la siguiente forma, cada intento fallido descuenta 0.5 puntos y cada acierto suma 1 punto.

- Modifique el juego para, en lo posible, no acceder a las 3 listas usando índices. Ayuda:

```
questions_to_ask = random.choices(list(zip(questions,
answers, correct_answers_index)), k=3)
```

- Modifique el juego para que no muestre preguntas repetidas (investigue la función `random.sample()`)

Nota: Por cada funcionalidad agregada se debe realizar al menos un commit que identifique el cambio.

Pautas

- **Puntos:** 10.
- **Fecha límite de entrega:** Viernes, 21 de marzo de 2025, 23:59
- **Modalidad de entrega:** Copie el enlace de su repositorio remoto con la resolución en la tarea de Cátedras.

Actividad Extra

Creación de un programa básico de gestión de inventario.

Desarrollar un programa en Python que permita gestionar un inventario simple de productos, incluyendo funciones básicas como agregar productos, eliminar productos y mostrar el inventario.

El programa debe tener un menú interactivo que permita al usuario seleccionar las siguientes operaciones:

- Agregar un nuevo producto al inventario, solicitando al usuario el nombre y la cantidad inicial del producto.
- Eliminar un producto existente del inventario, solicitando al usuario el nombre del producto a eliminar.
- Mostrar el inventario actual, que incluya el nombre y la cantidad de cada producto.
- Salir del programa.

El inventario puede ser almacenado utilizando un diccionario simple, donde las claves sean los nombres de los productos y los valores sean las cantidades. Se deben manejar situaciones simples como la introducción de productos duplicados o la eliminación de productos inexistentes.