# Ejercicios de CodoACodo

## Unidad 1

1. Escribe un programa que muestre por pantalla “Hello World”
2. Escriba un programa que escriba en pantalla el resultado de sumar 3+5
3. Escribe un programa que pida el nombre del usuario y escriba un texto que diga “Hola nombreUsuario”
4. Escribe un programa que pida un número, pida otro número y escriba el resultado de sumar estos dos números
5. Escriba un programa que pida dos números y escriba en pantalla cuál es el mayor
6. Escribe un programa que pida 3 números y escriba en la pantalla el mayor de los tres
7. Escribe un programa que pida un número y diga si es divisible por 2
8. Escribe un programa que pida un número y nos diga si es divisible por 2, 3, 5 o 7 (solo hay que comprobar si lo es por uno de los cuatro)
9. Añadir al ejercicio anterior que nos diga por cuál de los cuatro es divisible
10. Escribir un programa que nos diga en pantalla los divisores de un número dado
11. Escribir un programa que nos diga si un número dado es primo
12. Pide una nota. Muestra la calificación según la nota: 0-3 (muy deficiente), 3-5 (insuficiente), 5-6 (suficiente), 6-7 (bien), 7-9 (notable), 10 (sobresaliente)
13. Realiza un programa que escriba una pirámide del 1 al 30 de la siguiente forma

```
1
22
333
4444
55555
666666
  ..........
```

14. Haz un programa que escriba una pirámide inversa de los números del 1 al número que indique el usuario de la siguiente forma (suponiendo que indica 6):

```
666666
55555
4444
333
22
1
```

15. Crear un programa que escriba los números del 1 al 500, y que indique cuales son múltiplos de 4 y de 9 y que cada 5 líneas muestre una línea horizontal. Por ejemplo:

```
1
2
3
4 (Múltiplo de 4)
5
------------------------------------------------------------
6
7
8 (Múltiplo de 4)
9 (Múltiplo de 9)
10
```

## Unidad 2

1. Desarrollar una función que reciba tres números positivos y devuelva el mayor de los tres, sólo si éste es único (mayor estricto). En caso de no existir el mayor estricto devolver -1. No utilizar operadores lógicos (and, or, not). Desarrollar también un programa para ingresar los tres valores, invocar a la función y mostrar el máximo hallado, o un mensaje informativo si éste no existe.
2. Desarrollar una función que reciba tres números enteros positivos y verifique si corresponden a una fecha válida (día, mes, año). Devolver True o False según la fecha sea correcta o no. Realizar también un programa para verificar el comportamiento de la función.
3. Un comercio de electrodomésticos necesita para su línea de cajas un programa que le indique al cajero el cambio que debe entregarle al cliente. Para eso se ingresan dos números enteros, correspondientes al total de la compra y al dinero recibido. Informar cuántos billetes de cada denominación deben ser entregados al cliente como vuelto, de tal forma que se minimice la cantidad de billetes. Considerar que existen billetes de $500, $100, $50, $20, $10, $5 y $1. Emitir un mensaje de error si el dinero recibido fuera insuficiente. Ejemplo: si la compra es de $317 y se abona con $500, el vuelto debe contener 1 billete de $100, 1 billete de $50, 1 billete de $20, 1 billete de $10 y 3 billetes de $1.
4. Escribir dos funciones separadas para imprimir por pantalla los siguientes patrones de asteriscos, donde la cantidad de filas se recibe como parámetro:
5. Crear una función lambda que devuelva el cuadrado de un valor recibido cómo parámetro. Desarrollar además un programa para verificar el comportamiento de la función.
6. Crear una función lambda para comprobar si un número es par o impar. Desarrollar además un programa para verificar el comportamiento de la función.
7. Crear una lista con los cuadrados de los números entre 1 y N (ambos incluidos), donde N se ingresa desde el teclado. Luego se solicita imprimir los últimos 10 valores de la lista.
8. Eliminar de una lista de palabras que se encuentren en una segunda lista. Imprimir la lista original, la lista de palabras a eliminar y la lista resultante
9. Escribir una función que reciba una lista como parámetro y devuelva True si la lista está ordenada en forma ascendente o False en caso contrario. Por ejemplo, ordenada([1, 2, 3]) retorna True y ordenada(['b', 'a']) retorna False. Desarrollar además un programa para verificar el comportamiento de la función.
10. Desarrollar una función que determine si una cadena de caracteres es capicúa, sin utilizar cadenas auxiliares ni rebanadas. Escribir además un programa que permita verificar su funcionamiento.

11. Leer una cadena de caracteres e imprimirla centrada en pantalla. Suponer que la misma tiene 80 columnas
12. Escribir una función que reciba como parámetro una tupla conteniendo una fecha (día,mes,año) y devuelva una cadena de caracteres con la misma fecha expresada en formato extendido. Por ejemplo, para (12, 10,17) devuelve “12 de Octubre de 2017”. Escribir también un programa para verificar su comportamiento.
13. Ingresar una frase desde el teclado y usar un conjunto para eliminar las palabras repetidas, dejando un solo ejemplar de cada una. Finalmente mostrar las palabras ordenadas según su longitud.
14. Desarrollar una función eliminar_claves() que reciba como parámetros un diccionario y una lista de claves. La función debe eliminar del diccionario todas las claves contenidas en la lista, devolviendo el diccionario modificado y un valor de verdad que indique si la operación fue exitosa. Desarrollar también un programa para verificar su comportamiento.
15. Escribir una función para eliminar una subcadena de una cadena de caracteres, a partir de una posición y cantidad de caracteres dados, devolviendo la cadena resultante. Escribir también un programa para verificar el comportamiento de la misma. Escribir una función utilizando rebanadas.

---

## Unidad 3

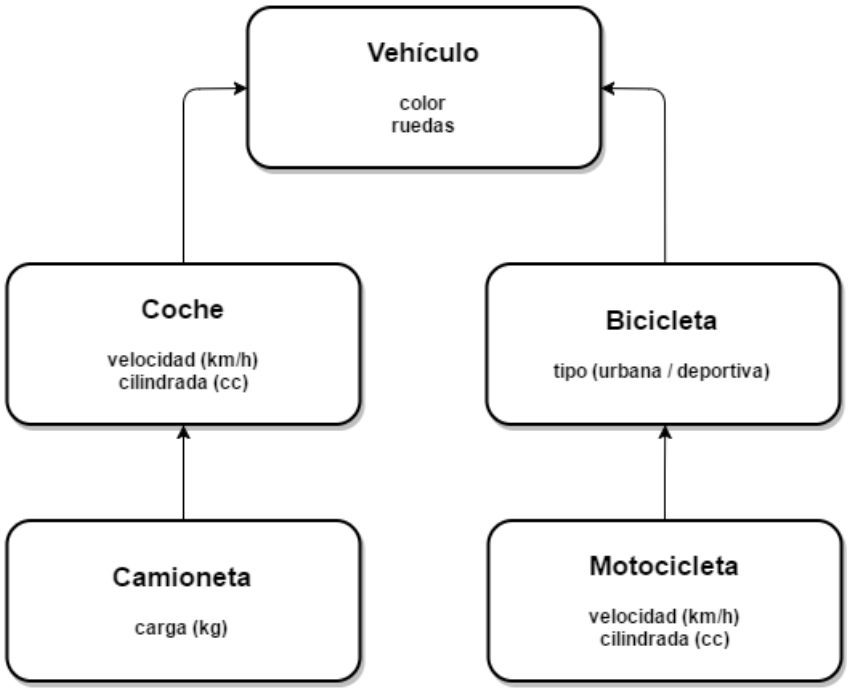
1. Crear una clase Persona con los métodos “set_nombre”, “set_edad”, “get_nombre”, “get_edad” y “print_persona”. Luego crear dos objetos del tipo Persona e imprimirlos por consola.
2. Agregarle a la clase anterior un constructor que reciba nombre y edad.
3. Agregarle a la clase anterior un método “es_mayor_de_edad” que devuelva True o False.
4. Agregarle un método “es_mayor_que” el cual recibe un objeto persona y compara su edad con la del objeto actual.
5. Agregarle un método estático “get_mayor” que reciba dos objetos Persona y devuelva el de edad mayor.
6. Realizar un programa que conste de una clase llamada Alumno que tenga como atributos el nombre y la nota del alumno. Definir los métodos para inicializar sus atributos, imprimirlos y mostrar un mensaje con el resultado de la nota y si ha aprobado o no.´
7. Desarrollar un programa que cargue los datos de un triángulo. Implementar una clase con los métodos para inicializar los atributos, imprimir el valor del lado con un tamaño mayor y el tipo de triángulo que es (equilátero, isósceles o escaleno)
8. Realizar un programa en el cual se declaren dos valores enteros por teclado utilizando el método **init**. Calcular después la suma, resta, multiplicación y división. Utilizar un método para cada una e imprimir los resultados obtenidos. Llamar a la clase Calculadora.
9. Realizar una clase que administre una agenda. Se debe almacenar para cada contacto el nombre, el teléfono y el email. Además deberá mostrar un menú con las siguientes opciones: Añadir contacto, Listar contactos, Buscar contacto, Editar contacto, Cerrar agenda.

10. En un banco tienen clientes que pueden hacer depósitos y extracciones de dinero. El banco requiere también al final del día calcular la cantidad de dinero que se ha depositado. Se deberán crear dos clases, la clase cliente y la clase banco. La clase cliente tendrá los atributos nombre y cantidad y los métodos **init**, depositar, extraer, mostrar_total. La clase banco tendrá como atributos 3 objetos de la clase cliente y los métodos **init**, operar y deposito_total.

---

## Unidad 4

1. Crea la clase Vehiculo, extiende la clase y realiza la siguiente implementación:




   Crea al menos un objeto de cada subclase y añádelos a una lista llamada vehiculos.
   Nota: Puedes utilizar el atributo especial de clase name para recuperar el nombre de la clase de un objeto: type(objeto).**name**

2. Continua con el ejercicio anterior y realiza una función llamada catalogar() que reciba la lista de vehiculos y los recorra mostrando el nombre de su clase y sus atributos.
3. Continua con el ejercicio anterior y modifica la función catalogar() para que reciba un argumento optativo ruedas, haciendo que muestre únicamente los que su número de ruedas concuerde con el valor del argumento. También debe mostrar un mensaje “Se han encontrado {} vehículos con {} ruedas:” únicamente si se envía el argumento ruedas. Ponla a prueba con 0, 2 y 4 ruedas como valor.
4. Localiza el error en el siguiente bloque de código. Crea una excepción para evitar que el programa se bloquee y además explica en un mensaje al usuario la causa y/o solución: resultado = 10/0
5. Realiza una función llamada agregar_una_vez(lista, el) que reciba una lista y un elemento. La función debe añadir el elemento al final de la lista con la condición de no repetir ningún elemento. Además si este elemento ya se encuentra en la lista se debe invocar un error de tipo ValueError que debes capturar y mostrar este mensaje en su lugar: Error: Imposible añadir elementos duplicados => [elemento].

Cuando tengas la función intenta añadir los siguiente valores a la lista 10, -2, "Hola" y luego muestra su contenido.
Sugerencia: podés utilizar la sintaxis “elemento in lista”.
elementos = [1, 5, -2]