Refactoring 1

- Estudiante: Javier López
- Problema: Problema 7

Código Inicial

```
import math

class User:
    def __init__(self, t):
        self.t = t

def cPrice(ori,dest,packw, frag, usr):
    d = calc(ori,dest)
    c = cst(d,packw,frag)
    c = dsc(c,usr)
    return c


def calc(p1, p2):
    la1, lo1, la2, lo2 = map(math.radians, [p1[0], p1[1], p2
    delta_la = la2 - la1
    delta_lo = lo2 - lo1
    a = math.sin(delta_la / 2) ** 2 + math.cos(la1) * math.c
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    R = 6371.0
    dis = R * c

    return dis

def cst(dis, packw, frag):
    if packw<=5:
        c = dis * 10
    else:
```

```
            c = dis * 10 + (packw - 5) * 2

        if frag:
            c = c * 1.5

        return c

    def dsc(c, usr):
        if usr.t == 'gold':
            c = c * 0.8
        elif usr.t == 'silver':
            c = c * 0.9
        return c

    def main():
        usr = User('gold')
        ori = (4.602, -74.065)
        dest = (40.748, -73.986)
        packw = 30
        frag = True
        c = cPrice(ori,dest,packw, frag, usr)
        print(c)

    if __name__ == "__main__":
        main()
```

Solución

```
    import math

    class User:
        def __init__(self, user_type):
            self.user_type = user_type

    def calculate_shipping_price(origin, destination, package_we
        distance = calculate_distance_between_coordinates(origi
        cost = calculate_shipping_cost(distance, package_weight,
```

```
    cost = apply_discount(cost, user)
    return cost

def haversine_distance(lat1, lon1, lat2, lon2):
    delta_lat = lat2 - lat1
    delta_lon = lon2 - lon1
    a = math.sin(delta_lat / 2) ** 2 + math.cos(lat1) * mat
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    earth_radius = 6371.0
    distance = earth_radius * c
    return distance

def calculate_distance_between_coordinates(point1, point2):
    lat1, lon1, lat2, lon2 = map(math.radians, [point1[0], p
    return haversine_distance(lat1, lon1, lat2, lon2)

def calculate_shipping_cost(distance, package_weight, is_fra
    if package_weight <= 5:
        cost = distance * 10
    else:
        cost = distance * 10 + (package_weight - 5) * 2

    if is_fragile:
        cost = cost * 1.5

    return cost

def apply_discount(cost, user):
    if user.user_type == 'gold':
        cost = cost * 0.8
    elif user.user_type == 'silver':
        cost = cost * 0.9
    return cost

def main():
    user = User('gold')
    origin = (4.602, -74.065)
    destination = (40.748, -73.986)
    package_weight = 30
    is_fragile = True
    shipping_price = calculate_shipping_price(origin, desti
```

```
    print(shipping_price)

if __name__ == "__main__":
    main()
```

Refactorings Utilizados

Change Function Declaration

Se cambian los nombres de prácticamente todas las funciones para que sean descriptivas tanto en su nombre como en sus parámetros. Se realizó debido a que casi todas las variables estaban nombradas con abreviaciones y eran muy difíciles de entender. La efectividad fue alta pues ahora es claro en el código que se está haciendo.

Extract Function

Se realizó la extracción de la función `harversine_distance` de la función `calculate_distance_between_coordinates` debido a que esta última era muy compleja y no se entendía que estaba sucediendo. Considero que mejora mucho la mantenibilidad del código.