

Centro Universitario de Ciencias Exactas e Ingenierías

CUCEI



Computación Tolerante a Fallas

D06/2023B

Profesor: Michel Emanuel López Franco

Workflow managers

Gutiérrez Galán Ruben Alejandro

Código: 214798315

## Contenido

Introducción: .....	3
Prefect: .....	3
Forma de instalación: .....	3
Ejemplo básico de uso:.....	4
Ventajas de usar Prefect: .....	4
Desventajas de usar Prefect:.....	4
Algunos otros ejemplos:.....	5
Conclusiones: .....	7
Referencias:.....	7

## Introducción:

Prefect es una herramienta de orquestación del flujo de trabajo que permite a los desarrolladores crear, observar y reaccionar ante canalizaciones de datos. Es la forma más sencilla de transformar cualquier función de Python en una unidad de trabajo que pueda observarse y orquestarse. ¡Solo trae tu código Python, agrega algunos decoradores y listo!

--Prefect--

## Prefect:

Se trata de una biblioteca de código abierto que se utiliza para la orquestación y administración de flujos de trabajo (workflow) en Python. Prefect facilita la creación, planificación, programación y ejecución de flujos de trabajo de datos complejos y automatizados.

Algunas características y conceptos clave de la librería prefect incluyen:

- **Flujos de trabajo declarativos:** Prefect permite definir flujos de trabajo de manera declarativa, lo que significa que puedes describir cómo deben ejecutarse tus tareas sin preocuparte por la lógica de control y la administración de tareas en sí.
- **Control de flujo:** Puedes especificar dependencias entre tareas y gestionar el flujo de datos entre ellas.
- **Planificación inteligente:** Prefect es capaz de planificar la ejecución de tareas de manera inteligente, teniendo en cuenta la disponibilidad de recursos y las dependencias.
- **Programación y ejecución distribuida:** Puedes ejecutar tus flujos de trabajo tanto en una sola máquina como de forma distribuida en un clúster de recursos.
- **Monitoreo y administración:** Prefect proporciona herramientas para supervisar y administrar el estado de tus flujos de trabajo, lo que facilita la detección y resolución de problemas.
- **Versionado de flujos de trabajo:** Puedes versionar y gestionar tus flujos de trabajo como código, lo que facilita la colaboración y la administración del ciclo de vida.

## Forma de instalación:

Para utilizar prefect, primero debes instalarlo como una dependencia en tu entorno de Python. Puedes hacerlo utilizando pip:

```
pip install prefect
```

### Ejemplo básico de uso:

```
import prefect
from prefect import task, Flow

@task
def hello_task():
    return "Hola, mundo!"

with Flow("MiFlujoDeTrabajo") as flow:
    resultado = hello_task()

flow.run()
```

### Ventajas de usar Prefect:

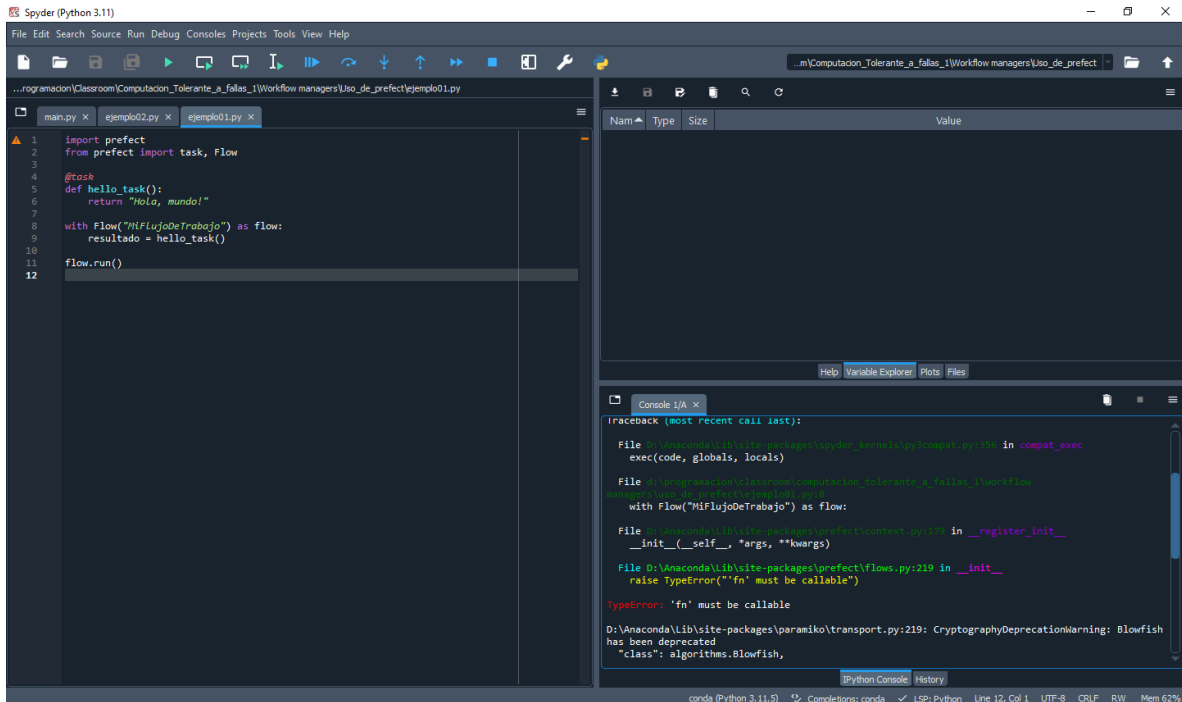
- **Declaratividad:** Prefect permite definir flujos de trabajo de manera declarativa, lo que facilita la descripción de la lógica del flujo de trabajo sin preocuparse por los detalles de implementación.
- **Flexibilidad:** Puedes crear flujos de trabajo simples o muy complejos, y Prefect se adapta a una amplia variedad de casos de uso, desde la automatización de tareas diarias hasta la gestión de flujos de datos complejos.
- **Planificación inteligente:** Prefect es capaz de planificar la ejecución de tareas de manera eficiente, teniendo en cuenta las dependencias y la disponibilidad de recursos, lo que puede ahorrar tiempo de ejecución.
- **Escalabilidad:** Prefect es adecuado para flujos de trabajo tanto en una sola máquina como en clústeres de recursos distribuidos, lo que lo hace escalable y adecuado para proyectos de todos los tamaños.
- **Control de flujo avanzado:** Ofrece una amplia gama de opciones para definir y gestionar dependencias entre tareas, lo que facilita la implementación de flujos de trabajo complejos.
- **Monitoreo y administración:** Prefect proporciona herramientas para monitorear y administrar el estado de tus flujos de trabajo, lo que facilita la detección y resolución de problemas.
- **Versiónado de flujos de trabajo:** Puedes gestionar tus flujos de trabajo como código, lo que facilita la colaboración y el seguimiento de cambios a lo largo del tiempo.

### Desventajas de usar Prefect:

- **Curva de aprendizaje:** Aunque Prefect es poderoso, puede tener una curva de aprendizaje empinada, especialmente si eres nuevo en la orquestación de flujos de trabajo o la programación distribuida.
- **Requisitos de infraestructura:** Para ejecutar flujos de trabajo distribuidos, necesitas tener acceso a una infraestructura adecuada, lo que puede ser costoso y complejo de configurar.

- **Complejidad adicional:** Algunos proyectos más simples pueden no justificar el uso de una herramienta tan completa como Prefect, lo que podría introducir una complejidad innecesaria.
- **Compatibilidad con versiones:** Dado que Prefect es una biblioteca en constante desarrollo, es importante verificar la compatibilidad de versiones al actualizar tu código.

### Algunos otros ejemplos:



The screenshot shows the Spyder Python IDE interface. The main editor displays a Python script named `ejemplo01.py` with the following code:

```
1 import prefect
2 from prefect import task, Flow
3
4 @task
5 def hello_task():
6     return "Hola, mundo!"
7
8 with Flow("MiFlujoDeTrabajo") as flow:
9     resultado = hello_task()
10
11
12 flow.run()
```

The right-hand side of the IDE shows the Variable Explorer and the Python Console. The console displays a traceback error:

```
Traceback (most recent call last):
  File .../comparacion.py:140 in comparacion
    exec(code, globals, locals)
  File .../comparacion.py:140 in comparacion
    with Flow("MiFlujoDeTrabajo") as flow:
  File .../prefect/flows.py:219 in __init__
    raise TypeError("'fn' must be callable")
TypeError: 'fn' must be callable
```

The status bar at the bottom indicates the environment is `conda (Python 3.11.5)` and the file encoding is `UTF-8`.

Se presenta el error al momento de ejecutar el programa, este está basado en los ejemplos que vienen en el material proporcionado (en los cuales funciona), pero en mi caso no funciona.

```

1 from prefect import flow, task
2 @task
3 def create_message():
4     msg="Hello from task"
5     return(msg)
6
7 @flow
8 def something_else():
9     result=10
10    return(result)
11
12 @flow
13 def Hello_World():
14     sub_flow_message=something_else()
15     task_message=create_message()
16     new_message=task_message+str(sub_flow_message)
17     print(new_message)
18
19 Hello_World()

```

```

has been deprecated
"class": algorithms.Blowfish,

In [2]: runfile('D:/Programacion/Classroom/Computacion_Tolerante_a_fallas_1/Workflow managers/
Usa_de_prefect/main.py', wdir='D:/Programacion/Classroom/Computacion_Tolerante_a_fallas_1/
Workflow managers/Usa_de_prefect')
10:05:22.399 | INFO | prefect.engine - Created flow run 'naughty-coyote' for flow 'Hello-
World'
10:05:22.655 | INFO | Flow run 'naughty-coyote' - Created subflow run 'radical-civet' for flow
'something_else'
10:05:22.809 | INFO | Flow run 'radical-civet' - Finished in state 'Completed'
10:05:22.855 | INFO | Flow run 'naughty-coyote' - Created task run 'create_message-0' for task
'create_message'
10:05:22.857 | INFO | Flow run 'naughty-coyote' - Executing 'create_message-0' immediately...
10:05:23.005 | INFO | Task run 'create_message-0' - Finished in state 'Completed'
10:05:23.055 | INFO | Flow run 'naughty-coyote' - Finished in state 'Completed'('All states
completed.')
Hello from Task10

In [3]:

```

Escrito de esta forma si funciona el programa, este se basó en otro video material, la forma en que escribió el flujo y las tareas es diferente a lo que encuentras en el material de apoyo, pero si funciona.

```

1 import os
2 import shutil
3 from prefect import Flow, task
4
5 # Define una tarea para copiar archivos
6 @task
7 def copy_files(source_folder, destination_folder):
8     try:
9         # Verifica si la carpeta de destino existe; si no, créala
10        if not os.path.exists(destination_folder):
11            os.makedirs(destination_folder)
12
13        # Lista todos los archivos en la carpeta de origen
14        files = os.listdir(source_folder)
15
16        # Copia cada archivo de la carpeta de origen a la de destino
17        for file in files:
18            source_path = os.path.join(source_folder, file)
19            destination_path = os.path.join(destination_folder, file)
20            shutil.copy2(source_path, destination_path)
21
22        return f"Copied {len(files)} files from {source_folder} to {destination_folder}"
23    except Exception as e:
24        return f"Error: {str(e)}"
25
26 # Define el flujo principal
27 with Flow("FileCopyFlow") as flow:
28     source_folder = "D:/Programacion/Classroom/Computacion_Tolerante_a_fallas_1/Workflow managers/Usa_de_prefect"
29     destination_folder = "D:/Programacion/Classroom/Computacion_Tolerante_a_fallas_1/Workflow managers/Usa_de_prefect"
30     copy_task = copy_files(source_folder, destination_folder)
31
32 # Ejecuta el flujo
33 if __name__ == "__main__":
34     flow.run()
35
36

```

```

In [1]: Traceback (most recent call last):
File ~\AppData\Local\Programs\Python\Python310\python.exe in compat_exec
exec(code, globals, locals)
File D:/Programacion/Classroom/Computacion_Tolerante_a_fallas_1/Workflow managers/Usa_de_prefect/main.py in <module>
with Flow("FileCopyFlow") as flow:
File D:/Programacion/Classroom/Computacion_Tolerante_a_fallas_1/Workflow managers/Usa_de_prefect/main.py in __init__(self, *args, **kwargs)
File D:/Programacion/Classroom/Computacion_Tolerante_a_fallas_1/Workflow managers/Usa_de_prefect/main.py in __init__(self, *args, **kwargs)
raise TypeError("'fn' must be callable")
TypeError: 'fn' must be callable

In [2]:

```

Este el programa que intento desarrollar para copiar y pasar archivos de una carpeta origen a otra, pero presenta el mismo problema que el programa de ejemplo del video de apoyo.

Nombre	Fecha de modificación	Tipo	Tamaño
Destino	02/10/2023 10:11 a. m.	Carpeta de archivos	
Origen	02/10/2023 10:12 a. m.	Carpeta de archivos	

## Conclusiones:

Se presentaron algunos inconvenientes, no me es posible el ejecutar de manera correcta los flujos y tareas propios de la librería, he consultado en diferentes fuentes para verificar que las sintaxis este correcta y en todas se usa de la misma manera sin embargo cuando creo los programas ejemplo que se proporcionan estos tienen un error de ejecución, he probado escribir de otra manera el código pero no me es posible solucionarlo se adjuntaron algunos pantallazos del error que se muestra ejecutando tanto el código que deseaba para el copiado de archivos, como un ejemplo básico y me muestra un error similar, desconozco si sea error de las versiones usadas tanto de Python como de prefect.

## Referencias:

- Kahan Data Solutions. (2023, March 15). *Getting Started with Prefect / Task Orchestration & Data Workflows* [Video]. YouTube.  
<https://www.youtube.com/watch?v=D5DhwVNHWeU>
- PyData. (2019, January 4). *Task failed successfully - Jeremiah Lowin* [Video]. YouTube. [https://www.youtube.com/watch?v=TlawR\\_gi8-Y](https://www.youtube.com/watch?v=TlawR_gi8-Y)
- Prefect. (2020, April 17). *Getting Started with Prefect (PyData Denver)* [Video]. YouTube. <https://www.youtube.com/watch?v=FETN0iivZps>
- *Welcome to Prefect - Prefect Docs*. (n.d.). <https://docs.prefect.io/2.13.4/>