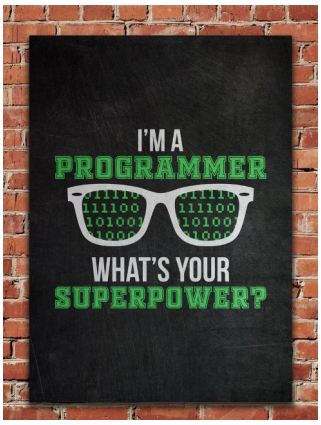
Trabajos Prácticos de Programación I

La presentación y aprobación de los trabajos son necesarios para la regularización de la materia.
Los trabajos se deben presentar comprimidos en un archivo (zip, rar o 7z) con el apellido y nombre del alumno. Cada trabajo práctico debe estar en un archivo .py, si algún trabajo requiere más de un archivo, éstos deben ser presentados también.

Trabajo práctico 1: Calculadora de impuesto

En un municipio la gente paga varios impuestos. El impuesto más importante, denominado *Impuesto Personal de Ingresos* (*IPI*, para abreviar) tiene que pagarse una vez al año y se evalúa utilizando la siguiente regla:



- Si el ingreso del ciudadano no es superior a 85.718 pesos, el impuesto es igual al 18% del ingreso menos 596 pesos y 2 centavos (esta fue la llamada *exención fiscal*).
- Si el ingreso es superior a esta cantidad, el impuesto es igual a 14,319 pesos y 2 centavos, más el 32% del excedente sobre 85.718 pesos.

Su tarea es escribir una **calculadora de impuestos**.

- Debe aceptar un valor de punto flotante: el ingreso del ciudadano.
- A continuación, debe imprimir el impuesto calculado, redondeado a pesos totales (sin centavos).

Nota: Este municipio nunca devuelve dinero a sus ciudadanos. Si el impuesto calculado es menor que cero, solo significa que no hay impuesto (el impuesto es igual a cero). Tenga esto en cuenta durante sus cálculos.
El código solo lee un valor de entrada y genera un resultado.
Pruebe su código con los datos siguientes

Entrada de muestra: `10000`
Resultado esperado: `El impuesto es: 1204.0 pesos`

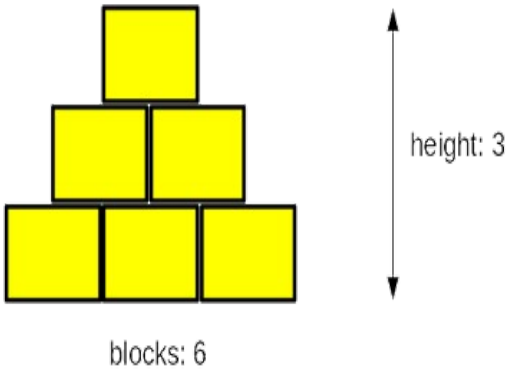
Entrada de muestra: `100000`
Resultado esperado: `El impuesto es: 18889.0 pesos`

Entrada de muestra: `1000`
Resultado esperado: `El impuesto es: 0.0 pesos`

Entrada de muestra: `-100`
Resultado esperado: `El impuesto es: 0.0 pesos`

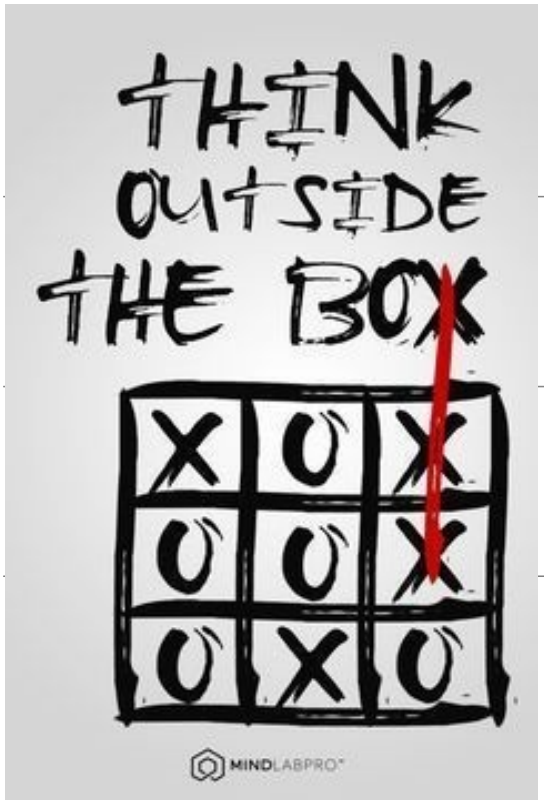
Trabajo práctico 2: Pirámide

Un niño y su padre, un programador de computadoras, juegan con bloques de madera. Están construyendo una pirámide.
Su pirámide es un poco rara, ya que en realidad es una pared en forma de pirámide, es plana. La pirámide se apila de acuerdo con un principio simple: cada capa inferior contiene un bloque más que la capa superior.
La figura ilustra la regla utilizada por los constructores:



Su tarea es escribir un programa que lea la cantidad de bloques que tienen los constructores, y generar la altura de la pirámide que se puede construir utilizando estos bloques.
El programa debe devolver la altura de la pirámide y cuantos bloques sobraron.

Nota: La altura se mide por el número de **capas completas**: si los constructores no tienen la cantidad suficiente de bloques y no pueden completar la siguiente capa, terminan su trabajo inmediatamente.
Pruebe su código con los siguientes datos:



Entrada de muestra: `6`
Salida esperada:
```
La altura de la pirámide es: 3
Sobraron: 0 piezas
```

Entrada de muestra: `20`
Salida esperada:
```
La altura de la pirámide es: 5
Sobraron: 5 piezas
```

Entrada de muestra: `1000`
Salida esperada:
```
La altura de la pirámide es: 44
Sobraron: 10 piezas
```

Entrada de muestra: `2`
Salida esperada:
```
La altura de la pirámide es: 1
Sobraron: 1 piezas
```

Trabajo práctico 3: Primos

*Un número natural es **primo** si es mayor que 1 y no tiene divisores más que 1 y si mismo.*
¿Complicado? De ninguna manera. Por ejemplo, 8 no es un número primo, ya que puede dividirlo entre 2 y 4 (no podemos usar divisores iguales a 1 y 8, ya que la definición lo prohíbe).
Por otra parte, 7 es un número primo, ya que no podemos encontrar ningún divisor para él que no sean 1 y 7. Su tarea es escribir una función que verifique si un número es primo o no.
La función:

- Se llama `esPrimo`.
- Toma un argumento (el valor a verificar).
- Devuelve `True` si el argumento es un número primo, y `False` de lo contrario.

Para probar la función, realizar un programa que muestre los número primos menores o iguales a uno ingresado por el usuario.

Trabajo práctico 4: Contraseña

Crear un generador de contraseñas aleatorias en Python que deben cumplir los siguientes requisitos:

- Entre 10 y 15 caracteres
- Al menos una letra mayúscula
- Al menos una letra minúscula
- Al menos cuatro números, no consecutivos ni repetidos
- Al menos un carácter no alfanumérico, por ejemplo: #!@] *($

Trabajo práctico 5: Ta-Te-Ti

Su tarea es escribir **un simple programa que simule jugar a ta-te-ti con el usuario**. Para hacerlo más fácil, Hemos decidido simplificar el juego. Aquí están nuestras reglas:

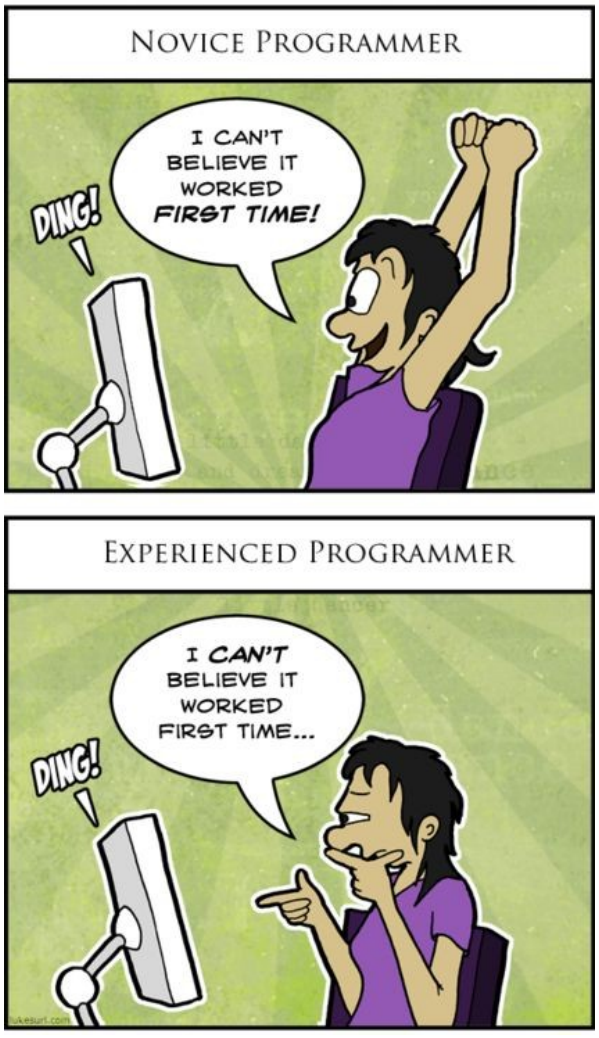
- La maquina jugará utilizando las `'X'`s.
- El usuario jugará utilizando las `'O'`s.
- El primer movimiento es de la maquina: siempre coloca una `'X'` en una posición al azar.
- Todos los cuadros están numerados comenzando con el `1` (observe el ejemplo para que tenga una referencia).
- El usuario ingresa su movimiento introduciendo el numero de cuadro elegido. El numero debe de ser valido, un valor entero entre `1` y `9`, y no puede ser un cuadro que ya esté ocupado por una `X` o un `O`.
- El programa verifica si el juego ha terminado. Existen cuatro posibles veredictos: el juego continua, el juego termina en empate, gana el usuario, o la maquina gana.
- La maquina responde con su movimiento y se verifica el estado del juego.
- Opcionalmente puede implementar algún tipo de inteligencia artificial, o la maquina elegirá un cuadro de manera aleatoria.

El ejemplo del programa es el siguiente:

1	2	3
4	5	6
7	8	9

```
La compu juega el movimiento: 5
```

1	2	3
4	X	6
7	8	9




```
Ingresa tu movimiento: 1
```

0	2	3
4	X	6
7	8	9

```
La compu juega el movimiento: 2
```

0	X	3
4	X	6
7	8	9

```
Ingresa tu movimiento: 8
```

0	X	3
4	X	6
7	0	9

```
La compu juega el movimiento: 6
```

0	X	3
4	X	X
7	0	9

```
Ingresa tu movimiento: 4
```

O	X	3
O	X	X
7	O	9

```
La compu juega el movimiento: 3
```

O	X	X
O	X	X
7	O	9

```
Ingresa tu movimiento: 7
```

O	X	X
O	X	X
O	O	9

```
¡Has Ganado!
```

Trabajo práctico 6: Juego de azar

Lotería Chaqueña le encarga realizar un simulador para un nuevo juego que lanzará próximamente. Es una variación de la Quiniela Poceada existente.

Modalidad
Se trata de un juego poceado. Para definir el extracto del juego, se tomarán al azar 20 (veinte) números de dos cifras (del 00 al 99).
Cada apostador deberá confeccionar su boleta eligiendo 8 (ocho) números de dos dígitos, distintos entre si, del 00 al 99.
Si los números elegidos están dentro de los 20 números del extracto del sorteo, sin importar el orden, obtendrá un premio.
Ganará un premio de la Poceada quien acierte 8, 7, 6 ó 5 números. Por lo cual se necesita saber cuantos ganadores acertaron con cada cantidad de aciertos.

Se necesita realizar un programa que simule los sorteos.
Primeramente se deben generar al azar una cantidad n de boletas.
Luego realizar el sorteo de los 20 números (seleccionados al azar).
Y por último realizar los cálculos de resultados (cuantas boletas hubo con 8, 7, 6 ó 5 aciertos)

Trabajo práctico 7: Ahorcado

Juego de adivinanza El Ahorcado, en el que el jugador tiene que adivinar las letras para encontrar una palabra oculta. La versión más básica del juego no tiene que incluir gráficos, puedes hacerlo un proyecto basado en texto.

En primer lugar, necesitará una lista de palabras para el sistema para elegir una al azar (se puede cargar de un archivo o generarla de un texto). Después, tendrá que elegir las funciones apropiadas para revisar si la entrada del usuario es una letra, si la palabra oculta la contiene, y si es así, cuántas veces. Su código también tendrá que usar la función print para mostrar las letras encontradas o no, y limitar el número de intentos.

Trabajo práctico 8: Analizador de texto

Crear un programa que le pida al usuario que ingrese un texto, lo que sea.
Luego el programa le va a pedir al usuario que ingrese tres letras a su elección.
Se procesará el texto para devolver:
1. Cuantas veces aparece cada letra que eligió el usuario, sin importar si aparecen en mayúsculas o minúsculas.
2. Cuantas palabras hay en el texto, y cuantas veces aparece cada una
3. Primera y última letra del texto.
4. Mostrar el texto invirtiendo el orden de las palabras.
5. Decir si determinada palabra ingresada por el usuario se encuentra en el texto.
6. Cuantas palabras hay de cada cantidad de letras (cuantas con una letra, cuantas con dos, cuantas con tres, etc.).

Trabajo práctico 9: Libreta de direcciones

Una aplicación de libreta de direcciones es una aplicación que almacena información de contacto como nombre, direcciones y números de teléfono. También debes crear funciones que permitan a los usuarios de la aplicación:
- Ingresar, actualizar y eliminar información de contacto,
- ubicar las direcciones con coordenadas en un mapa (investigar folium),
- mandar whatsapp a un determinado contacto (investigar cómo se puede hacer),
- cualquier otra función que considere o necesite.