Exercicios PYTHON

Conceptos Básicos de POO en Python

```
class Calculator():
         def __init__ (self, operador1, operador2) #es el constructor
                  self.op1 = operador1
                  self.op2 = operador2
         def suma(self):
                  return self.op1 + self.op2
         def resta(self):
                  return self.op1 - self.op2
         def producto(self):
                  return self.op1 * self.op2
         def dividir(self):
                  return self.op1 / self.op2
                                                       #creamos un objeto
>>> calcula = Calculadora(3,2)
>>> calcula.suma()
5
>>> calcula.producto()
6
....
```

Algunhas excepcións:

- •NameError: Esta excepción levántase cando o programa non pode encontrar un nome local o global.

 O nome está incluido no mensaxe de erro.
- •TypeError: Esta excepción levántase cando unha función se lle pasa un obxecto de tipo inapropiado como seu argumento. Os detalles sobre o tipo incorrecto son proporcionados na mensaxe de erro.
- •ValueError: Esta excepción ocurre cando un argumento de función ten o tipo correcto pero un valor inapropiado.
- •NotImplementedError: Esta excepción levántase cando se supón que un obxecto ten una operación que non foi implementada aún. Aínda así, nestes casos é preferible usar TypeError
- •ZeroDivisionError: Esta excepción levántase cando o segundo argumento para unha operación de división é cero.
- FileNotFoundError: Esta excepción levántase cando o arquivo o diccionario solicitado non existe.

Ejemplo:

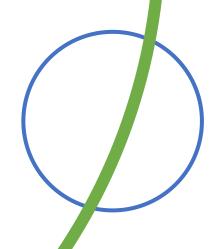
```
import math
Lista = [10, -5, 1.2, 'apple']
for N in lista:
          try:
                    factorial = math.factorial(N)
          except TypeError:
                    print("Tipos de entrada no compatible con la operación.")
          except ValueError:
                    print("Solo acepta números positivo. ", N," no es un número positivos.")
         else:
                    print("El factorial of ", N ,"es", factorial)
          finally:
                     print("Fin de la iteración")
```

```
class Circulo:
         pi = 3.1415
         def __init__(self, radio):
                  self.radio = radio
         def area(self):
                  return Circulo.pi * self.radio ** 2
if __name__ == '__main__':
         radio = float(input('Dame el valor del radio: '))
         c1 = Circulo(radio)
         print('El área es: ', c1.area())
   _name__ == '__main__' el entorno de ejecución
self
                                       parámetro sin valor para referenciar al objeto mismo
```

```
El método llamado ___Str__ que se invoca cada vez que se llama a las funciones print o str∕
   class Tarjeta:
         def __init__(self, numero, cantidad = 0):
               self.numero = numero
               self.saldo = cantidad
               return
         def __str__(self):
               return 'Tarjeta número {} con saldo {:.2f}€'.format(self.numero, str(self.saldo))
   >>> t = tarjeta('0123456789', 1000)
   >>> print(t)
   Tarjeta número 0123456789 con saldo 1000.00€
```

```
class Tarjeta:
         def __init__(self, id, cantidad = 0):
                                                           herencia
                    self.id = id
                    self.saldo = cantidad
                   return
          def mostrar_saldo(self): # Método de la clase Tarjeta que hereda la clase Tarjeta_descuento
                    print('El saldo es', self.saldo, '€.')
                   return
class Tarjeta_descuento(Tarjeta):
          def __init__(self, id, descuento, cantidad = 0):
                    self.id = id
                    self.descuento = descuento
                    self.saldo = cantidad
                   return
          def mostrar_descuento(self): # Método exclusivo de la clase Tarjeta_descuento
                    print('Descuento de', self.descuento, '% en los pagos.')
                   return
>>> t = Tarjeta_descuento('0123456789', 2, 1000)
>>> t.mostrar_saldo()
El saldo es 1000 €.
>>> t.mostrar_descuento()
Descuento de 2 % en los pagos.
```





Crea una clase llamada Cuenta que tendrá los siguientes atributos: titular (que es una persona) y cantidad (puede tener decimales).

El'titular será obligatorio y la cantidad es opcional. Construye los siguientes métodos para la clase:

- Un constructor, donde los datos pueden estar vacíos.
- Los **setters y getters** para cada uno de los atributos. El atributo no se puede modificar directamente, sólo ingresando o retirando dinero.
- mostrar(): Muestra los datos de la cuenta.
- ingresar(cantidad): se ingresa una cantidad a la cuenta, si la cantidad introducida es negativa, no se hará nada.
- retirar(cantidad): se retira una cantidad a la cuenta. La cuenta puede estar en números rojos.
 - Nota: ejemplos de métodos setters y getters:

```
# getter method
  def get_age(self):
    return self._age

# setter method
  def set_age(self, x):
    self._age = x
```