1ª CONVOCATORIA FP-PYTHON (COSAS A SABER)

Parsear una NamedTuple dentro de un campo de otra NamedTuple:

```python
Artista = NamedTuple("Artista",
                     [("nombre", str),
                     ("hora_comienzo", time),
                     ("cache", int)])

Festival = NamedTuple("Festival",
                     [("nombre", str),
                     ("fecha_comienzo", date),
                     ("fecha_fin", date),
                     ("estado", str),
                     ("precio", float),
                     ("entradas_vendidas", int),
                     ("artistas", List[Artista]),
                     ("top", bool)
                    ])
```

```python
#FUNCIONES AUXILIARES:
def parsea_artista(cadena: str) -> Artista:
    trozos = cadena.split("_")

    nombre = str(trozos[0].strip())
    hora_comienzo = datetime.strptime(trozos[1].strip(), "%H:%M").time()
    cache = int(trozos[2].strip())

    return Artista(nombre, hora_comienzo, cache)

def parsea_list_artistas(cadena: str) -> List[Artista]:
    if len(cadena.strip()) > 0:
        trozos = cadena.split("-")
        return [parsea_artista(trocito) for trocito in trozos]
    else:
        return []
```

Parsear una List[str] dentro de un campo de una NamedTuple:

```python
Suscripcion = NamedTuple("Suscripcion",
                         [("nombre", str),
                          ("dni", str),
                          ("fecha_inicio", date),
                          ("fecha_fin", date | None),
                          ("tipo_plan", str),
                          ("num_perfiles", int),
                          ("precio_mensual", float),
                          ("addons", List[str])
                         ])
```

```python
#FUNCIONES AUXILIARES:
def parsea_list_addons(cadena: str) -> List[str]:
    if len(cadena.strip()) > 0:
        return [c.strip() for c in cadena.split(",")]
    else:
        return []
```

Parsear un Set[str] dentro de un campo de una NamedTuple:

```python
Commit = NamedTuple("Commit",
       [("id", str), # Identificador alfanumérico del commit
        ("mensaje", str), # Mensaje asociado al commit
        ("fecha_hora", datetime) # Fecha y hora en la que se registró el commit
       ])
Repositorio = NamedTuple("Repositorio",
      [("nombre", str),  # Nombre del repositorio
       ("propietario", str), # Nombre del usuario propietario
       ("lenguajes", Set[str]),  # Conjunto de lenguajes usados
       ("privado", bool),  # Indica si es privado o público
       ("commits", List[Commit])  # Lista de commits realizados
       ])
```

```python
def parsea_set_lenguajes(cadena: str) -> Set[str]:
    res = set()

    if len(cadena.strip()) > 0:
        trozos = cadena.strip().split(",")
        res.update(trocito.strip() for trocito in trozos)
        return res

    else:
        return res
```

```python
#FUNCIONES AUXILIARES:
def parsea_commits(cadena: str) -> Commit:
    trozos = cadena.strip().split("#")

    id = str(trozos[0].strip())
    mensaje = str(trozos[1].strip())
    fecha_hora = datetime.strptime(trozos[2].strip(), "%Y-%m-%d %H:%M:%S")

    return Commit(id, mensaje, fecha_hora)

def parsea_list_commits(cadena: str) -> List[Commit]:
    cadena = cadena.strip().strip("[]")  # Elimina los corchetes del principio y final
    if len(cadena) > 0:
        trozos = cadena.split(";")
        return [parsea_commits(trocito) for trocito in trozos]
    else:
        return []
```