Ejercicio 1

```
def lee_festivales (archivo: str) -> List[Festival]:
    res = []
    with open(archivo, encoding= "utf-8") as f:
        lector = csv.reader(f)
        next(lector)
        for nombre, fecha_ini, fecha_fin, estado, precio, entradas, artistas, top \
               in lector:
            fecha_ini = parsea_fecha(fecha_ini)
            fecha_fin = parsea_fecha(fecha_fin)
            precio = float(precio.strip())
            entradas = int(entradas.strip())
            artistas = parsea_artistas(artistas.strip())
            top = parsea_top(top.strip())
            festival=Festival(nombre.strip(),
                              fecha_ini,
                              fecha_fin,
                              estado.strip(),
                              precio,
                              entradas,
                              artistas,
                              top)
            res.append(festival)

    return sorted(res, key = lambda f: f.fecha_comienzo)

def parsea_fecha(fecha_str: str) -> date:
    return datetime.strptime(fecha_str, "%Y-%m-%d").date()

def parsea_artistas(artistas_str: str) -> List[Artista]:
    trozos = artistas_str.split("-")
    return [parsea_artista(trozo) for trozo in trozos]

def parsea_artista(artista_str: str) -> Artista:
    trozos = artista_str.split("_")
    nombre = trozos[0].strip()
    hora = parsea_hora(trozos[1].strip())
    cache = int(trozos[2].strip())
    return Artista(nombre, hora, cache)

def parsea_hora(hora_str: str) -> time:
    return datetime.strptime(hora_str, "%H:%M").time()

def parsea_top(top_str: str) -> bool:
    res = None
    if (top_str == "sí"):
        res = True
    elif (top_str == "no"):
        res = False
    return res
```

Ejercicio 2

```
def total_facturado(festivales: List[Festival], fecha_ini: Optional[date] = None,
                    fecha_fin: Optional[date] = None) -> float:
    return sum(facturacion(festival)
                    for festival in festivales
                        if festival.estado == "CELEBRADO"
                           and en_fecha(festival, fecha_ini, fecha_fin))
```

```
def facturacion (festival: Festival) -> float:
    return festival.entradas_vendidas * festival.precio

def en_fecha (festival: Festival, fecha_ini: Optional[date] = None,
              fecha_fin: Optional[date] = None) -> bool:
    return (fecha_ini == None or fecha_ini <= festival.fecha_comienzo) \
                and (fecha_fin == None or festival.fecha_fin <= fecha_fin)
```

Ejercicio 3

```
def artista_top(festivales: List[Festival]) -> Tuple[int, str]:

    c = contar_festivales_por_artista (festivales)

    m = max(c.items(), key=lambda it:it[1] )
    return (m[1], m[0])

def contar_festivales_por_artista(festivales: List[Festival]) -> Dict[str, int]:
    return Counter(artista.nombre for festival in festivales
                        for artista in festival.artistas
                            if festival.estado == 'CELEBRADO')
# alternativa:
def contar_festivales_por_artista2(festivales: List[Festival]) -> Dict[str, int]:
    res = defaultdict(int)
    for f in festivales:
        if f.estado == 'CELEBRADO':
            for a in f.artistas:
                res[a.nombre] += 1
    return res
```

Ejercicio 4

```
def mes_mayor_beneficio_medio(festivales: List[Festival]) -> str:
    d = beneficios_festivales_por_mes(festivales)
    d_medias = {mes: statistics.mean(beneficios) \
                    for mes, beneficios in d.items()}
    # alternativa:
    # d_medias = {mes: sum(beneficios) / len(beneficios) \
    #                 for mes, beneficios in d.items()}

    return max(d_medias.keys(), key = d_medias.get)

    # alternativa:
    # return max(d_medias.items(), key = lambda it:it[1])[0]

def beneficios_festivales_por_mes(festivales: List[Festival]) -> Dict[str, List[float]]:
    d = defaultdict(list)
    for festival in festivales:
        mes = mes_str(festival.fecha_comienzo)
        d[mes].append(calcula_beneficio(festival))
    return d

def mes_str(fecha: date) -> str:
    meses = ["Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", \
             "Agosto","Septiembre", "Octubre", "Noviembre", "Diciembre"]
    return meses[fecha.month - 1]

def calcula_beneficio (festival: Festival) -> float:
    cache_total = sum(artista.cache * 1000 for artista in festival.artistas)
    return festival.entradas_vendidas * festival.precio - cache_total
```

Ejercicio 5

```
def artistas_comunes(festivales: List[Festival], festi1: str, festi2: str)
                                                                    -> List[str]:
    a1 = a2 = set()
    for festival in festivales:
        if f.nombre == festi1:
            a1 = {artista.nombre for artista in festival.artistas}
        if f.nombre == festi2:
            a2 = {artista.nombre for artista in festival.artistas}
    return list(a1.intersection(a2))

# alternativa, suponiendo que el festival no se repite:
def artistas_comunes2(festivales: List[Festival], festi1: str, festi2: str)
    d = defaultdict(set)
    for festival in festivales:
        d[festival.nombre] = {artista.nombre for artista in festival.artistas}

    artistas1 = d.get(festi1, set())
    artistas2 = d.get(festi2, set())
    return list(artistas1.intersection(artistas2))
```

Ejercicio 6

```
def festivales_top_calidad_por_duracion(festivales: List[Festival],
                    n: Optional[int] = 3) -> Dict[int, List[str]]:
    d = festivales_por_duracion (festivales)
    return {duracion: festivales_top_calidad(lista_festivales, n) \
                for duracion, lista_festivales in d.items()}

def festivales_por_duracion(festivales: List[Festival]) -> Dict[int, List[Festival]]:
    res = defaultdict(list)
    for fest in festivales:
        duracion = (fest.fecha_fin - fest.fecha_comienzo).days
        res[duracion].append(fest)
    return res

def festivales_top_calidad(festivales: List[Festival], n: int = 3) -> List[str]:
    festivales_ord = sorted(festivales, key = lambda festival: calidad(festival),
                                reverse = True)[:n]
    return[festival.nombre for festival in festivales_ord]

def calidad(festival: Festival) -> float:
    return festival.entradas_vendidas / len(festival.artistas)
```

Test

```
def test_lee_festivales (festivales: List[Festival]) -> None:
    print("Test lee_festivales")
    print(f"Total festivales: {len(festivales)}")
    print("Los tres primeros:")
    print(festivales[:2])

def test_total_facturado(festivales: List[Festival], fecha_ini: Optional[date] = None,
                                    fecha_fin: Optional[date] = None) -> None:
    res = total_facturado(festivales, fecha_ini, fecha_fin)
    print(f"Entre {fecha_ini} y {fecha_fin} el total es: {res}")

def test_artista_top(festivales: List[Festival]) -> None:
    num_festivales, artista = artista_top(festivales)
```

```
    print(f"El artista que ha actuado en más festivales es {artista}, con {num_festivales} festiva
les.")

def test_mes_mayor_beneficio_medio(festivales: List[Festival]) -> None:
    res = mes_mayor_beneficio_medio(festivales)
    print(f"El mes con más beneficio medio es {res}")

def test_artistas_comunes(festivales: List[Festival], festi1: str, festi2: str) -> None:
    res = artistas_comunes(festivales, festi1, festi2)
    print(f"Los artistas comunes entre {festi1} y {festi2} son {res}")

def test_festivales_top_calidad_por_duracion(festivales: List[Festival], n: int = 3) -> None:
    res = festivales_top_calidad_por_duracion(festivales, n)
    print(f"Los mejores {n} festivales por número de días son:")
    for duracion, festivales in res.items():
        print(f"{duracion}: {festivales}")

if __name__=="__main__":

    festivales = lee_festivales("data\\festivales.csv")
    test_lee_festivales(festivales)

    print("Test Ej2: total_facturado")
    test_total_facturado(festivales, None, None)
    test_total_facturado(festivales, fecha_fin=date(2024, 6, 15))
    test_total_facturado(festivales, fecha_ini=date(2024, 6, 15))
    test_total_facturado(festivales, date(2024, 6, 1), date(2024, 6, 15))
    test_total_facturado(festivales, date(2024, 6, 1), date(2024, 6, 23))


    print("Test Ej3: artista_top")
    test_artista_top(festivales)

    print("Test Ej4: mes_mayor_beneficio_medio")
    test_mes_mayor_beneficio_medio(festivales)

    print("Test Ej5: artistas_comunes")
    test_artistas_comunes(festivales,"Creamfields", "Tomorrowland")
    test_artistas_comunes(festivales,"Primavera Sound", "Coachella")
    test_artistas_comunes(festivales,"Iconica Fest", "Primavera Sound")

    print("Test Ej6: festivales_top_mejor_ratio")
    test_festivales_top_calidad_por_duracion(festivales, 1)
    test_festivales_top_calidad_por_duracion(festivales, 3)
```