CUCEI

ACTIVIDAD 06

INTRODUCCIÓN AL SEGUNDO EJEMPLO PRÁCTICO UTILIZANDO EL ORQUESTADOR DE PREFECT

MATERIA:
Computación tolerante a fallas (I7036)
NRC:
179961
SECCIÓN:
D06
MAESTRO:
López Franco, Michel Emmanuel
HORARIO:
Lunes y Miércoles, 11:00 – 12:55
FECHA DE ENTREGA:
05/02/2023
ALUMNO:
J. Emmanuel Ortiz Renteria (219747611)

Contenido

Introducción

Prefect es una biblioteca de Python diseñada para ayudar en la gestión, programación y ejecución de flujos de trabajo (workflows) de manera eficiente y confiable. Con Prefect, los desarrolladores pueden definir, organizar y ejecutar fácilmente tareas complejas en secuencia o en paralelo, lo que facilita la automatización de procesos repetitivos y la gestión de flujos de trabajo complejos.

Uso de Prefect:

- **Definición de flujos de trabajo:** Proporciona una sintaxis simple y legible para definir flujos de trabajo mediante la creación de tareas individuales y la composición de estas tareas en un flujo de trabajo más grande.
- **Programación declarativa:** Fomenta un enfoque declarativo para definir flujos de trabajo, lo que significa que los desarrolladores solo necesitan especificar la lógica de sus tareas y dejar que Prefect se encargue de la ejecución y el monitoreo.
- **Ejecución confiable:** Prefect gestiona automáticamente la ejecución de tareas, asegurándose de que se ejecuten en el orden correcto y manejen cualquier fallo o error de manera adecuada. Además, Prefect ofrece capacidades integradas de recuperación y reintentos para manejar fallos transitorios y asegurar que los flujos de trabajo continúen ejecutándose de manera confiable. Es decir,
- **Monitoreo y supervisión:** Proporciona una interfaz gráfica de usuario que permite supervisar el estado y el progreso de los flujos de trabajo en tiempo real. Esta interfaz muestra información detallada sobre el rendimiento de las tareas, los errores ocurridos y el estado general de los flujos de trabajo, lo que facilita la identificación y resolución de problemas.

Requisitos

- Python 3.2+
- Prefect 2.0.0 (Instalación: pip install prefect)
- Sistema Operativo Windows XP o superior

Objetivo

Este programa tiene como objetivo el generar datos de identidades simples, los cuales maneja al azar:

- Un nombre (Primer nombre tal vez segundo y apellidos)
- Numero de identidad
- Una fecha de nacimiento, entre el cual se comprende un margen que se cumplen entre 18 y 45 años.

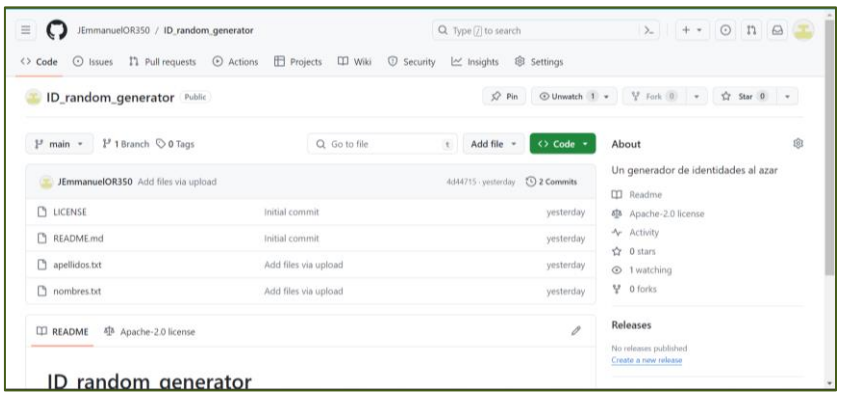
El uso indebido del programa es responsabilidad del usuario.

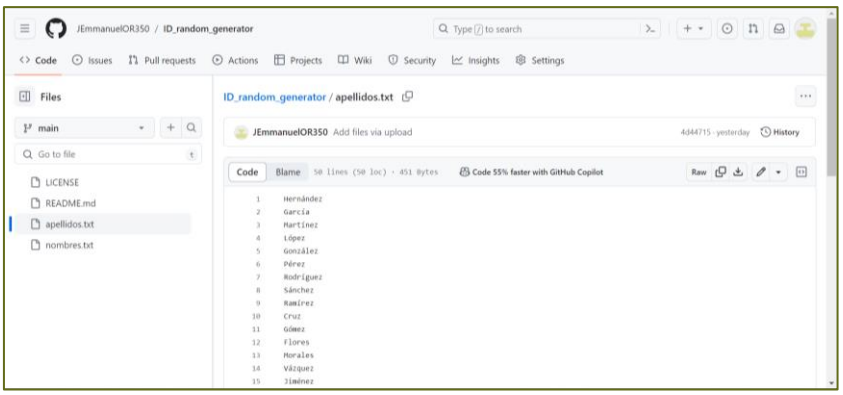
Explicación del programa

Este programa genera una identidad aleatoria constituida de un numero de ID (entre 1 y 1000), un nombre incluido apellidos, y una fecha de nacimiento. Después estos son agrupados dentro de un diccionario de datos (formato json).

El proceso se maneja de esta forma:

1. Generar un numero ID aleatorio: usando el task “generar_id_random” se genera un valor entero entre el 1 y 1000, y este es retornado dentro de una variable.
2. Generar nombre aleatorio:
    a. Existen dentro del repositorio dos archivos que contienen una lista de nombres y apellidos, esto son descargados desde Github. Y su contenido es decodificado en strings.





    b. Después usando el método “string.split()”, se crea un arreglo con strings por cada nombre y apellido.
    c. Primero se usa el método “.choice()” del arreglo de nombres para elegir un primer nombre, después usando un bloque if se hace una probabilidad de 20%, de volver a elegir un nombre al azar para concatenar un segundo nombre.
    d. Después se vuelve a usar el método “.choice()” para elegir dentro del arreglo de apellidos un candidato y concatenarlo al resultado actual del nombre. Se repite una vez mas para concatenar el apellido materno.

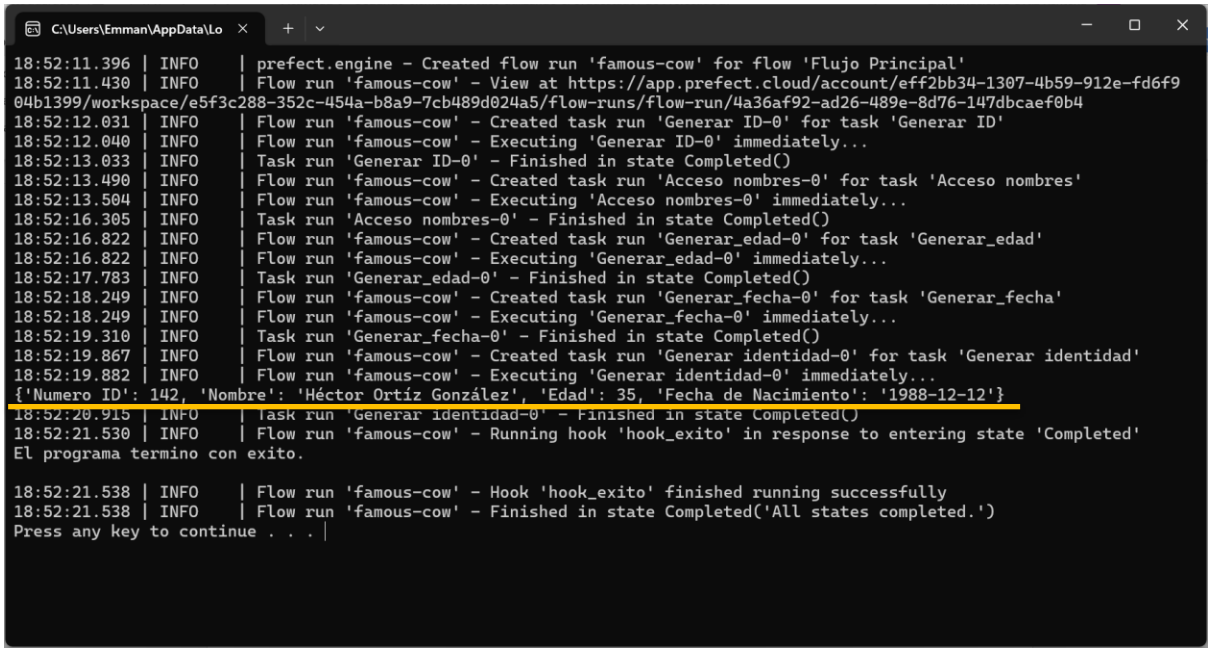
    e. Ahora la cadena donde se guarda el nombre completo es retornada.
3. Crear fecha de nacimiento:
    a. Primero se manda a llamar una función que regresa un numero entero al azar entre 18 y 45 que determina los años.
    b. Los años son retornados y se crea un objeto “date” el cual es el equivalente a la fecha actual menos los años guardados anteriormente, que representa el límite final.
    c. Después se elige al azar una fecha entre el limite final y 365 días anteriores, esa fecha es guardada y convertida en un string. Al final la fecha es retornada.
4. Empaquetar: Los datos son enviados a un task el cual le da formato json a los datos.
5. Imprimir estado del programa: el flujo de trabajo principal tiene como parámetros el uso de dos funciones en caso de que el proceso falle más allá de los reintentos indicados se imprimirá en la consola un mensaje que indica que hubo complicaciones por parte de la ejecución, mientras que si todo acabo sin problema imprime un mensaje en consola acerca del logro.

Código Fuente del programa:

```
from datetime import date, timedelta
import requests
import random
from prefect import task, flow

def hook_exito(flow, flow_run, state):
    print("El programa termino con exito.\n")

def hook_fracaso(task, flow_run, state):
    print("El proceso fracaso.\n")

@task(name="Generar ID",retries=3)
def generar_id_random():
    n_id = random.randint(1,1000)
    return n_id

@task(name="Generar_fecha",retries=3)
def fecha_random(edad):
    hoy = date.today()
    margen_anyo = int(hoy.year) - edad
    fin = date(margen_anyo,int(hoy.month),int(hoy.day))
    inicio = fin - timedelta(days=365)
    fecha = inicio + (fin - inicio) * random.random()
    fecha = str(fecha)
    return fecha

@task(name="Generar_edad",retries=3)
def edad_random():
    return random.randint(18,45)

@task(name="Acceso nombres",retries=3)
def generar_nombre():
    #Accede al repositorio y descarga el contenido de archivos de texto
    nombres =
requests.get('https://raw.githubusercontent.com/JEmmanuelOR350/ID_random_ge
nerator/main/nombres.txt', stream=True)
    apellidos =
requests.get('https://raw.githubusercontent.com/JEmmanuelOR350/ID_random_ge
nerator/main/apellidos.txt', stream=True)
    #Descodificar el contenido
    nombres=nombres.content.decode("utf-8")
    apellidos=apellidos.content.decode("utf-8")
    nombres=nombres.split()
    apellidos=apellidos.split()

    #Generar el nombre
    nombre_completo = random.choice(nombres)
    if((random.randint(1,100) <= 20)):
        nombre_completo = nombre_completo +" "+ str(random.choice(nombres))
    nombre_completo = nombre_completo +" "+ random.choice(apellidos)
    nombre_completo = nombre_completo +" "+ random.choice(apellidos)
    return nombre_completo

def new_func():
    estado = ValueError()

@task(name="Generar identidad",retries=3)
def generar_identidad(n_id,nombre_com,edad,fecha):
    identidad = {"Numero ID":n_id,"Nombre":nombre_com,"Edad":edad,"Fecha de
```

```
Nacimiento":fecha}
    print(identidad)
    return identidad

@flow(name="Flujo Principal", on_completion=[hook_exito],
on_failure=[hook_fracaso])
def crear_meta():
    nid = generar_id_random()
    nombre = generar_nombre()
    edad = edad_random()
    cumpleanyos = fecha_random(edad)
    generar_identidad(nid,nombre,edad,cumpleanyos)

if __name__ == '__main__':
    crear_meta()
```

Capturas del programa:



Identidad creada en el programa e impresa.

Contenido del mando de vigilancia de Flujos:

```
                                    Mar 16th, 2024
Info
Created task run 'Generar ID-0' for task 'Generar ID'
06:52:12 PM
prefect.flow_runs
Info
Executing 'Generar ID-0' immediately...
06:52:12 PM
prefect.flow_runs
Info
Finished in state Completed()
06:52:13 PM
Generar ID-0
prefect.task_runs
Info
Created task run 'Acceso nombres-0' for task 'Acceso nombres'
06:52:13 PM
prefect.flow_runs
Info
Executing 'Acceso nombres-0' immediately...
06:52:13 PM
prefect.flow_runs
Info
Finished in state Completed()
06:52:16 PM
Acceso nombres-0
prefect.task_runs
Info
Created task run 'Generar_edad-0' for task 'Generar_edad'
06:52:16 PM
prefect.flow_runs
```

```
Info
Executing 'Generar_edad-0' immediately...
06:52:16 PM
prefect.flow_runs
Info
Finished in state Completed()
06:52:17 PM
Generar_edad-0
prefect.task_runs
Info
Created task run 'Generar_fecha-0' for task 'Generar_fecha'
06:52:18 PM
prefect.flow_runs
Info
Executing 'Generar_fecha-0' immediately...
06:52:18 PM
prefect.flow_runs
Info
Finished in state Completed()
06:52:19 PM
Generar_fecha-0
prefect.task_runs
Info
Created task run 'Generar identidad-0' for task 'Generar identidad'
06:52:19 PM
prefect.flow_runs
Info
Executing 'Generar identidad-0' immediately...
06:52:19 PM
prefect.flow_runs
Info
Finished in state Completed()
06:52:20 PM
Generar identidad-0
prefect.task_runs
Info
Running hook 'hook_exito' in response to entering state 'Completed'
06:52:21 PM
prefect.flow_runs
Info
Finished in state Completed('All states completed.')
06:52:21 PM
prefect.flow_runs
Info
Hook 'hook_exito' finished running successfully
06:52:21 PM
prefect.flow_runs
```