**Python Módulo I – Examen final Parte práctica**

Se tomará en cuenta las **buenas prácticas**, **las tabulaciones o sangrados respectivos** y así como el uso de **flake8** para el adecuado diseño de código de nuestros problemas

* + 1. Escriba un programa donde tendrá los siguientes requisitos (**4 ptos**): **Reglas:** • Clase base Persona o Atributos: nombre, edad, nacionalidad="peruana", saldo=0.0.
  + o Métodos para esta clase: ▪ actualizar\_nombre(nombre) y actualizar\_edad(edad) (validar edad > 0).
  + ▪ cumplir\_anios() (incrementa edad en 1).
  + ▪ mostrar\_saldo() (imprime el saldo actual).
  + ▪ transferir(destino, monto) (si no hay fondos suficientes, imprimir “Saldo insuficiente”; si hay, basarse en self y acreditar a destino).
  + • Crear la clase que hereda: Empleado(Persona) o Atributo adicional: sueldo (float).
  + o Métodos para esta clase: ▪ aumento\_sueldo(porcentaje=0.30) (incrementa el sueldo en 30% por defecto).
  + ▪ prediccion(anio\_objetivo, edad\_param=None) ▪ Retorna el mensaje: “En el año XXXX tendrá XX años”.
  + ▪ Si edad\_param se pasa y es menor que self.edad, imprimir “No es posible realizar la operación” y no calcular.
  + • Pruebas mínimas o Instanciar al menos dos Empleado.
  + o Aplicar aumento\_sueldo 2 veces y mostrar el sueldo incrementado.
  + o Realizar una transferencia entre ambos empleados y mostrar saldos antes y después de ambos
  1. o Probar un caso de saldo insuficiente.
  2. o Usar prediccion(...) con y sin edad\_param inválido.
  3. Utilizar el concepto de **módulo** necesariamente. Y escribir un programa para el manejo de listas el cuál cumplirá los siguientes requerimientos (**2 ptos**): Crear una función que le permitirá almacenar X números aleatorios en una lista y finalmente los imprimirá por consola al llamar la función. X solo puede ser entero. No otro tipo de dato. Y también un índice existente en la lista (usar para esto excepciones)
  4. Crear una función que le permita almacenar los números **no** repetidos de la lista anterior, la función retornará este valor para que pueda ser impreso por consola.
  5. Crear una función donde se creará una lista para ordenar de mayor a menor la lista que se creó en el ítem anterior (número no repetidos) y otra lista para ordenarlas de menor a mayor, retornar este valor e imprimirlos por consola.
  6. Crear una función para indicar cuál es el mayor número par de la lista (lista de la regla 2), retornar este valor e imprimirlo por consola.
  7. Crear el archivo main.py, donde solo llamarás las anteriores funciones que se encontrarán alojadas en un módulo (probarlo para dos casos mínimo)

**Reglas:**

* 1. (**2 ptos**) Crear un decorador conteo. El decorador retornará la cantidad de parámetros que hayas usado en la función y que a su vez evaluará que deba ser mayor que 1 para procesar esta lógica, caso contrario indicarlo con un mensaje respectivamente al usuario.
  2. Al final de la función decorada indicará mediante un mensaje que la función fue ejecutada exitosamente.
  3. La función que vas a crear va a capturar, **la edad**, **nombre de un alumno**, **la hora** y **el minuto** en que fue registrado (usar la librería correspondiente de tiempo)
  4. La función que será decorada también estará pasando 4 notas que calculará la media del estudiante.

**Reglas:**

Mostrando un mensaje siguiente: “Pedro de 30 años ha sido registrado a las 16 horas con 20 minutos”

* 1. (**2 ptos**) Crear un programa usando decoradores para mostrar solo la hora y el minuto del momento que se usa el decorador Al ejecutar el decorador mostrará un mensaje: “El decorador está siendo Crear la función decorador adecuadamente que sumará los elementos de la función que pasará como parámetro de la función decoradora
  2. Crear una función**, por ejemplo:** usando 6 números e indicar el mayor de todos ellos (o x números) para decorarla con la función anterior.
  3. Usar la propiedad de N parámetros para la función a decorar usando sus **key y values** (\*\*kwards) para usar más de 6 valores en la función (value\_7 = 10, value\_8 = 22, value\_9=45) y visualizar los resultados usando el decorador implementado con un mínimo tres ejemplos.

**Reglas:**

ejecutado a las **H** con **m** minutos y S segundos”

1. Agregar a tu **repositorio de exámenes** tu carpeta con el nombre de **“Examen final”** con tus soluciones y realizar un **push** con todas tus soluciones de tus soluciones resueltas (**2 ptos.)**

**Indicaciones:**

* Tiempo total del examen: 110 minutos
* Realizar los problemas que puedas resolver, lo importante es aplicar lo primero que sepas más.
* Cada solución estará dentro de un archivo .py (Ejem.; ejercicio\_final\_01.py) y con la propuesta del problema en la parte superior como comentario.
* Adjuntar al correo las soluciones al archivo comprimido:

**examen\_final\_nombre\_apellido.rar** (.zip)

* Enviar el link del repositorio que has creado en github respectivamente y en cuál estarán todas sus soluciones al siguiente correo:

docente.cerseu.unmsm@gmail.com

* Soluciones entregadas con alguna herramienta de IA se le penaliza sobre todo el examen con **-8 ptos**
* Asunto: Parte práctica – Examen final