Fundamentos de Programación

Análisis del consumo eléctrico de viviendas

Se quieren analizar los datos de los consumos eléctricos de las viviendas que son servidas por una determinada empresa distribuidora de electricidad durante el pasado año 2023. Para ello, se van a programar una serie de funciones en Python, en un proyecto con la siguiente estructura:

- **/src**: Contiene los diferentes módulos de Python que conforman el proyecto.
  - **consumo_electrico.py**: Contiene funciones para explotar los datos de facturación.
  - **consumo_electrico_test.py**: Contiene funciones de test para probar las funciones del módulo `consumo_electrico.py`. En este módulo está el main.
  - Puede añadir otros módulos para funciones auxiliares si así lo desea.
- **/data**: Contiene el dataset del proyecto
  - **consumos_sevilla_2023.csv**: Archivo con las facturas de las viviendas.

Ejercicios a realizar

La distribuidora ofrece a sus clientes dos tipos de tarifa:

- En la tarifa **única**, el precio del consumo eléctrico es igual a lo largo de todo el día.
- En la tarifa por **tramos**, hay dos precios del consumo eléctrico, uno para el consumo en el denominado horario "punta" (suele coincidir con las horas del día en las que hay un mayor consumo medio de energía) y otro precio para el consumo en el denominado horario "valle" (suele coincidir con las horas del día en las que hay un menor consumo medio de energía).

El precio cobrado por kWh de consumo, ya sea el de la tarifa única o los precios de cada horario en la tarifa por tramos, se mantiene fijo cada mes, y es el mismo para todos los clientes. Cada cliente mantiene un mismo tipo de tarifa durante todo el año. Además del consumo, en cada factura también se cobra el coste de la potencia contratada (una cantidad fija por cliente).

Para analizar los datos de los consumos eléctricos, la compañía dispone de un archivo en formato CSV codificado en UTF-8. En cada línea del archivo se recoge la siguiente información: un identificador de la vivienda, el tipo de vivienda (`"Apartamento"` o `"Casa"`), el barrio en el que está situado la vivienda, el tipo de tarifa contratada (`"única"` o `"tramos"`), las fechas de inicio y fin del periodo de facturación, el coste de la potencia contratada en euros, el consumo en horario punta y valle en kWh, el precio en euros por kWh, y el importe total en euros. Se muestran a continuación un par de líneas de ejemplo del archivo (no son líneas consecutivas):

```
id_vivienda,tipo_vivienda,barrio,tipo_tarifa,periodo_inicio,periodo_fin,coste_pote
ncia_eur,consumo_punta_kwh,consumo_valle_kwh,precio_kwh,importe_total_eur
001,Apartamento,Triana,tramos,2023-01-01,2023-01-
31,49.73,101.33,94.57,0.26/0.13,88.37
002,Casa,Nervión,única,2023-02-01,2023-02-28,56.93,106.21,115.15,0.17,94.56
```

Observe que el precio por kWh puede ser único o diferenciado en horario punta y valle: en el caso de la tarifa única, el precio es el mismo para ambos horarios (aparece un sólo número real en ese campo), y es distinto en el caso de la tarifa por tramos (aparecen dos números reales, correspondientes a los precios del horario punta y del horario valle, separados por el carácter `/`).

Utilice obligatoriamente la `NamedTuple` **Factura** que se define a continuación. Observe que en esta tupla se almacenarán siempre de manera diferenciada el precio por kWh para el horario punta y para el horario valle (en el caso de que la tarifa sea única, ambos campos almacenarán un mismo valor).

```
from typing import NamedTuple
from datetime import date

IntervaloFechas = NamedTuple("IntervaloFechas",
                     [("inicio", date), ("fin", date)])

Factura = NamedTuple("Factura",
                     [("id_vivienda", str),
                      ("tipo_vivienda", str),
                      ("barrio", str),
                      ("tipo_tarifa", str),
                      ("periodo_facturado", IntervaloFechas),
                      ("coste_potencia", float),
                      ("consumo_punta", float),
                      ("consumo_valle", float),
                      ("precio_punta", float),
                      ("precio_valle", float),
                      ("importe_total", float)])
```

Se pide implementar las siguientes funciones (en el módulo `consumo_electrico.py`) y sus tests correspondientes (en el módulo `consumo_electrico_test.py`). Las puntuaciones indicadas para cada ejercicio incluyen la realización de dichos tests. Se muestran salidas posibles para los tests de cada ejercicio. Tenga en cuenta que se pueden definir funciones auxiliares cuando se considere necesario, y que en la implementación de algunas de las funciones solicitadas puede ser oportuno utilizar funciones implementadas en ejercicios anteriores.

1. **lee_facturas**: recibe la ruta de un fichero CSV y devuelve una lista de tuplas de tipo Factura conteniendo todos los datos almacenados en el fichero. La lista devuelta debe estar ordenada por el campo `periodo_facturado.inicio`. (1 punto)

   ```
   def lee_facturas(ruta_fichero: str) -> List[Factura]
   ```

   ```
   Test lee_facturas
   Total facturas: 1200
   Mostrando las dos primeras y las dos últimas:
   Factura(id_vivienda='001', tipo_vivienda='Apartamento', barrio='Triana',
   tipo_tarifa='tramos',
   ```

```
periodo_facturado=IntervaloFechas(inicio=datetime.date(2023, 1, 1),
fin=datetime.date(2023, 1, 31)), coste_potencia=49.73, consumo_punta=101.33,
consumo_valle=94.57, precio_punta=0.26, precio_valle=0.13,
importe_total=88.37)
Factura(id_vivienda='002', tipo_vivienda='Casa', barrio='Nervión',
tipo_tarifa='única',
periodo_facturado=IntervaloFechas(inicio=datetime.date(2023, 1, 1),
fin=datetime.date(2023, 1, 31)), coste_potencia=56.93, consumo_punta=114.32,
consumo_valle=112.61, precio_punta=0.2, precio_valle=0.2,
importe_total=102.32)
...
Factura(id_vivienda='099', tipo_vivienda='Casa', barrio='Este-Alcosa-
Torreblanca', tipo_tarifa='tramos',
periodo_facturado=IntervaloFechas(inicio=datetime.date(2023, 12, 1),
fin=datetime.date(2023, 12, 31)), coste_potencia=68.8, consumo_punta=132.09,
consumo_valle=133.04, precio_punta=0.26, precio_valle=0.15,
importe_total=123.1)
Factura(id_vivienda='100', tipo_vivienda='Apartamento', barrio='Bellavista-
La Palmera', tipo_tarifa='única',
periodo_facturado=IntervaloFechas(inicio=datetime.date(2023, 12, 1),
fin=datetime.date(2023, 12, 31)), coste_potencia=68.13,
consumo_punta=134.02, consumo_valle=106.22, precio_punta=0.15,
precio_valle=0.15, importe_total=104.17)
```

2. **extrae_precio_por_mes**: recibe una lista de facturas y un tipo de tarifa, y devuelve un diccionario en el que las claves son cadenas `"año-mes"` (por ejemplo, `"2023-01"`) y los valores son tuplas con el precio por kWh en horario punta y valle para dicho tipo de tarifa de cada mes. Tenga en cuenta que el precio del kWh es igual para todos los clientes del mismo tipo de contrato para cada mes. (1 punto)

   Nota: Para obtener la cadena `"año-mes"` puede utilizar el método `strftime` del tipo `date`, pasándole la cadena de formato `"%Y-%m"`.

```
def extrae_precio_por_mes(facturas: List[Factura], tipo_tarifa: str) ->
Dict[str, Tuple[float, float]]
```

```
Test extrae_precio_por_mes
(Se muestra el resultado obtenido para dos llamadas a la función)

Precio por mes (tarifa única)
    2023-01: (0.2, 0.2)
    2023-02: (0.17, 0.17)
    2023-03: (0.19, 0.19)
    2023-04: (0.25, 0.25)
    2023-05: (0.14, 0.14)
    2023-06: (0.16, 0.16)
    2023-07: (0.15, 0.15)
    2023-08: (0.15, 0.15)
    2023-09: (0.26, 0.26)
    2023-10: (0.24, 0.24)
```

```
    2023-11: (0.21, 0.21)
    2023-12: (0.15, 0.15)

Precio por mes (tarifa por tramos)
    2023-01: (0.26, 0.13)
    2023-02: (0.3, 0.11)
    2023-03: (0.24, 0.1)
    2023-04: (0.21, 0.11)
    2023-05: (0.27, 0.14)
    2023-06: (0.21, 0.15)
    2023-07: (0.23, 0.15)
    2023-08: (0.18, 0.12)
    2023-09: (0.17, 0.18)
    2023-10: (0.3, 0.16)
    2023-11: (0.28, 0.14)
    2023-12: (0.26, 0.15)
```

3. **busca_vivienda_mayor_consumo_acumulado**: recibe una lista de facturas y devuelve una tupla con el identificador de la vivienda con mayor consumo acumulado, y el valor de dicho consumo acumulado. El consumo acumulado de una vivienda es la suma de los consumos tanto de horario punta como de horario valle de todas las facturas de esa vivienda contenidas en la lista recibida. (1 punto)

```
def busca_vivienda_mayor_consumo_acumulado(facturas: List[Factura]) ->
Tuple[str, float]
```

```
Test busca_vivienda_mayor_consumo_acumulado
La vivienda con mayor consumo acumulado es la 056 con un consumo total de
5033.36 kWh.
```

4. **barrios_mayor_consumo_valle_medio**: recibe una lista de facturas y un entero `top_n`, y devuelve una lista con los `top_n` barrios con mayor consumo medio en horario valle. (1,5 puntos)

```
def barrios_mayor_consumo_valle_medio(facturas: List[Factura], top_n: int) -
> List[str]
```

```
Test barrios_mayor_consumo_valle_medio
Los tres barrios con mayor consumo medio en horario valle son:
    San Pablo-Santa Justa
    Sur
    Triana
```

5. **compara_importe_tipos_factura**: recibe una lista de facturas y un identificador de vivienda, y permite comparar el importe total pagado por esa vivienda con el que se hubiera pagado con el otro tipo de

tarifa. Devuelve una tupla con el cambio de tipo de contrato contemplado (`"tramos->única"` o `"única->tramos"`), el importe total actual y el importe total que se habría facturado con el cambio de tarifa. Si no se encontrara la vivienda indicada en la lista de facturas, la función devuelve `None`. Recuerde cómo se calcula el importe total de una factura:

- En la tarifa **única**, se multiplica el consumo en kWh por el precio por kWh, y se le suma el coste de la potencia contratada.
- En la tarifa por **tramos**, se multiplican y suman el consumo en kWh de cada tramo (punta y valle) por el precio por kWh de cada tramo, y se le suma el coste de la potencia contratada.

Tenga en cuenta que cada vivienda mantiene un mismo tipo de tarifa en todo el periodo para el que se están analizando los datos. (1,5 puntos)

```
def compara_importe_tipos_factura(facturas: List[Factura], id_vivienda: str)
-> Optional[Tuple[str, float, float]]
```

```
Test compara_importe_tipos_factura
La vivienda 005, haciendo un cambio de tarifa 'única->tramos', habría pagado
un total de
1150.55 euros en lugar de 1153.82 si hubiera cambiado de tipo de tarifa.
```

6. **busca_cambios_beneficiosos**: recibe una lista de facturas y calcula para cuántas viviendas resulta beneficioso hacer un cambio de tarifa (es decir, hubieran pagado menos en total si hubieran tenido la otra tarifa). Devuelve una lista con el tipo de cambio de tarifa, el número de cambios beneficiosos de ese tipo encontrados y el total que habrían ahorrado por esos cambios. Por ejemplo, si la función devolviera `[('tramos->única', 33, 190.3615), ('única->tramos', 23, 127.0188)]`, significaría que se han encontrado 33 viviendas que habrían ahorrado dinero si hubieran tenido la tarifa única en lugar de la tarifa por tramos (el ahorro total habría sido de 190,3615 euros), y que se han encontrado 23 viviendas que habrían ahorrado dinero si hubieran tenido la tarifa por tramos en lugar de la tarifa única (el ahorro total habría sido de 127,0188 euros). (2 puntos)

```
def busca_cambios_beneficiosos(facturas: List[Factura]) -> List[Tuple[str,
int, float]]
```

```
Test busca_cambios_beneficiosos
Se han encontrado 33 cambios de tarifa de tipo 'tramos->única' que han
supuesto un ahorro total de 190.36 euros.
Se han encontrado 23 cambios de tarifa de tipo 'única->tramos' que han
supuesto un ahorro total de 127.02 euros.
```

7. **calcula_mes_incremento_maximo_consumo_acumulado**: recibe una lista de facturas y devuelve el `"año-mes"`(por ejemplo, `"2023-04"`) en el que se ha producido el mayor incremento en el consumo

acumulado de todas las viviendas del tipo indicado con respecto al mes anterior, y el valor de dicho consumo acumulado. Si no se indica ningún tipo de vivienda (se deja el valor `None`) se contemplarán todos los tipos de vivienda. El consumo acumulado de las viviendas de un tipo es la suma de los consumos en horario punta y valle de todas las viviendas de ese tipo. (2 puntos)

Nota: Para obtener la cadena `"año-mes"` puede utilizar el método `strftime` del tipo `date`, pasándole la cadena de formato `"%Y-%m"`.

```
def calcula_mes_incremento_maximo_consumo_acumulado(facturas: List[Factura],
tipo_vivienda: Optional[str] = None) -> Tuple[str, float]:
```

```
Test calcula_mes_incremento_maximo_consumo_acumulado
El mes con mayor incremento en el consumo acumulado fue 2023-04 con un
incremento de 142.98 kWh.
El mes con mayor incremento en el consumo acumulado para las viviendas de
tipo 'Casa' fue 2023-10 con un incremento de 85.78 kWh.
```