# Prefect Task and Flow Creation Guide

## Prefect Installation and Environment Setup

### Create Python Virtual Environment & Install and setup Prefect

1. Create Python Virtual Environment:  
   python3 -m venv prefect\_env1
2. Activate the Virtual Environment:  
   .\prefect\_env1\Scripts\activate
3. Install Prefect:  
   pip install prefect
4. Check Prefect Version:  
   prefect –version
5. Start Prefect Server (Local):  
   prefect server start
6. Set Prefect API URL:  
   prefect config set PREFECT\_API\_URL=http://127.0.0.1:4200/api

## Creating a Simple Prefect Task

In Prefect, a task is a basic unit of work, often used to perform a specific operation or function. ([Write and run tasks - Prefect](https://docs.prefect.io/3.0/develop/write-tasks))

### Example Task: say\_hello

from prefect import task

@task
def say_hello(name: str):
    print(f"Hello, {name}!")


### Example Task: say\_goodbye

A close-up of text

Description automatically generated

Explanation: This task takes a name parameter and prints a greeting message.  
Usage: Define multiple tasks as needed to perform individual operations within a flow.

## Creating a Simple Prefect Flow

A flow is a collection of tasks, defining the sequence in which the tasks should be executed.

### Example Flow: hello\_goodbye\_flow A computer code with text Description automatically generated with medium confidence

Explanation: This flow is defined to call both say\_hello and say\_goodbye tasks sequentially, greeting and then saying goodbye to the name provided.

WIP --- WIP --- WIP ---

## Scheduling Prefect Flow

## 5. Prefect – Snowflake connection

### **Install prefect-snowflake (https://docs.prefect.io/integrations/prefect-snowflake/index#install-prefect-snowflake)**

The following command will install a version of prefect-snowflake compatible with your installed version of prefect. If you don’t already have prefect installed, it will install the newest version of prefect as well.

pip install "prefect[snowflake]"

### **Registering blocks (https://docs.prefect.io/integrations/prefect-snowflake/index#registering-blocks)**

Register blocks in this module to make them available for use.

prefect block register -m prefect\_snowflake

**Saving credentials to a block**

Note, to use the load method on a block, you must already have a block saved through code or saved through the UI.

Below is a walkthrough on saving a SnowflakeCredentials block through code.

1. Head over to [**https://app.snowflake.com/**](https://app.snowflake.com/).
2. Login to your Snowflake account, for example: nh12345.us-east-2.snowflake, with your username and password.
3. Use those credentials to fill replace the placeholders below.

from prefect\_snowflake import SnowflakeCredentials

credentials = SnowflakeCredentials(

account="ACCOUNT-PLACEHOLDER", # resembles nh12345.us-east-2.snowflake

user="USER-PLACEHOLDER",

password="PASSWORD-PLACEHOLDER"

)

credentials.save("CREDENTIALS-BLOCK-NAME-PLACEHOLDER")

Then, to create a SnowflakeConnector block:

1. After logging in, click on any worksheet.
2. On the left side, select a database and schema.
3. On the top right, select a warehouse.
4. Create a short script, replacing the placeholders below.

from prefect\_snowflake import SnowflakeCredentials, SnowflakeConnector

credentials = SnowflakeCredentials.load("CREDENTIALS-BLOCK-NAME-PLACEHOLDER")

connector = SnowflakeConnector(

credentials=credentials,

database="DATABASE-PLACEHOLDER",

schema="SCHEMA-PLACEHOLDER",

warehouse="COMPUTE\_WH",

)

connector.save("CONNECTOR-BLOCK-NAME-PLACEHOLDER")

You can now easily load the saved block, which holds your credentials and connection info:

from prefect\_snowflake import SnowflakeCredentials, SnowflakeConnector

SnowflakeCredentials.load("CREDENTIALS-BLOCK-NAME-PLACEHOLDER")

SnowflakeConnector.load("CONNECTOR-BLOCK-NAME-PLACEHOLDER")