EJERCICIOS

MUSK

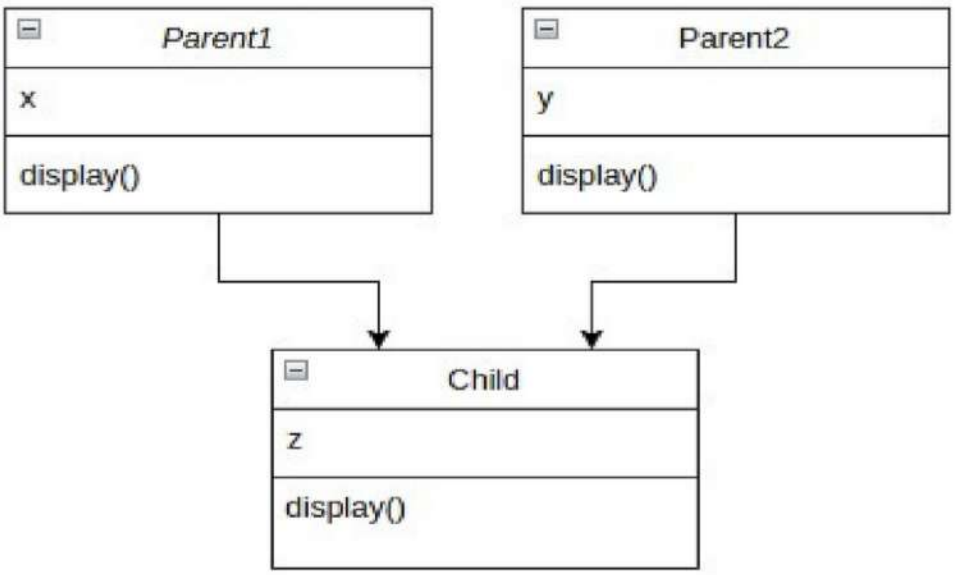
EJERCICIO 1

Crea una clase Staff con los atributos role, depty salary. Crea una clase Profesor que herede de la clase anterior y que además tenga como atributos nombre y edad. Haz que sea posible instanciar un profesor dándole valor a todos los atributos.

EJERCICIO 2

Representa el siguiente diagrama con sus clases, atributos y métodos correspondientes.

Cada método display debe imprimir el nombre de la clase, atributos y valores de la instancia en ese momento. Ejemplo: In displaymethodofParent1x=10



EJERCICIO 3

Crea una clase Car que herede de Vehicle y que sobreescriba los métodos max_speed() y change_gear(). Instancia dos objetos de cada clase y compruebaque la salida de cada método es distinta

```
class Vehicle:
    def __init__(self, name, color, price):
        self.name = name
        self.color = color
        self.price = price

    def show(self):
        print('Details:', self.name, self.color, self.price)

    def max_speed(self):
        print('Vehicle max speed is 150')

    def change_gear(self):
        print('Vehicle change 6 gear')
```

EJERCICIO 1

Dadas las siguientes clases con el output de sus respectivos métodos, crea una interfaz formal que las implemente.

```
svm = SVM()
svm.preprocess_data(data=None, y=None)
svm.fit()
svm.predict()

dt = DecisionTree()
dt.preprocess_data(data=None, y=None)
dt.fit()
dt.predict()

output:

Preprocessing data at SVM
Training at SVM
Evaluating at SVM
Preprocessing data at DecisionTree
Training at DecisionTree
Evaluating at DecisionTree
```

EJERCICIO 2

Repite el ejercicio anterior esta vez creando una interfaz informal.

EJERCICIO 3

Crea una clase virtual llamada Algoritmo con los atributos nombre, tarea y aprendizaje que sea superclase de la clase BaseClassifier del problema anterior. Comprueba con el método issubclass que Algoritmo es padre de BaseClassifier.

EJERCICIO 1

Escribe un script en Python para mostrar los distintos formatos de fecha y hora.

a) Fecha y hora actuales
b) Año actual
c) Mes del año
d) Número de la semana del año
e) Día de la semana
f) Día del año
g) Día del mes
h) Día de la semana

EJERCICIO 2

Escribe un programa en Python para convertir una cadena a datetime.

```
INPUT : Jan 1 2014 2:43PM
OUTPUT : 2014-07-01 14:43:00
```

EJERCICIO 3

Escribe un programa en Python para obtener la hora actual.

EJERCICIO 4

Escribe un programa en Python para restar cinco días a la fecha actual.

EJERCICIO 5

Escribe un programa en Python para convertir una cadena de marcas de tiempo unix en una fecha legible.

```
INPUT Unix timestamp string : 1284105682
OUTPUT : 2010-09-10 13:31:22
```

EJERCICIO 6

Escribe un programa en Python para sumar 5 segundos con la hora actual

EJERCICIO 7

Escribe un programa en Python para obtener el número de la semana.

EJERCICIO 8

Escribe un programa en Python para seleccionar todos los domingos de un año determinado.

EJERCICIO 9

Escribe un programa en Python para contar el número de lunes del primer día del mes desde 2015 hasta 2016.

EJERCICIO 10

Escribe un programa en Python para crear 12 fechas fijas a partir de una fecha especificada en un periodo determinado. La diferencia entre dos fechas será de 20.

EJERCICIO 1

Implementa una función generadora que dadas dos listas del mismo tamaño, devuelva la multiplicación entre los elementos de cada una, el primer elemento de la lista 1 por el primero de la lista 2, el segundo con el segundo y así sucesivamente. Sigue la siguiente estructura:

```
def prod(l1, l2):
    ...
    except StopIteration:
        pass
    return solution
```

Añadiendo el bloque except capturamos la excepción de Stop Iteration que se produce al acabar de leer todos los elementos de un generador.

EJERCICIO 2

Implementa un generador, que dado un entero n, genere n números aleatorios. Puedes usar el método random de la librería random para generar números aleatorios.

EJERCICIO 3

Implementa un generador de Fibonacci que genere n números de la secuencia de Fibonacci, que tiene la forma:

0, 1, 1, 2, 3, 5, 8, 13, ...
Cuyos dos primeros valores son 0 y 1 por definición y el resto se calculan sumando los dos últimos valores de la sucesión.

EJERCICIO 4

Implementa un generador, que dado un entero n, imprima todos los números inferiores a n multiplicados por dos.

EJERCICIO 5

Implementa un generador, que dado un entero n, genere n número senares.

EJERCICIO 1

Crea una función que genere una excepción e imprima su tipo, los argumentos de la excepción y su mensaje de error.

EJERCICIO 2

Crea una función que compute la diferencia entre dos enteros. En caso de que la diferencia sea negativa genera una excepción inventada por ti que informe sobre ello. Por ejemplo. la excepción podría llamarse NegativeDifferenceException.

EJERCICIO 3

Crea una función que calcule la división entre dos números. Debe imprimir el mensaje 'Los parámetros deben ser número enteros' cuando se genera una excepción de tipo y 'El divisor no puede ser 0' cuando se genera un zerodivisionerror.

EJERCICIO 4

Añade a la función anterior, un mensaje que se imprima al final de la ejecución de la función independientemente de si se ha generado excepción o no.


MUSK