ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
CCPG1043 / CCPG1801 - FUNDAMENTOS DE PROGRAMACIÓN
PRIMERA EVALUACIÓN
I TÉRMINO 2025-2026/ Julio 4, 2025

Compromiso de Honor

Yo, ..., matrícula ..... del paralelo ..... de Fundamentos de Programación, declaro que he sido informado y conozco las normas disciplinarias que rigen a la ESPOL, en particular el Código de Ética y el Reglamento de Disciplina. Al aceptar este compromiso de honor, asumo la responsabilidad de realizar este examen de manera honesta, sin recurrir a prácticas de plagio, fraude o deshonestidad académica por medios físicos o electrónicos. Me comprometo a responder las preguntas con mis propias habilidades y conocimientos, sin recibir ayuda no autorizada. Además, garantizo que respetaré los derechos de propiedad intelectual y no divulgaré ni copiaré el contenido del examen.

Acepto el presente compromiso, como constancia de haber leído y aceptado la declaración anterior y me comprometo a seguir fielmente las directrices que se indican para la realización de la presente evaluación. Estoy consciente que el incumplimiento del presente compromiso anulará automáticamente mi evaluación y podría ser objeto del inicio de un proceso disciplinario.

Firma compromiso: .....

Importante: El examen debe resolverse únicamente en las **hojas de trabajo**, siguiendo el orden establecido y utilizando una carilla para cada tema.
No está permitido escribir, marcar, dibujar, resaltar ni hacer anotaciones de ningún tipo en las **hojas impresas** con los temas del examen.

Tema 1	Tema 2	Tema 3	Team 4	Total	
					Firma Revisé mi calificación

Funciones y propiedades de referencia en Python.

random as ***rd***:	para ***listas***:	para ***cadenas***:
rd.randint(inicio,fin) ***rd***.randrange(inicio,fin, salto) ***rd***.choice(lista) ***rd***.choices(lista,k=cant) ***rd***.sample(lista,cant) ***rd***.shuffle(lista)	***lista***.append(...) ***lista***.extend(...) ***lista***.count(...) ***lista***.index(...) ***lista***.pop() ***lista***.insert(pos, elem) *elemento* in ***lista*** *range(**n**)* ***separador***.join(***lista***)	***cadena***.islower() ***cadena***.isupper() ***cadena***.isdigit() ***cadena***.isalpha() ***cadena***.lower() ***cadena***.upper() ***cadena***.split(x) ***cadena***.capitalize() ***cadena***.count(x) ***cadena***.replace(a,b)

TEMA 1

[25 puntos] Crea una función llamada **devolver_numero(dato)** que reciba una cadena de caracteres llamada *dato*. La función debe devolver el dato convertido en número flotante si la cadena tiene un formato numérico, caso contrario debe devolver 0.0.

Nota: El punto es el único carácter especial permitido y representa la separación decimal.

```
# Ejemplo:
"5.6" devuelve 5.6
"78" devuelve 78.0
"76.4.4" devuelve 0.0
"h.1" devuelve 0.0
```

TEMA 2

[30 puntos] Implementa la función **monto_categoria(lista_proyectos, categoria)** para calcular el monto total de los proyectos asociados a una categoría específica.

Cada elemento de **l_proyectos** tiene el siguiente formato:

```
Categoría;proyecto1:valor;proyecto2:valor;...
```

Ejemplo de la lista de proyectos por categoría:

```
l_proyectos = [
  "Ciencias Naturales;Monitor cardíaco:230;Primeros auxilios:210",
  "Tecnología;Clasificador de imágenes:250;Chatbot para Call Center:220;...",
  "Arte y Creatividad;App para matemáticas:180;Juego de lógica para niños:160;..."
]
```

La función debe:

- Buscar la categoría especificada (**sin distinguir mayúsculas/minúsculas**).
- Sumar los montos de todos los proyectos de esa categoría.
- Retornar el total.

Ejemplo:

```
# Para la lista de proyectos del ejemplo y la categoría buscada
cat = "CIencias naturalEs"

# Salida esperada:
440.00
```

TEMA 3

[25 puntos] Implemente un programa que simule una carrera de 10 turnos entre dos animales: **"Liebre"** y **"Tortuga"**. Cada animal tiene una velocidad máxima guardada en **vel_l** y **vel_t** respectivamente, que indica cuánto puede avanzar por turno, usando un número aleatorio entre 1 y la velocidad máxima, ambos incluidos.

Además, hay una lista llamada **l_trampas** que indica las casillas donde hay trampas para la **liebre**. Si cae en una, regresa a la posición anterior (donde estaba antes del último turno).

El avance de cada animal se acumula turno a turno, sumando las casillas recorridas en cada movimiento para determinar su posición final en la pista de la carrera. Además, debe mostrar con guiones el avance alcanzado como se muestra en el ejemplo de salida esperada.

Al final de los 10 turnos, gana el animal que haya avanzado más casillas en total. En caso de empate, se declara empate.

```
# Datos iniciales:
vel_l = 5 # liebre
vel_t = 3 # tortuga
l_trampas= [4, 7, 9, 13]
```

Salida esperada:

```
Turno 1:
Liebre : --- (posición 3)
Tortuga: -- (posición 2)
Turno 2:
Liebre cayó en trampa en casilla 4, regresa a su posición anterior
Liebre : --- (posición 3)
Tortuga: ----- (posición 5)
... otros turnos ...
Turno 10:
Liebre : --------- (posición 9)
Tortuga: -------------- (posición 14)

Ganador: Tortuga
```

TEMA 4

[20 puntos] Se requiere un programa que procese una lista con mediciones de temperatura de un dispositivo. Algunas mediciones contienen el valor "ERR", indicando un error de lectura. El objetivo del programa es:

1. Contar cuántos valores "ERR" hay en la lista.
2. Calcular la suma de las temperaturas.
3. Contar las mediciones válidas (numéricas).

Sin embargo, el código proporcionado contiene **cinco errores** que **no permiten ejecutar el programa**. **Reescriba todo el código** para solucionar los problemas.

Advertencia: No se deben introducir nuevos errores en el código reescrito. Cada error adicional será penalizado con 4 puntos.

```
def analizar_temperaturas(lista_temp):
    errores = 0
    suma = 0
    cantidad_validas = 0

    for t in lista_temp:
        if t = "ERR":
            errores =+ 1
        else:
            suma += t
            cantidad_validas += 1

    return errores, suma, cantidad_validas

datos = ["22.5", "ERR", "24", "21.8", "ERR", "23.3"]
e, s = analizar_temperaturas(datos)
print("Errores encontrados:" , E)
print("Suma de temperaturas válidas:", round(s, 2))
```