Ejercicio 1

```
def lee_reservas(ruta_fichero: str) -> List[Reserva]:
    with open(ruta_fichero, encoding='utf-8') as f:
        res = []
        lector = csv.reader(f)
        next(lector)

        for nombre, dni, fecha_entrada, fecha_salida, tipo_habitacion, num_personas,
                precio_noche, servicios_adicionales in lector:
            fecha_entrada = datetime.strptime(fecha_entrada, "%Y-%m-%d").date()
            fecha_salida = datetime.strptime(fecha_salida, "%Y-%m-%d").date()
            num_personas = int(num_personas)
            precio_noche = float(precio_noche)
            servicios_adicionales = servicios_adicionales.split(",")
                if servicios_adicionales else []
            res.append(Reserva(nombre, dni, fecha_entrada, fecha_salida,
                tipo_habitacion, num_personas,precio_noche, servicios_adicionales))

    return res
```

Ejercicio 2

```
def total_facturado(reservas: List[Reserva], fecha_ini: Optional[date] = None,
        fecha_fin: Optional[date] = None) -> float:

    return sum(r.precio_noche * dias(r)
                for r in reservas
                if (fecha_ini == None or fecha_ini <= r.fecha_entrada) and
                   (fecha_fin == None or r.fecha_entrada <= fecha_fin)
            )

def dias(reserva: Reserva) -> int:
    return (reserva.fecha_salida - reserva.fecha_entrada).days
```

Ejercicio 3

```
def reservas_mas_largas(reservas: List[Reserva], n: int = 3)
        -> List[Tuple[str, date]]:
    ordenadas = sorted(reservas, key = lambda r:dias(r), reverse=True)

    return [(r.nombre, r.fecha_entrada) for r in ordenadas[:n]]
```

Ejercicio 4

```
def cliente_mayor_facturacion(reservas: List[Reserva], servicios: Optional[Set[str]]
        = None) -> Tuple[str, float]:

    total_por_cliente = defaultdict(float)
    for r in reservas:
        if servicios == None or
                len(servicios.intersection(r.servicios_adicionales)) > 0:
            total_por_cliente[r.dni] += r.precio_noche * dias(r)
```

```
    return max(total_por_cliente.items(), key=lambda t:t[1])
```

Ejercicio 5

```
def servicios_estrella_por_mes(reservas: List[Reserva], tipos_habitacion:
        Optional[Set[str]] = None) -> Dict[str, str]:

    meses = ["Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio",
        "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"]

    servicios_por_mes = defaultdict(list)
    for r in reservas:
        if tipos_habitacion == None or r.tipo_habitacion in tipos_habitacion:
            nombre_mes = meses[r.fecha_entrada.month-1]
            servicios_por_mes[nombre_mes].extend(r.servicios_adicionales)
            # También se puede usar +=

    res = {}
    for mes, servicios in servicios_por_mes.items():
        res[mes] = Counter(servicios).most_common(1)[0][0]
    return res
```

Ejercicio 6

```
def media_dias_entre_reservas(reservas: List[Reserva]) -> float:
    dias = []
    reservas_ordenadas = sorted(reservas, key=lambda r:r.fecha_entrada)
    for r1, r2 in zip(reservas_ordenadas, reservas_ordenadas[1:]):
        dias.append((r2.fecha_entrada - r1.fecha_entrada).days)

    return sum(dias) / len(dias)
```

Ejercicio 7

```
def cliente_reservas_mas_seguidas(reservas: List[Reserva], min_reservas: int)
        -> str:

    reservas_por_cliente = defaultdict(list)
    for r in reservas:
        reservas_por_cliente[r.dni].append(r)

    medias_por_cliente= []
    for dni, reservas_cliente in reservas_por_cliente.items():
        if len(reservas_cliente) >= min_reservas:
            media = media_dias_entre_reservas(reservas_cliente)
            medias_por_cliente.append((dni, media))

    return min(medias_por_cliente, key=lambda t:t[1])
```