Ejercicios de Programación Funcional

Ejercicio 1

Escribir una función que aplique un descuento a un precio y otra que aplique el IVA a un precio. Escribir una tercera función que reciba un diccionario con los precios y porcentajes de una cesta de la compra, y una de las funciones anteriores, y utilice la función pasada para aplicar los descuentos o el IVA a los productos de la canasta y devolver el precio final de la canasta.

```
import numpy as np
from functools import reduce
import operator as op
def descto(precio,porcentaje):
    return precio*(1-porcentaje/100)

def IVA(precio,iva=21):
    return precio*(1+iva/100)

def calcular(dictionary, typeOperation=['iva','descuento']):
    suma_canasta = []
    if(typeOperation=='iva'):
        for i,value in dictionary.items():
            suma_canasta.append(IVA(value['precio'],21))
    else:
        for i,value in dictionary.items():
            suma_canasta.append(descto(value['precio'],value['porcentaje']))
    return suma_canasta

canasta = dict({'id01':{'precio':1000,'porcentaje':np.random.rand()*100},
                'id02':{'precio':2000,'porcentaje':np.random.rand()*100},
                'id03':{'precio':3000,'porcentaje':np.random.rand()*100},
                'id04':{'precio':4000,'porcentaje':np.random.rand()*100},
                'id05':{'precio':54000,'porcentaje':np.random.rand()*100},
                'id06':{'precio':12300,'porcentaje':np.random.rand()*100},
                'id07':{'precio':123400,'porcentaje':np.random.rand()*100}})

print(f"La suma total de la canasta con iva es: {reduce(op.add,calcular(canasta,'iva'))}")
print(f"La suma total de la canasta con descuento es: {reduce(op.add,calcular(canasta,'descue
```

```
La suma total de la canasta con iva es: 241637.0
La suma total de la canasta con descuento es: 54818.493719165046
```

Ejercicio 2

Escribir una función que simule una calculadora científica que permita calcular el seno, coseno, tangente, exponencial y logaritmo neperiano. La función preguntará al usuario el valor y la función a aplicar, y mostrará por pantalla una tabla con los enteros de 1 al valor introducido y el resultado de aplicar la función a esos enteros.

```
operaciones = ['sin','cos','tan','exp','ln']
```

```
import ipywidgets as widgets

type_op = widgets.Dropdown(options = operaciones,
                           value='sin',
                           description='¿Que operacion?:\n',
                           disabled=False)
```

```
value= float(input("Ingresa un numero:\n\n"))
def calcular(option):
    if(option=='sin'):
        return np.sin(value)
    if(option=='cos'):
        return np.cos(value)
    if(option=='tan'):
        return np.tan(value)
    if(option=='exp'):
        return np.e**(value)
    if(option=='ln'):
        if(value<=0):
            print("No se puede ingresar un numero negativo o cero, adios.")
            return -1
        else:
            return np.log(value)
```

```
widgets.interact(calcular,option=type_op);
```

Ejercicio 3

Escribir una función que reciba otra función y una lista, y devuelva otra lista con el resultado de aplicar la función dada a cada uno de los elementos de la lista.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
n = 10

def receive(func, lis):
  new_list = []
  for l in lis:
    new_list.append(func(l, n))
  return new_list

def calculare(v , m = 1):
  return m*v

receive(calculare, numbers)
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

Ejercicio 4

Escribir una función que reciba otra función booleana y una lista, y devuelva otra lista con los elementos de la lista que devuelvan True al aplicarles la función booleana.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

def receive(func, lis):
  new_list = []
  for l in lis:
    if func(l):
      new_list.append(l)
    else:
      continue
  return new_list

def pair_number(n):
  if n % 2 == 0:
    return True
  else:
    return False
```

```
receive(pair_number, numbers)
```

Ejercicio 5

Escribir una función que reciba una frase y devuelva un diccionario con las palabras que contiene y su longitud.

```
frase = input("Ingrese su frase: ")

def contador(frase):
  lista_palabras = frase.split(" ")
  return { palabra:len(palabra) for palabra in lista_palabras }

contador(frase)
```

```
Ingrese su frase: hola
{'hola': 4}
```

Ejercicio 6

Escribir una función reciba una lista de notas y devuelva la lista de calificaciones correspondientes a esas notas.

```
def calificacion(nota):
  if nota>7 or nota < 1:
    return "Nota fuera de rango"
  elif nota>=4:
    return "Aprobado"
  else:
    return "Reprobado"

calificacion(4)
calificacion(3)
```

Ejercicio 7

Escribir una función reciba un diccionario con las asignaturas y las notas de un alumno y devuelva otro diccionario con las asignaturas en mayúsculas y las calificaciones correspondientes a las notas.

```
def calificar_dic(dic):
  dic2 = {}
  for asig in dic:
    nota = dic[asig]
    resultado = ""
    if nota > 7 or nota < 1:
      resultado = "Nota fuera de rango"
    elif nota>=4:
      resultado = "Aprobado"
    else:
      resultado = "Reprobado"
    dic2.update({asig.upper(): resultado})
  #print("Calificaciones: ",dic2)
  return dic2


notas = {"Lenguaje" : 3.4, "Historia" : 4, "Matemática" : 4.5 , "Ciencias Sociales" : 3.9}
print ("Notas: ",notas)
print("Calificaciones: ",calificar_dic(notas))
```

```
Notas:  {'Lenguaje': 3.4, 'Historia': 4, 'Matemática': 4.5, 'Ciencias Sociales': 3.9}
Calificaciones:  {'LENGUAJE': 'Reprobado', 'HISTORIA': 'Aprobado', 'MATEMÁTICA': 'Aproba
```

Ejercicio 8

Escribir una función reciba un diccionario con las asignaturas y las notas de un alumno y devuelva otro diccionario con las asignaturas en mayúsculas y las calificaciones correspondientes a las notas aprobadas.

```
def calificar_dic(dic):
  dic2 = {}
  for asig in dic:
    nota = dic[asig]
    resultado = ""
    if nota > 7 or nota < 4:
      resultado = nota
    elif nota>=4:
      resultado = "Aprobado"
    else:
```

```
        resultado = nota
      if resultado == "Aprobado":
        dic2.update({asig.upper(): resultado})
    #print("Calificaciones: ",dic2)
    return dic2


notas = {"Lenguaje" : 3.4, "Historia" : 4, "Matemática" : 4.5 , "Ciencias Sociales" : 3.9}
print ("Notas: ",notas)
print("Calificaciones: ",calificar_dic(notas))
```

```
Notas:  {'Lenguaje': 3.4, 'Historia': 4, 'Matemática': 4.5, 'Ciencias Sociales': 3.9}
Calificaciones:  {'HISTORIA': 'Aprobado', 'MATEMÁTICA': 'Aprobado'}
```

Ejercicio 9

Escribir una función que calcule el módulo de un vector.

```
import math

def calc_mod(x,y,z=0):
  suma = x**2 + y**2 + z**2
  mod = math.sqrt(suma)
  return mod

print(calc_mod(2,2,2))
```

Ejercicio 10

Una inmobiliaria de una ciudad maneja una lista de inmuebles como la siguiente:

```
[{'año': 2000, 'metros': 100, 'habitaciones': 3, 'garaje': True, 'zona': 'A'},
{'año': 2012, 'metros': 60, 'habitaciones': 2, 'garaje': True, 'zona': 'B'},
{'año': 1980, 'metros': 120, 'habitaciones': 4, 'garaje': False, 'zona': 'A'},
{'año': 2005, 'metros': 75, 'habitaciones': 3, 'garaje': True, 'zona': 'B'},
{'año': 2015, 'metros': 90, 'habitaciones': 2, 'garaje': False, 'zona': 'A'}]
```

Construir una función que permita hacer búsqueda de inmuebles en función de un presupuesto dado. La función recibirá como entrada la lista de inmuebles y un precio, y devolverá otra lista con los inmuebles cuyo precio sea menor o igual que el dado. Los inmuebles de la lista que se devuelva

deben incorporar un nuevo par a cada diccionario con el precio del inmueble, donde el precio de un inmueble se calcula con las siguiente fórmula en función de la zona:

- Zona A: precio = (metros * 1000 + habitaciones * 5000 + garaje * 15000) * (1-antiguedad/100)
- Zona B: precio = (metros * 1000 + habitaciones * 5000 + garaje * 15000) * (1-antiguedad/100) * 1.5

```
lista = [{'año': 2000, 'metros': 100, 'habitaciones': 3, 'garaje': True, 'zona': 'A'},
{'año': 2012, 'metros': 60, 'habitaciones': 2, 'garaje': True, 'zona': 'B'},
{'año': 1980, 'metros': 120, 'habitaciones': 4, 'garaje': False, 'zona': 'A'},
{'año': 2005, 'metros': 75, 'habitaciones': 3, 'garaje': True, 'zona': 'B'},
{'año': 2015, 'metros': 90, 'habitaciones': 2, 'garaje': False, 'zona': 'A'}]

# Función recibirá como entrada la lista de inmuebles y un precio
def presupuestador(presupuesto, inmuebles):
  en_presup = []
  for x in inmuebles:
    precio = 0
    precio += x['metros']*1000
    precio += x['habitaciones']*5000
    if x['garaje']==True:
      precio += 15000
    antiguedad = 2022 - x['año']
    precio *= (1-antiguedad/100)

    if x['zona'] == "B":
      precio *= 1.5
    x.update({'precio':precio})
    if precio<=presupuesto:
      en_presup.append(x)
  return en_presup

presupuesto=float(input("Ingrese su presupuesto: "))
presupuestador(presupuesto,lista)
```

```
Ingrese su presupuesto: 100000
[{'año': 1980,
  'garaje': False,
  'habitaciones': 4,
  'metros': 120,
  'precio': 81200.00000000001,
  'zona': 'A'},
 {'año': 2015,
  'garaje': False,
  'habitaciones': 2,
  'metros': 90,
  'precio': 93000.0,
  'zona': 'A'}]
```

Ejercicio 11

Escribir una función que reciba una muestra de números y devuelva los valores atípicos, es decir, los valores cuya puntuación típica sea mayor que 3 o menor que -3.

Nota: La puntuación típica de un valor se obtiene restando la media y dividiendo por la desviación típica de la muestra.

```
from statistics import mean, stdev

muestra = [1, 67, 6, 4, 4335, 6, 7, 89, 9, 10, 121, 13, 15, 1020]

def calcular_atipicos(m):
    media = mean(m)
    desviacion = stdev(m)
    def f(n):
        puntuacion = (n - media) / desviacion
        return (puntuacion < -3) or (puntuacion > 3)
    return f

def recibir(m):
    return list(filter(calcular_atipicos(m), m))


print(recibir(muestra))
```

```
 [4335]
```

✓ 12 s completado a las 17:50