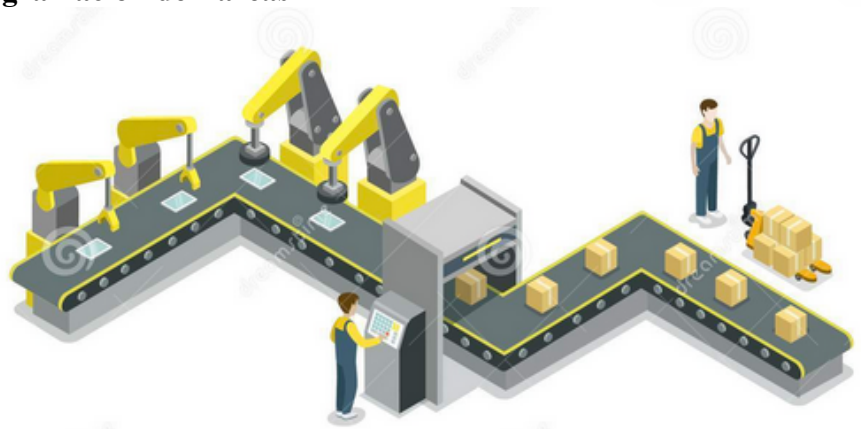
Ciclo 1 Fundamentos de Programación

Reto 3: Programación de Tareas



Una línea de producción de electrodomésticos, cuenta con una serie de máquinas que deben realizar las tareas establecidas para la construcción de los productos. Estas tareas deben ser atendidas en un momento específico, debido a los materiales utilizados y procesos que se realizan durante el ensamblado de dichos electrodomésticos. La fábrica cuenta con 20 máquinas, por lo tanto, está en capacidad de procesar 20 tareas de forma simultánea, sin embargo, se busca utilizar el menor número posible de máquinas y no ocuparlas por más de media jornada (4 horas o 240 minutos), con el propósito de disminuir el desgaste de estas, y poder realizar durante el resto de la jornada tareas de mantenimiento. Consecuentemente, se requiere monitorear la ocupación promedio de los equipos, para tomar decisiones operativas con respecto a la ampliación o disminución de la cantidad de tareas que se atenderán en el siguiente listado de solicitudes.

Se requiere entonces implementar una función que distribuya dichas tareas en las máquinas, con la siguiente estrategia ágil establecida por el asesor de la planeación operativa de la empresa productora:

```
(1) Abrir programación para todas las máquinas.
(2) Abrir el primer itinerario de toda la programación.
(3) Para cada una de las tareas del listado recibido:
(4)     Si la tarea actual puede agregarse al itinerario iniciado, es decir,
        si cumple de manera simultánea con las siguientes condiciones:
        - Agregando la tarea, el tiempo de ocupación es menor o igual a 240 min.
        - El tiempo de finalización de la última tarea en el itinerario es
          menor o igual al tiempo de inicio de la tarea actual.
(5)             Adicionar la tarea al final del itinerario actual.
        De lo contrario:
(6)             Cerrar el itinerario actual agregándolo a la programación.
(7)             Iniciar un nuevo itinerario con la tarea actual.
(8) Si del recorrido de las tareas quedó un itinerario en construcción:
(9)     Adicionarlo a la programación.
```

Algoritmo 1. Estrategia ágil de distribución de tareas.

Las tareas son recibidas mediante un listado donde están consignadas respetando un orden cronológico de menor a mayor, dependiendo del momento del día en el que inician. Para facilitar la representación de la hora en la que deben ser realizadas las tareas, las horas del día se expresarán estrictamente en minutos, por lo tanto, la jornada va desde el minuto 0 hasta el minuto 1440. Por ejemplo, las 5:00 am corresponderá al minuto 300 del día, y las 8:00 pm corresponderán al minuto 1200 del día. El listado de tareas entonces, en cada posición, alojará un diccionario con la siguiente información:

- **Hora o tiempo de inicio de la tarea**, abreviada con la llave *'t0'*, corresponde a la hora del día en minutos cuando debe iniciarse la tarea.
- **Hora o tiempo de finalización de la tarea**, abreviada con la llave *'tf'*, corresponde a la hora del día en minutos cuando debe finalizarse la tarea.
- **Duración de la tarea**, abreviada con la llave *'duracion'* (se omiten tildes en las llaves de los diccionarios para prevenir problemas de codificación), corresponde al tiempo que consume la realización de la tarea.
- **Identificador de la tarea**, abreviado con la llave *'id'*, corresponde a la etiqueta que se ha establecido para esta tarea al interior de la línea de producción.

Durante la generación de los itinerarios, recorriendo cada una de las tareas, como se especifica en el Algoritmo 1, se realiza una revisión de factibilidad. Esto último, quiere decir que si la última tarea registrada en el itinerario que se está construyendo para una de las máquinas, termina antes de que inicie la tarea actual, podrá agregarse esa tarea en el itinerario que se está construyendo, disminuyendo la ocupación general de los equipos con los que cuenta la empresa. En la Figura 1 se ilustra una situación factible, es decir, cuando no hay traslape entre el tiempo de finalización (*'tf'*) de la Tarea2 y el tiempo de inicio (*'t0'*) de la Tarea4, generando un itinerario parcial 'Tarea2-Tarea4' con una duración parcial de 148 minutos, teniendo la posibilidad de incorporar tareas que no se traslapen y que duren como máximo 92 minutos, dado que la ocupación máxima permitida es de 240 minutos.

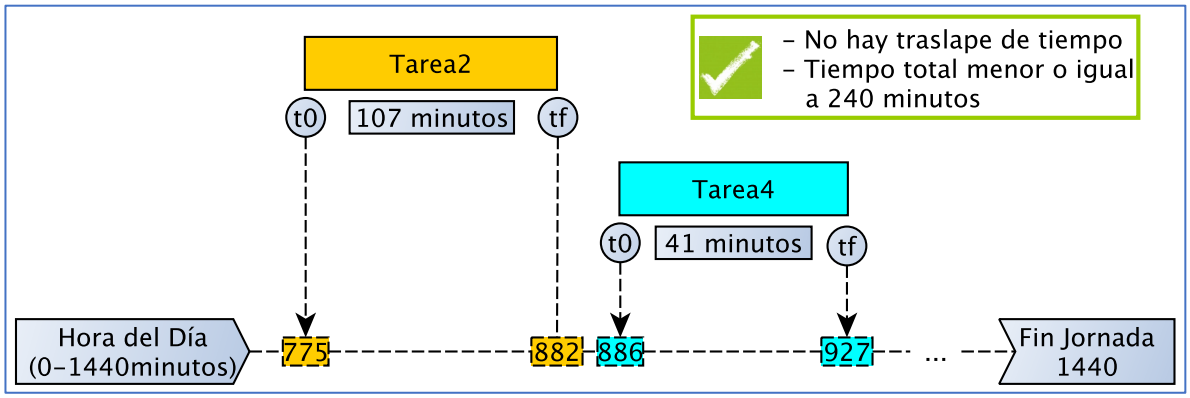


Figura 1. Incorporación de tarea factible: sin traslape de tiempo y tiempo máximo respetado. Nuevos servicios podrían incorporarse.

En la Figura 2, se ilustra una situación de infactibilidad al intentar incorporar la Tarea19 al itinerario parcial que contiene la Tarea3, dado que **el tiempo de finalización *'tf'* de la Tarea3 no es menor o igual al tiempo de inicio *'t0'* de la Tarea19**. Adicionalmente, la duración de ambas tareas presenta un tiempo acumulado de 346 minutos, el cual **no es menor o igual a 240 minutos.**

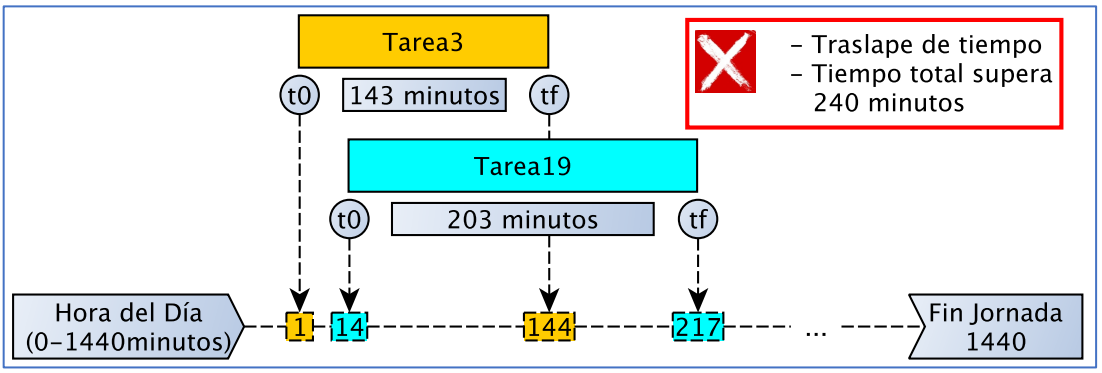


Figura 2. Incorporación de tarea infactible: traslape de tiempo y tiempo máximo incumplidos.

La construcción de los itinerarios es necesaria, sin embargo, de la función que se debe implementar, se espera un diccionario que contenga el **número de máquinas que van a ser ocupadas** y el **tiempo de ocupación promedio**, dado que es una función orientada al apoyo de la toma de decisiones.

Partiendo de lo anterior, a continuación se especifican los detalles técnicos del requerimiento. En primera medida se presenta el esqueleto de la función, indicado en la Figura 3, la cual, recibe una lista de diccionarios que contiene la especificación de las tareas, y retorna un diccionario con el número de máquinas ocupadas o itinerarios generados y el tiempo de ocupación promedio de estas.

```
def programadorTareas(listaTareas : list) -> dict:
    "Desarrollar el algoritmo de programación de tareas"
    pass
```

Figura 3. Prototipo de la función

En la siguiente tabla, se especifican los parámetros (campos, atributos o ítems) que llegan encapsulados en el diccionario alojado en cada una de las posiciones de **listaTareas**, única entrada de la función solicitada**.** Los nombres de dichos parámetros y sus contenidos no tienen tildes, previniendo problemas de codificación como se mencionó anteriormente.

Especificación de la lista de entrada

Ítem	Tipo de Dato	Descripción
listaTareas	list	Lista con los diccionarios que contienen la especificación de cada una de las tareas.

listaTareas[i][‘t0’]	int	Valor entero mayor o igual a cero con la hora de inicio de la tarea i en minutos.
listaTareas[i][‘tf’]	int	Valor entero mayor o igual a cero con la hora de inicio de la tarea i en minutos.
listaTareas[i][‘duracion’]	int	Valor entero mayor o igual a cero con la duración en minutos de la tarea i. Este campo también se puede obtener con la diferencia de la hora de finalización y la hora de inicio de la tarea i.
listaTareas[i][‘id’]	str	Cadenas que contiene la etiqueta o nombre que se le ha asignado a la tarea i al interior de la organización.

Estructura de la Salida

Tipo de Retorno	Descripción	Ejemplo
dict	Diccionario con el número de itinerarios generados (valor entero o int) y la ocupación promedio de las máquinas (valor flotante o float redondeado a dos decimales con función round).	{'maquinasOcupadas': 15, 'ocupacionPromedio': 122.6}

A continuación se presentan 3 casos de prueba (ejemplos), mostrando cómo debe responder la función solicitada a unas entradas específicas:

Caso de Prueba 1:

Llamado:

```
print(programadorTareas([{'t0': 1, 'tf': 144, 'duracion': 143, 'id': 'Tarea3'},
{'t0': 14, 'tf': 217, 'duracion': 203, 'id': 'Tarea19'}, {'t0': 60, 'tf': 212,
'duracion': 152, 'id': 'Tarea6'}, {'t0': 97, 'tf': 118, 'duracion': 21, 'id':
'Tarea5'}, {'t0': 134, 'tf': 365, 'duracion': 231, 'id': 'Tarea20'}, {'t0': 143,
'tf': 272, 'duracion': 129, 'id': 'Tarea8'}, {'t0': 220, 'tf': 255, 'duracion':
35, 'id': 'Tarea9'}, {'t0': 231, 'tf': 400, 'duracion': 169, 'id': 'Tarea10'},
{'t0': 324, 'tf': 444, 'duracion': 120, 'id': 'Tarea11'}, {'t0': 405, 'tf': 465,
'duracion': 60, 'id': 'Tarea12'}, {'t0': 410, 'tf': 462, 'duracion': 52, 'id':
'Tarea13'}, {'t0': 414, 'tf': 463, 'duracion': 49, 'id': 'Tarea14'}, {'t0': 554,
'tf': 615, 'duracion': 61, 'id': 'Tarea15'}, {'t0': 632, 'tf': 691, 'duracion':
59, 'id': 'Tarea16'}, {'t0': 656, 'tf': 726, 'duracion': 70, 'id': 'Tarea17'},
{'t0': 711, 'tf': 765, 'duracion': 54, 'id': 'Tarea18'}, {'t0': 766, 'tf': 791,
'duracion': 25, 'id': 'Tarea1'}, {'t0': 775, 'tf': 882, 'duracion': 107, 'id':
'Tarea2'}, {'t0': 886, 'tf': 927, 'duracion': 41, 'id': 'Tarea4'}, {'t0': 997,
'tf': 1055, 'duracion': 58, 'id': 'Tarea7'}]))
```

Resultado esperado:

```
{'maquinasOcupadas': 15, 'ocupacionPromedio': 122.6}
```

Caso de Prueba 2:

Llamado:

```
print(programadorTareas([{'t0': 4, 'tf': 65, 'duracion': 61, 'id': 'Proceso19'},
{'t0': 6, 'tf': 68, 'duracion': 62, 'id': 'Proceso20'}, {'t0': 272, 'tf': 296,
'duracion': 24, 'id': 'Proceso6'}, {'t0': 350, 'tf': 388, 'duracion': 38, 'id':
'Proceso7'}, {'t0': 375, 'tf': 393, 'duracion': 18, 'id': 'Proceso10'}, {'t0':
385, 'tf': 405, 'duracion': 20, 'id': 'Proceso12'}, {'t0': 435, 'tf': 455,
'duracion': 20, 'id': 'Proceso5'}, {'t0': 566, 'tf': 582, 'duracion': 16, 'id':
'Proceso9'}, {'t0': 646, 'tf': 666, 'duracion': 20, 'id': 'Proceso11'}, {'t0':
688, 'tf': 707, 'duracion': 19, 'id': 'Proceso3'}, {'t0': 692, 'tf': 709,
'duracion': 17, 'id': 'Proceso4'}, {'t0': 725, 'tf': 742, 'duracion': 17, 'id':
'Proceso8'}, {'t0': 756, 'tf': 780, 'duracion': 24, 'id': 'Proceso13'}, {'t0':
759, 'tf': 787, 'duracion': 28, 'id': 'Proceso16'}, {'t0': 773, 'tf': 792,
'duracion': 19, 'id': 'Proceso1'}, {'t0': 832, 'tf': 849, 'duracion': 17, 'id':
'Proceso2'}, {'t0': 871, 'tf': 888, 'duracion': 17, 'id': 'Proceso18'}, {'t0':
922, 'tf': 945, 'duracion': 23, 'id': 'Proceso14'}, {'t0': 1008, 'tf': 1026,
'duracion': 18, 'id': 'Proceso17'}, {'t0': 1083, 'tf': 1100, 'duracion': 17,
'id': 'Proceso15'}]))
```

Resultado esperado:

```
{'maquinasOcupadas': 7, 'ocupacionPromedio': 70.71}
```

Caso de Prueba 3:

Llamado:

```
print(programadorTareas([{'t0': 3, 'tf': 62, 'duracion': 59, 'id': 'Etapa18'},
{'t0': 11, 'tf': 148, 'duracion': 137, 'id': 'Etapa19'}, {'t0': 37, 'tf': 147,
'duracion': 110, 'id': 'Etapa20'}, {'t0': 80, 'tf': 111, 'duracion': 31, 'id':
'Etapa7'}, {'t0': 202, 'tf': 224, 'duracion': 22, 'id': 'Etapa16'}, {'t0': 346,
'tf': 378, 'duracion': 32, 'id': 'Etapa2'}, {'t0': 401, 'tf': 452, 'duracion':
51, 'id': 'Etapa5'}, {'t0': 408, 'tf': 453, 'duracion': 45, 'id': 'Etapa6'},
{'t0': 468, 'tf': 516, 'duracion': 48, 'id': 'Etapa8'}, {'t0': 476, 'tf': 504,
'duracion': 28, 'id': 'Etapa9'}, {'t0': 549, 'tf': 586, 'duracion': 37, 'id':
'Etapa14'}, {'t0': 678, 'tf': 729, 'duracion': 51, 'id': 'Etapa4'}, {'t0': 791,
'tf': 827, 'duracion': 36, 'id': 'Etapa10'}, {'t0': 824, 'tf': 844, 'duracion':
20, 'id': 'Etapa15'}, {'t0': 886, 'tf': 961, 'duracion': 75, 'id': 'Etapa13'},
{'t0': 890, 'tf': 906, 'duracion': 16, 'id': 'Etapa17'}, {'t0': 922, 'tf': 958,
'duracion': 36, 'id': 'Etapa1'}, {'t0': 1021, 'tf': 1082, 'duracion': 61, 'id':
```

```
'Etapa3'}, {'t0': 1102, 'tf': 1195, 'duracion': 93, 'id': 'Etapa11'}, {'t0':
1435, 'tf': 1496, 'duracion': 61, 'id': 'Etapa12'}]))
```

Resultado esperado:

```
{'maquinasOcupadas': 9, 'ocupacionPromedio': 116.56}
```

Recordar: En la plataforma debe subirse una función con **el mismo nombre**, **la misma cantidad de argumentos, y el retorno debe tener exactamente la estructura de la salida que se presenta en este documento.**