Universidad Nacional Mayor de San Marcos
Universidad del Perú, Decana de América

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
PROGRAMA: ESPECIALIZACION EN PYTHON

MÓDULO I

Reforzamiento N°05

Importante: Subir todas sus soluciones a su repositorio donde se encuentra las soluciones de los anteriores reforzamientos.

Clases:

1. Crear una clase **Empleado** que contenga los siguientes métodos, uno que pida ingresar un nombre, apellido y edad, un método para pedir su sueldo actual y otro método que lo imprima ambos resultados, pero estarán contenidos en un diccionario. Comprobar los métodos instanciado la clase respectivamente al menos para 3 empleados. Considerar en el constructor los valores necesarios.

2. Crear una clase llamada **Circulo** que contenga radio en su constructor y que contenga un método área que devuelva el área de un círculo.

Adicionalmente habrá un método que devuelva el perímetro del círculo. También habrá un método donde pedirá el radio del círculo.

Instanciar mínimo 2 veces la clase y mostrar resultados por consola.

4. Crear una clase **Alumno** que tenga como atributos el nombre, edad y la nota final del alumno. Crear los métodos para inicializar sus atributos, otro para imprimirlos y un método para mostrar un mensaje con el resultado de la nota, si ha aprobado (mayor o igual a 13) o no el alumno. Instanciar la clase por lo menos 4 veces (4 alumnos)

5. Crear una clase **Persona** con los siguientes requerimientos.

La clase tendrá como atributos el nombre, edad y sueldo de una persona. Implementar los métodos necesarios para inicializar los atributos (constructor), un método para mostrar los datos e indicar si la persona es mayor de edad o no y otro método que bonificación que retornará el 20% adicional de su sueldo, en caso sea mayor de edad. Un método para saber cuántos meses ha estado trabajando en la empresa

Instanciar por lo menos la clase con 3 diferentes personas.

6. Crear una clase **PersonaPrestamo** que hereda de la clase anterior (problema 5) donde tendrá los métodos: Uno, que indicará si la persona está apta para un préstamo solo si ha estado más de 12 meses trabajando en la empresa en caso contrario se le informa que no es posible darle el préstamo y la otra condición adicional es si su edad es mayor de 25 años.

Agregar un segundo método a esta nueva clase que te indicará si estás aprobado recibirás 10 veces tu sueldo de préstamo, el total de préstamo otorgado es {cantidad_prestamo}.

Instanciar 3 veces la clase para mostrar diferentes resultados.

7. Desarrolla una clase **Agenda** que administrará contactos. Dentro de una lista se debe almacenar un diccionario para cada contacto el nombre, el teléfono, email y DNI. Deberá tener los siguientes métodos:

Añadir contacto, Mostrar contactos y Buscar contacto (Por DNI)

Estructura de la lista cada vez que vas agregando un contacto:

contactos = [

{'nombre': "Luis", 'telefono': "997667945", 'email': "luishh@gmail.com", 'dni': 44234239},

{'nombre': "Milagros", 'telefono': "997654687", 'email': "milagros19c@gmail.com", 'dni': 43423211}

]

Agregar por lo menos 4 personas (instanciándolas) para poder luego realizar la búsqueda de estas personas en la agenda y saber si existen, en caso contrario indicar: "´Nombre´ no se encuentra anotado en la agenda"

8. Crear una clase **Persona** que contenga dos atributos: nombre y edad. Nombre y edad se ingresarán por teclado en el constructor.

Declarar una segunda clase llamada **Empleado** que herede de la clase **Persona** y agregue un atributo sueldo. Crearás un método que indicará si debe pagar impuestos (que sería el 10% de su sueldo y si sueldo es superior a 4000 soles)

Instanciar la clase Empleado al menos para 3 empleados, mostrar el sueldo del empleado y cuánto debe pagar de impuesto.

9. Escribir una clase **Figura** que debe tener un atributo de nombre de la figura.

Habrá otra clase llamada **Rectangulo** que hereda de **Figura**. Pedirá una base y una altura en sus parámetros. Tendrá un método que devuelva el área y perímetro de este rectángulo.

También habrá otra clase llamada **Circulo** que hereda también de **Figura,** pedirá por parámetro el radio y este tendrá los métodos que calculará el área y otro método que calculará el perímetro del **círculo**

Realizar la instancia de la clase **figura** mínimo **4 veces** para mostrar el uso en ambas figuras (2 casos para ambas figuras) y resultados de todos los métodos mediante consola.

10. Crear una clase llamada **Soldado** para un juego sobre un mapa la cual deberá tener como atributos en el constructor posición X y posición Y las cuales iniciarán en 0, agregar un nombre a este soldado dentro de estos atributos.

La clase debe tener los siguientes métodos. Caminar hacia el eje X en donde se indicará cuántos pasos dará y otra clase que le permitirá caminar por el eje Y, en caso se indique un número negativo indicar que solo puede llegar hasta el 0 si es menor a los pasos ya dados.

Crear finalmente un método adicional que irá guardando los pasos que ha dado dentro de una lista para que finalmente al usar este método me muestre el historial de movimientos del Soldado, tanto en el eje X y en eje Y

Instanciar un mínimo de 5 movimientos para que muestre finalmente el historial de pasos de tu soldado

Módulos

1. Crear un módulo que validará nombres de usuarios. Dicho módulo, deberá tener una función y cumplir con los siguientes requisitos:

    - Crear un archivo principal (main.py) donde importará al módulo creado.
    - El nombre de usuario debe tener una longitud mínima de 7 caracteres y un máximo de 12.
    - El nombre de usuario debe ser alfanumérico (usar **isalnum()**)
    - Nombre de usuario con menos de 7 caracteres, retorna el mensaje "El nombre de usuario debe contener al menos 7 caracteres".
    - Nombre de usuario con más de 12 caracteres, retorna el mensaje "El nombre de usuario no puede contener más de 12 caracteres".

- Nombre de usuario con caracteres distintos a los alfanuméricos, retorna el mensaje "El nombre de usuario puede contener solo letras y números".
- Si el nombre de usuario es válido, la función retornará True.

2. Creando un archivo principal (main.py) donde llamará a un módulo (**operaciones.py**) el cuál contendrá las siguientes funciones:
   - La primera función cargará a 3 números enteros que pedirá al usuario que ingrese por consola un valor.
   - La segunda función solamente obtendrá la media de estos valores.
   - La última función te indicará cuál es el mayor de los 3 números
   - Desde el archivo principal importar al módulo y las funciones correspondiente usando las funciones anteriormente creadas en el módulo.
   - Usarlo al menos para 3 casos distintos.

3. Creando un archivo principal (main.py) donde importará a un módulo (operaciones.py) el cuál contendrá las siguientes funciones:
   - Una función que genere 30 números enteros aleatorios entre 0 y 100, y muestre en pantalla esta lista de números aleatorios
   - Otra función que ordene los valores de una lista y volver a mostrarla en pantalla
   - Otra función que me indicará cuál es el mayor de todos estos números de la lista

   Uso de la función random:

```python
import random

# Número decimal entre 0 y 1
print(random.random()) # Ejem: 0.9573551573973633

# Número entero entre dos valores (incluirá los extramos)
print(random.randint( a: 1,  b: 50)) # Ejem: 47

# Número entero entre dos valores (incluirá los extramos)
print(random.uniform( a: 1.3,  b: 9.3)) # Ejem: 5.2510440
```

4. Creando un archivo principal (main.py) donde importará a un módulo (operaciones.py) el cuál contendrá las siguientes funciones:
   - Una función que realizará la carga de un valor entero y que verifique que solamente tiene que ser numérico
   - Una función que mostrará por pantalla la raíz cuadrada de dicho valor
   - Y otra función que calculará el valor elevado al cuadrado y al cubo

de dicho número, puedes devolver un diccionario o una lista
- Utilizar el módulo math de python.

Decoradores

1. Crear una función decoradora que dará los buenos días antes de ejecutar una función llamada **saludo_inicial(nombre)** a ser decorada "Buenos días NOMBRE son las HORA horas con MINUTOS minutos" y luego de ser ejecutada lanzará un mensaje diciendo "Hasta luego NOMBRE que tenga buen día".

   La función a decorar pedirá el nombre de una persona y retornará el mensaje.

   Usar la función decorada al menos 3 veces
2. Haciendo el uso de ***args** y ****kwargs** aplicarlo debidamente para decorar una función que recibirá 6 argumentos los cuales se multiplicarán entre ellos (3 de ellos serán usado con ****kwargs**)
   Mensaje previo al usar el decorador "Está por ejecutarse la función" y mensaje luego de usar el decorador "La función ha sido ejecutado correctamente"
   Usar la función decorada al menos 3 veces

3. Crear un decorador **conteo_parametros(funcion)** donde imprimirá la cantidad de argumentos que tiene la función a decorar usando ***args** y ****kwargs**.
   Mensaje previo "La cantidad de argumentos que tiene la función es" mensaje post "La función decoradora terminó de ejecutarse correctamente"

   Ejemplo: Al usar una función suma que creamos:

   suma(4, 1, 10, 2, 50, 64)

   Salida:

   "La cantidad de argumentos que tiene la función es 5"

   "La función decoradora terminó de ejecutarse correctamente"

   Consideración:

   Todos los parámetros ingresados deben ser enteros, caso sean String alguno de ellos indicar al usuario: **"Solo está admitido datos enteros, no se podrá realizar la suma"**

   Usar la función al menos 3 veces

Consumo de JSON (opcional):

1. Queremos consumir un JSON que se encuentra alojado en la nube el cual nos traerá los datos de una persona como la edad, nombre, email, dirección o nombre de la compañía en donde trabaja.

La url a consumir usando Python es la siguiente:

https://jsonplaceholder.typicode.com/users

Obtener respectivamente los valores necesarios para formar la siguiente oración:

Bienvenido "nombre" "apellido", usted tiene "edad" años. El correo que nos proporcionó es "correo" y la compañía actual donde trabaja se llama "nombreCompañía". Dentro de sus datos también encontramos una website que es: "nombreWeb"

Finalmente agregar un usuario al json obtenido, pero sólo con los datos de nombre, apellido, edad y compañía donde trabaja y finalmente mostrarlo en consola (sólo ese usuario nuevo)

- Subir su siguiente carpeta de "reforzamientos 05" a su repositorio y la estructura final en GitHub debería verse del siguiente modo:

Indicaciones:

- Cada problema debe tener la descripción de lo solicitado como comentario en la parte superior de sus soluciones.
- Cada solución se desarrolla en un diferente archivo Python *.py y todos comprimidos en un solo archivo con el nombre de **nombre-apellido-reforzamiento-05.zip** o .rar
- **Correo** a enviar soluciones y link de repositorio: docente.cerseu.unmsm@gmail.com
- **Asunto:** Reforzamiento 05 – Módulo I
- Fecha máxima de entrega: sábado 4 de oct. hasta las 23:59 horas.