Italy esero

UNA COLLABORAZIONE TRA

esa asi Agenzia Spaziale Italiana

A SCUOLA CON LO SPAZIO

→ Guida per la creazione del programma per la Mission Space Lab di Astro Pi

Tutorial per team del progetto Astro Pi

Traduzione e adattamento Esero Italia

MUSEO NAZIONALE SCIENZA E TECNOLOGIA LEONARDO DA VINCI
CON
FONDAZIONE IDIS CITTÀ DELLA SCIENZA
INFINI-TO Planetario di Torino Museo dell'Astronomia e dello Spazio
MUSE
Museo del Bali IN PRATICA, LA SCIENZA.
PSIQUADRO la scienza interna

Introduzione

Questa guida è pensata per aiutarti, insieme al tuo team, a creare il programma per la Mission Space Lab 2024/25. In questa missione, il vostro compito sarà creare un programma che raccolga dati utilizzando i sensori e la fotocamera a luce visibile di un computer Astro Pi e che utilizzi questi dati per calcolare la velocità alla quale viaggia la Stazione Spaziale Internazionale (ISS). Metteremo a disposizione numerosi materiali di supporto per aiutarvi a scrivere e sviluppare il programma, incluso un progetto di esempio che utilizza fotografie storiche. Vi guideremo anche nell'adattare e testare il programma in modo che possa funzionare per 10 minuti a bordo della ISS e calcolare in tempo reale la velocità della stazione spaziale.

Questa non è una guida passo-passo completa su come creare un programma per risolvere il problema proposto in questa missione. Spetterà a te e al tuo team sviluppare idee, trovare soluzioni e capire come implementarle.

Molte risposte alle domande che potrebbero sorgere si trovano anche cercando online e vi incoraggiamo a fare ricerche e a provare soluzioni diverse se vi trovate in difficoltà. Organizzeremo anche un paio di webinar online programmati, durante i quali potrete fare domande direttamente al team dell'Astro Pi Mission Control. In alternativa, potete contattarci via email all'indirizzo **contact@astro-pi.org**.

Sono disponibili numerose risorse per aiutarvi a ottenere successo in ogni fase del vostro percorso con Astro Pi. Non conosci Mission Space Lab? Non preoccuparti! Visita il sito web di Astro Pi per ulteriori informazioni.

Cosa dovrete creare

Il vostro compito è progettare un programma che funzioni per 10 minuti a bordo della ISS e, durante questo tempo, raccolga dati per stimare la velocità della Stazione Spaziale Internazionale. Al termine dei 10 minuti, il programma dovrà aver scritto un file contenente la vostra stima della velocità della ISS espressa in chilometri al secondo.

Potete utilizzare il nostro **Astro Pi Replay Tool** per simulare l'esecuzione del vostro codice sugli Astro Pi presenti sulla ISS, in modo da testare che il programma funzioni correttamente in tempo reale.

Vi forniremo inoltre informazioni su come migliorare il programma per assicurarvi che funzioni senza problemi sulla ISS, rispettando anche le regole di sicurezza.

Di cosa avrete bisogno

Per completare questo progetto, avrete bisogno di:

- **Un computer con Python 3.11 o successivo**. Potete utilizzare un computer con Microsoft Windows, macOS o Linux. Le istruzioni per installare Python sono disponibili qui (in inglese).

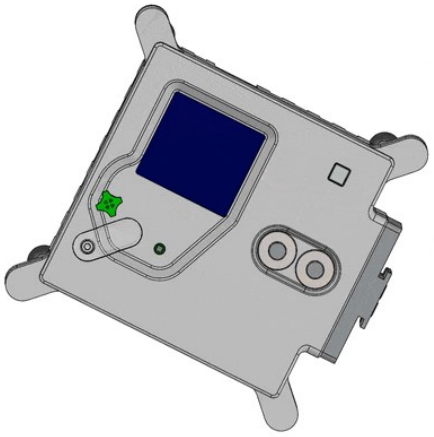
Una descrizione completa dei requisiti Python per la Mission Space Lab verrà fornita più avanti in questa guida.

- **Una connessione a internet**. Vi servirà per testare e inviare il programma.

I computer Astro Pi

I computer Astro Pi a bordo della ISS sono due Raspberry Pi 4 modificati con 8 GB di RAM, dotati di una scheda aggiuntiva Sense HAT e una fotocamera, il tutto racchiuso in un apposito case di volo in alluminio personalizzato. Il Sense HAT (V2) include sensori come temperatura, umidità, giroscopio, magnetometro, accelerometro e sensori di luce/colore, che vi permettono di misurare parametri come il campo magnetico locale e l'accelerazione. I computer sono anche equipaggiati con una potente Raspberry Pi High Quality Camera con lente da 5 mm, che consente di scattare foto straordinarie della Terra.

Per maggiori dettagli sui computer e sui sensori, potete trovare ulteriori informazioni qui (in inglese).



Conoscendo ciò che i sensori disponibili sugli Astro Pi possono fare, pensa in modo creativo su come utilizzarli per determinare la velocità della ISS. Non preoccuparti di fare tutto perfetto fin dall'inizio. Cerca di pensare a modi diversi, anche se sembrano insoliti. Da solo o in squadra, quante soluzioni riesci a immaginare per calcolare la velocità utilizzando questi strumenti?

Nella sezione successiva, imparerai a conoscere le diverse librerie Python disponibili che possono aiutarti nel tuo progetto, oltre a quelle che non puoi utilizzare per motivi di sicurezza. Non è necessario utilizzare tutte le librerie Python presentate in questa sezione, ma solo quelle che decidi di impiegare nel tuo programma per farlo funzionare come desideri.

L'ambiente Python Astro Pi

I computer Astro Pi sulla ISS hanno installato Python versione 3.11, quindi dovrai utilizzare questa versione o una versione superiore. Se stai utilizzando una versione più recente, fai attenzione, poiché potrebbero esserci alcune funzioni nuove che funzionano sul tuo computer ma non sugli Astro Pi.

Esistono alcune restrizioni sui moduli (parti) della libreria standard che puoi utilizzare. I seguenti moduli non sono consentiti, e se li utilizzi, il tuo programma non sarà accettato:

Librerie non consentite

Oltre all'ambiente Python standard, sugli Astro Pi sono installate librerie aggiuntive per aiutarti a completare la missione. Ogni libreria è spiegata brevemente qui sotto con esempi. Sono anche forniti link per ulteriori dettagli, se ne avessi bisogno. Ricorda di aggiungere questa pagina ai preferiti per poterla consultare in seguito!

Skyfield

Skyfield è un pacchetto astronomico che calcola le posizioni di stelle, pianeti e satelliti in orbita attorno alla Terra.
Ad esempio, è possibile utilizzare Skyfield per calcolare la posizione attuale di Marte:

```
from skyfield.api import Loader
from pathlib import Path

bsp_file = Path.home() / "de421.bsp"
load = Loader(bsp_file.parent)
planets = load(bsp_file.name)
mars = planets['Mars Barycenter']
ts = load.timescale()
barycentric = mars.at(ts.now())
print(barycentric)
```

Questo frammento funziona ma il file delle effemeridi (`de421.bsp`) è troppo grande per essere inviato nel payload finale! Per ovviare a questo problema, importa le effemeridi dalla libreria `astro_pi_orbit`, che si occuperà di importare il file per te (Documentazione: rhodesmill.org/skyfield)

```
from skyfield.api import load
from astro_pi_orbit import de421

planets = de421
mars = planets['Mars Barycenter']
ts = load.timescale()
barycentric = mars.at(ts.now())
print(barycentric)
```

astro_pi_orbit

La libreria astro_pi_orbit fornisce funzionalità per assistere i partecipanti all'Astro Pi Mission Space Lab nel lavoro con i dati orbitali. Può essere utilizzato per:

1) Trovare la posizione attuale della ISS
2) Accedere ai file delle effemeridi de421 o de440s (i file sono troppo grandi per essere forniti da solo)
3) Accedere alla traiettoria della ISS

Documentazione: [https://astro-pi.github.io/astro-pi-orbit/]

Picamzero

La libreria Python per il controllo del modulo fotocamera Raspberry Pi sugli Astro Pi è picamera-zero. Per iniziare, dai un'occhiata a questa guida al progetto (in inglese) per una pratica guida su come usarlo.

```
from picamzero import Camera
from time import sleep

camera = Camera()

# Take a picture every minute for 3 hours
for i in range(3*60):
    camera.take_photo(f'image_{i:03d}.jpg')
    sleep(60)
```

Documentazione: https://raspberrypifoundation.github.io/picamera-zero

GPIO Zero

GPIO Zero è una libreria GPIO (General-Purpose Input/Output) semplice, ma potente. La maggior parte delle sue funzionalità è limitata a bordo della ISS: ad esempio, l'unico pin a cui è consentito accedere è il pin GPIO 12, dove è collegato il sensore di movimento. Tuttavia, alcune delle sue altre funzionalità possono essere utili nel tuo esperimento, come il dispositivo interno CPUTemperature.

Confronta la temperatura della CPU del Raspberry Pi con la lettura della temperatura del Sense HAT:

```
from sense_hat import SenseHat
from gpiozero import CPUTemperature

sense = SenseHat()
cpu = CPUTemperature()

while True:
    print(f'CPU: {cpu.temperature}')
    print(f'Sense HAT: {sense.temperature}')
```

Documentazione: gpiozero.readthedocs.io

NumPy

numpy è una libreria progettata per aiutare a lavorare con i numeri in Python. Viene utilizzato più spesso per fare matematica con grandi griglie di numeri, che numpy chiama "array".

È possibile utilizzare numpy per elaborare i singoli pixel in ogni immagine della fotocamera scattata utilizzando la funzione capture_array di picamzero. Ad esempio, è possibile filtrare l'immagine in modo da mostrare solo il canale rosso:

```
from picamzero import Camera
import numpy as np
import PIL

camera = Camera()
image_as_array = camera.capture_array()
red_channel = image_as_array[:, :, 0]
Image.fromarray(red_channel).save("red-channel.jpg")
```

Documentazione: https://numpy.org/doc/stable/user/index.html

SciPy

SciPy è una libreria Python gratuita e open source utilizzata per il calcolo scientifico e il calcolo tecnico. SciPy contiene moduli per l'ottimizzazione, l'algebra lineare, l'integrazione, l'interpolazione, funzioni speciali, FFT (Fast Fourrier Transform), l'elaborazione di segnali e immagini, risolutori ODE (Ordinary Differential Equations) e altre attività comuni nella scienza e nell'ingegneria. Potrebbe essere necessario utilizzare questa libreria per risolvere una particolare equazione.

Documentazione: docs.scipy.org/doc

pandas

pandas è una libreria open source che fornisce strutture dati e strumenti di analisi dei dati ad alte prestazioni e facili da usare.

```
import pandas as pd

df = pd.read_csv("my_test_data.csv")
df.describe()
```

Documentazione: pandas.pydata.org

logzero

logzero è una libreria utilizzata per semplificare la registrazione. I log sono registrazioni di ciò che è accaduto durante l'esecuzione di un programma e possono essere molto utili per il debug.

I log sono classificati in diversi livelli in base alla gravità. Utilizzando i vari livelli in modo appropriato, sarai in grado di regolare la quantità di informazioni che ottieni sul tuo programma in base alle tue esigenze di debug.

```
from logzero import logger

logger.debug("hello")
logger.info("info")
logger.warning("warning")
logger.error("error")
```

Documentazione: logzero.readthedocs.io

Matplotlib

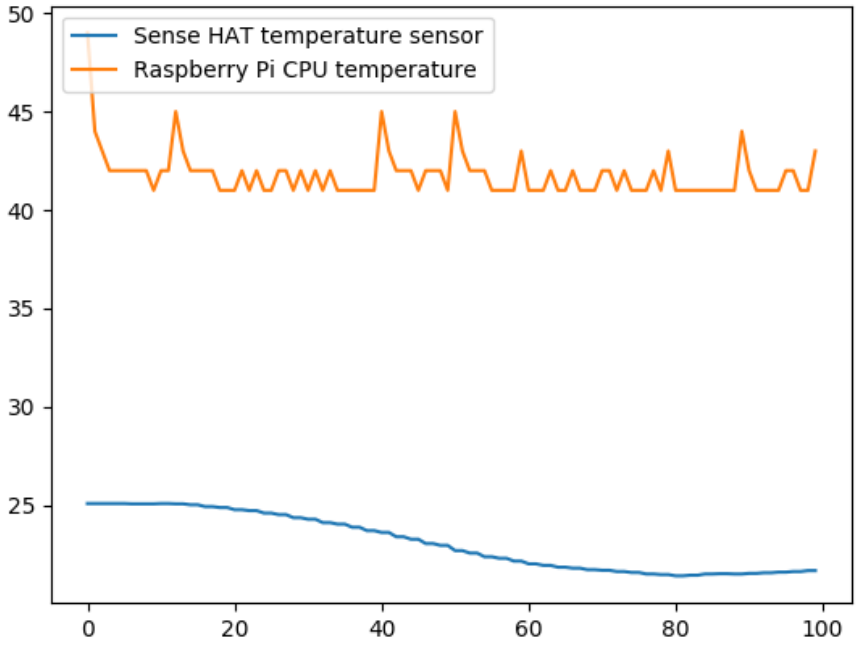
matplotlib è una libreria di plottaggio 2D che produce figure di qualità editoriale in una varietà di formati cartacei e ambienti interattivi. È possibile utilizzarlo per analizzare i risultati delle esecuzioni dei test.

```python
from sense_hat import SenseHat
from gpiozero import CPUTemperature
import matplotlib.pyplot as plt
from time import sleep

sense = SenseHat()
cpu = CPUTemperature()

st, ct = [], []
for i in range(100):
    st.append(sense.temperature)
    ct.append(cpu.temperature)
    sleep(1)

plt.plot(st)
plt.plot(ct)
plt.legend(['Sense HAT temperature sensor', 'Raspberry Pi CPU temperature'], loc='upper left')
plt.show()
```



Documentazione: matplotlib.org

Pillow

Pillow è una libreria di elaborazione delle immagini. Fornisce un ampio supporto per i formati di file, un'efficiente rappresentazione interna e capacità di elaborazione delle immagini abbastanza potenti.

La libreria di immagini di base è progettata per un accesso rapido ai dati memorizzati in alcuni formati di pixel di base. Dovrebbe fornire una solida base per uno strumento generale di elaborazione delle immagini.

Documentazione: pillow.readthedocs.io

OpenCV

opencv è una libreria open source di visione artificiale. Ad esempio, è possibile utilizzare OpenCV per il rilevamento dei bordi.

Documentazione: docs.opencv.org

exif

exif consente di leggere e modificare i metadati Exif delle immagini utilizzando Python. Potresti volerlo utilizzare per incorporare i dati GPS in qualsiasi immagine scattata o per analizzare le foto scattate a bordo della ISS.

Documentazione: pypi.org/project/exif

scikit-learn

scikit-learn è un insieme di strumenti semplici ed efficienti per il data mining e l'analisi dei dati, accessibili a tutti e riutilizzabili in vari contesti. È progettato per funzionare con numpy, scipy e matplotlib.

Documentazione: scikit-learn.org

scikit-image

scikit-image è una libreria di elaborazione delle immagini open source. Include algoritmi per la segmentazione, le trasformazioni geometriche, la manipolazione dello spazio colore, l'analisi, il filtraggio, la morfologia, il rilevamento delle caratteristiche e altro ancora.

Documentazione: scikit-image.org

reverse-geocoder

Il reverse-geocoder inverso prende una coordinata di latitudine/longitudine e restituisce la città più vicina.

Se utilizzato con skyfield, il reverse-geocoder inverso può determinare dove si trova attualmente la ISS:

```
import reverse_geocoder
from skyfield.api import Loader
from pathlib import Path

tle_file = Path.home() / "iss.tle"
load = Loader(tle_file.parent)
if not tle_file.exists():
    load.download("https://celestrak.com/NORAD/elements/stations.txt", filename=tle_file.name)
satellites = load.tle_file(tle_file.name)
iss = satellites[0]
ts = load.timescale()

coordinates = iss.at(ts.now()).subpoint()
coordinate_pair = (
    coordinates.latitude.degrees,
    coordinates.longitude.degrees)

location = reverse_geocoder.search(coordinate_pair)
print(location)
```

Questo output mostra che la ISS si trova attualmente sopra Hamilton, New York:

```
[OrderedDict([
    ('lat', '42.82701'),
    ('lon', '-75.54462'),
    ('name', 'Hamilton'),
    ('admin1', 'New York'),
    ('admin2', 'Madison County'),
    ('cc', 'US')
])]
```

Nota: il `reverse-geocoder` della libreria non può essere eseguito utilizzando lo strumento di riproduzione online in quanto utilizza il multiprocessing, che è incompatibile con l'ambiente dello strumento. Se desideri utilizzare questa libreria, dovrai testare le sezioni pertinenti del tuo codice localmente nel tuo editor di codice o utilizzando la versione del plug-in Thonny dello strumento Replay.

Documentazione: github.com/thampiman/reverse-geocoder

sense_hat

La libreria `sense_hat` è la libreria principale utilizzata per raccogliere i dati utilizzando l'HAT Astro Pi Sense. Dai un'occhiata a questa guida al progetto (in inglese) per iniziare.

È possibile stampare l'umidità utilizzando il codice seguente:

```
from sense_hat import SenseHat
sense = SenseHat()
print(str(sense.get_humidity()))
```

Documentazione

1. https://sense-hat.readthedocs.io/en/latest/
2. Documentazione aggiuntiva per il sensore di luce e colore

Poiché ci sono molte restrizioni di sicurezza quando si esegue un programma a bordo della ISS, queste sono le uniche librerie di terze parti che ti sarà permesso di utilizzare se il tuo programma viene eseguito sugli AstroPi.

Configurazione dell'ambiente di programmazione

Si consiglia di utilizzare Thonny per creare il programma.

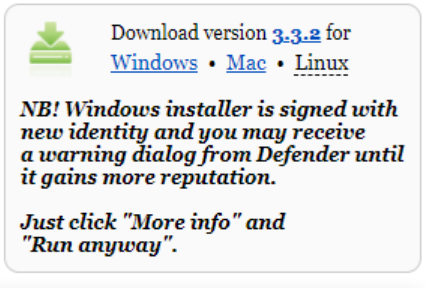
Installare Thonny su un Raspberry Pi

1. Thonny è già installato su Raspberry Pi OS, ma potrebbe essere necessario aggiornarlo all'ultima versione.
2. Apri una finestra di terminale, facendo clic sull'icona nell'angolo in alto a sinistra dello schermo o premendo contemporaneamente i tasti Ctrl+Alt+T
3. Nella finestra, digita quanto segue per aggiornare il tuo sistema operativo e Thonny

```
sudo apt update && sudo apt upgrade -y
```

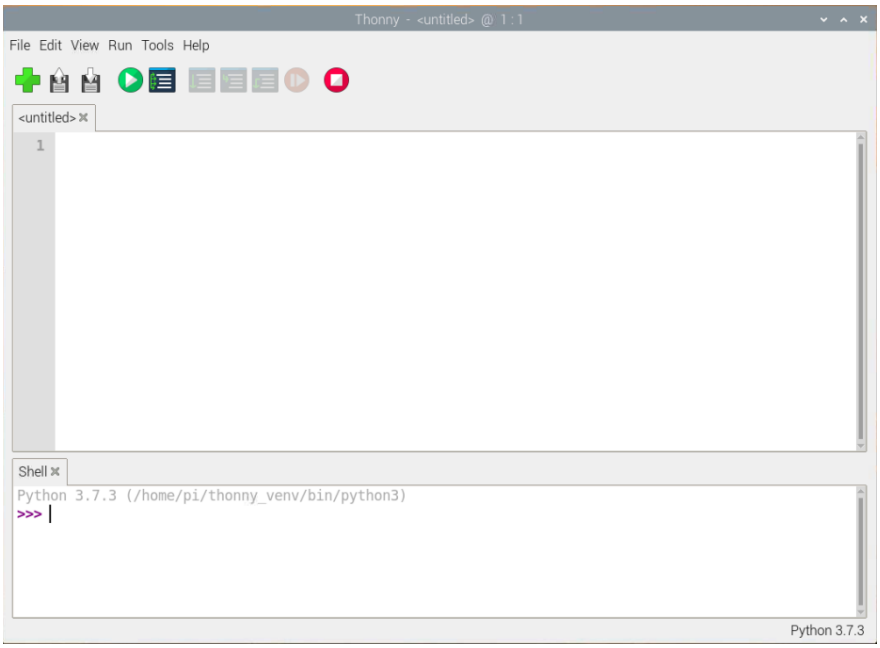
Installare Thonny su altri sistemi operativi

1. Su Windows, macOS e Linux, puoi installare l'ultimo IDE Thonny o aggiornare una versione esistente
2. In un browser Web, vai a thonny.org
3. Nell'angolo in alto a destra della finestra del browser, vedrai i link per il download per Windows e macOS e le istruzioni per Linux
4. Scarica i file pertinenti ed eseguili per installare Thonny



Apertura Thonny

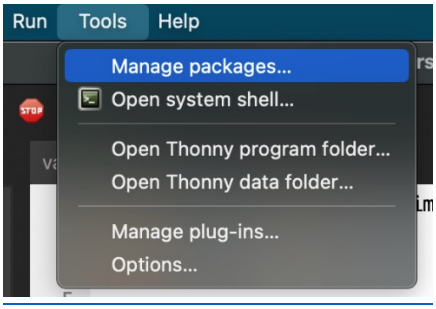
Apri Thonny. Dovrebbe assomigliare a questo:



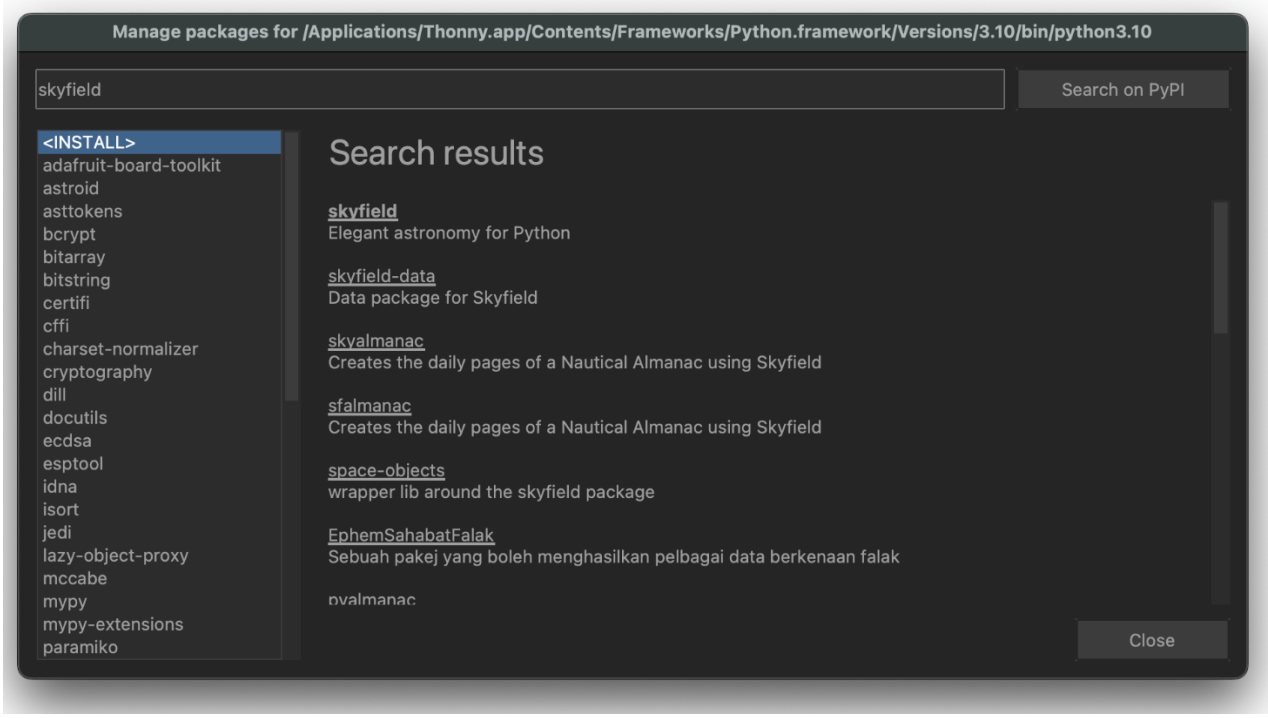
È possibile utilizzare Thonny per scrivere codice Python standard. Digita quanto segue nella finestra principale, quindi fai clic sul pulsante **Esegui** (ti verrà chiesto di salvare il file).

```
print('Hello World!')
```

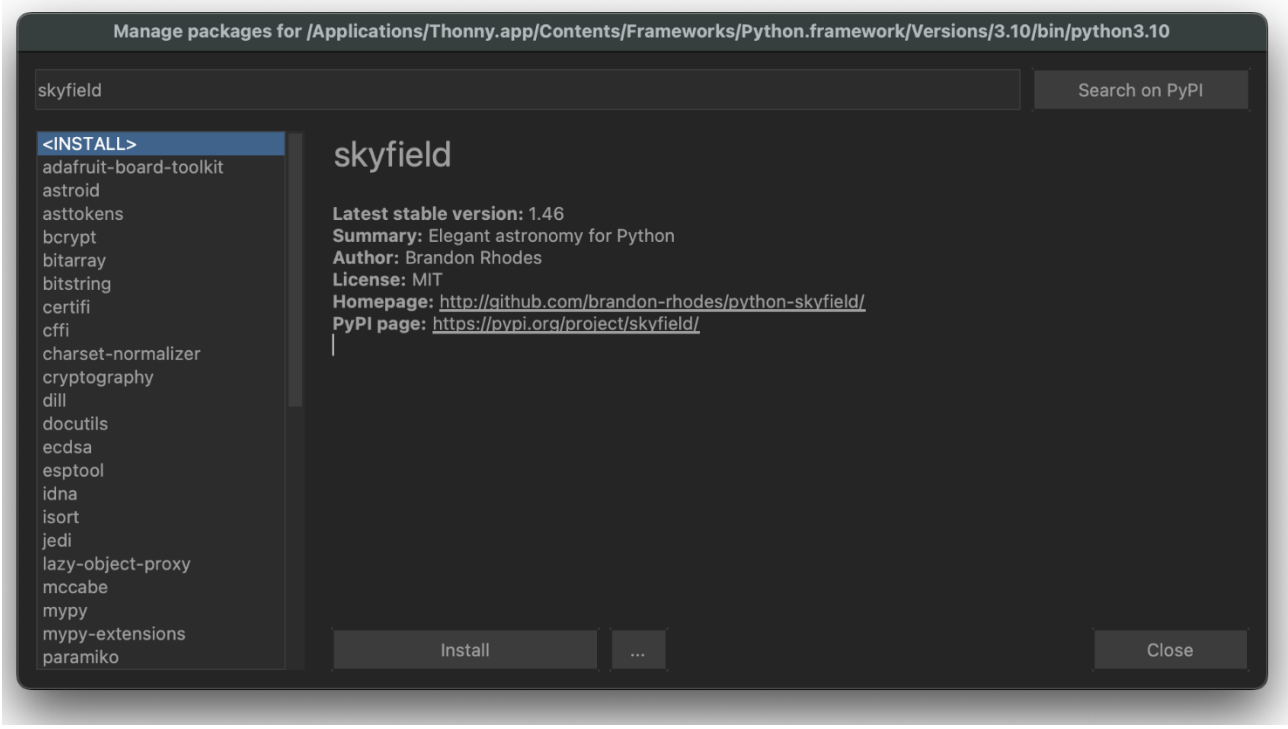
Per installare una qualsiasi delle librerie Python, apri Thonny e fai clic su **Strumenti > Gestisci pacchetti...**.



Cerca la libreria che desideri digitandone il nome nella barra di ricerca.



Seleziona il file corretto dai risultati della ricerca, quindi premi **Installa**.



Se si utilizza un IDE diverso per scrivere il codice, sarà necessario seguire le istruzioni locali per scaricare le librerie desiderate da PyPi.

Uno sguardo al futuro

Ora che hai configurato il tuo ambiente di programmazione, è il momento di pensare a come il tuo team affronterà questa missione. Discuti su come sceglierai il tuo metodo, dividi i compiti e pianifichi il tuo programma. Parla con il mentore del tuo team delle tue idee, dei tuoi progressi e di eventuali ostacoli lungo il percorso. Avranno molte idee per aiutarti a pianificare.

Scrivere il tuo programma e le tue risorse per aiutarti

Questa sezione ti aiuterà a iniziare a scrivere il tuo programma e fornirà collegamenti ad altre guide al progetto che ti aiuteranno a sviluppare alcune delle competenze di codifica di cui potresti aver bisogno. Puoi scegliere quali guide di progetto vuoi guardare a seconda di quale dei sensori e/o della fotocamera utilizzerai nel tuo programma. A questo punto, dovresti aver già trascorso un po' di tempo con il tuo team e il tutor del tuo team per pianificare il tuo programma e aver deciso quali dati raccoglierai per fare i tuoi calcoli.
Ti consigliamo di iniziare a scrivere il tuo programma a piccoli passi e di non cercare di fare tutto in una volta.

Per mantenere tutto organizzato, crea una cartella in cui archiviare tutti i file del tuo progetto. Per il nome della cartella, potresti voler utilizzare il nome del tuo team.

Il file main.py

Ogni invio deve includere un file denominato main.py. Questo è il file da cui verrà eseguito il programma e che verrà testato dal Controllo Missione Astro Pi. Quando si esegue il programma finito, dovrebbe fare tutto il necessario per stimare la velocità della ISS. Inizia creando un file per il tuo programma principale e aggiungi il codice che riesci a far funzionare man mano che procedi.

Crea un nuovo file in Thonny e salva come main.py nella cartella del progetto.

Metti alla prova il tuo programma con lo strumento di riproduzione Astro Pi

L'Astro Pi Replay Tool agisce come una sorta di simulatore che puoi utilizzare sulla Terra e che farà sì che il tuo programma si comporti come se fosse in esecuzione su un Astro Pi a bordo della ISS. Ti consente di testare il tuo codice prima che vada nello spazio senza bisogno di avere un Raspberry Pi, una fotocamera o un Sense HAT. La simulazione non è perfetta, tuttavia, e produrrà solo foto e dati dei sensori all'interno del proprio set di dati, ma dovrebbe comunque consentire di verificare che il programma funzioni durante l'esecuzione a bordo della ISS.
Esiste una versione online e una versione offline, disponibili come plug-in Thonny, per testare il programma. Si consiglia di utilizzare la versione online dello strumento.

Accesso allo strumento di riproduzione di Astro Pi online

Il modo più semplice per verificare se il tuo programma funzionerà sulla ISS è caricare il tuo file main.py sullo strumento online Astro Pi Replay Tool.

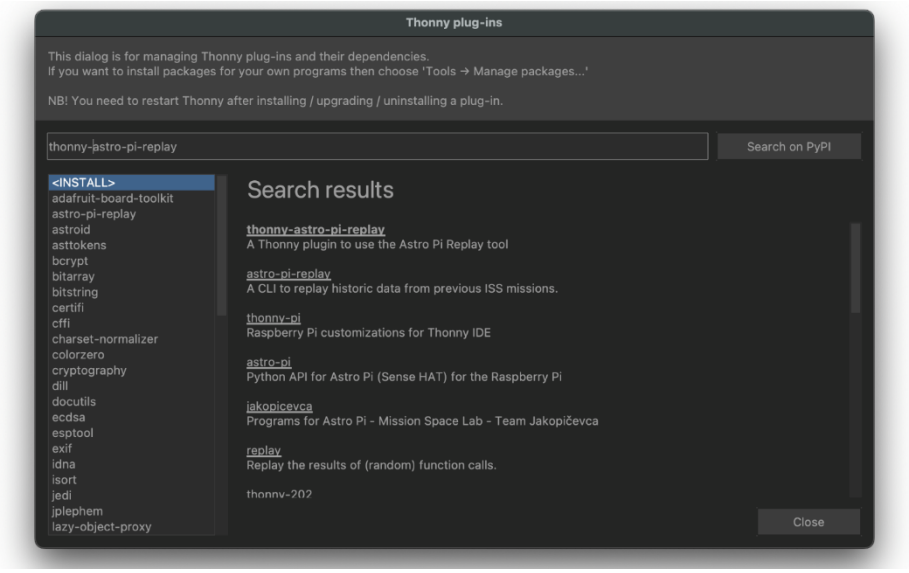
Per caricare il tuo programma è sufficiente aprire il link e trascinare e rilasciare, oppure selezionare, il file main.py e fare clic su **Esegui**. Lo strumento Replay eseguirà il programma per intero e mostrerà le immagini e i dati acquisiti, insieme a tutti i file emessi dal programma.

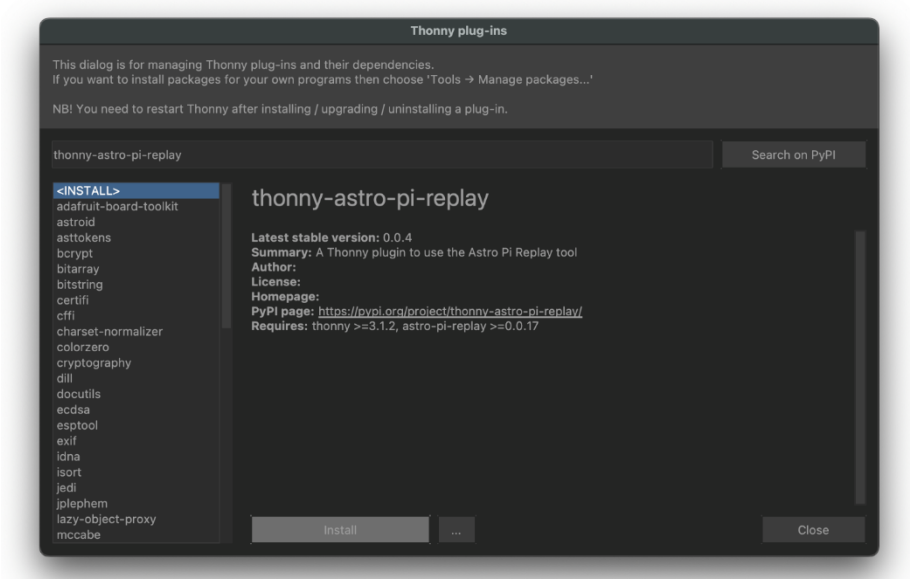
Assicurati che il tuo programma abbia un output con la tua stima finale della velocità in km (chilometri al secondo).

Installazione dello strumento di riproduzione Astro Pi su Raspberry Pi Bookworm

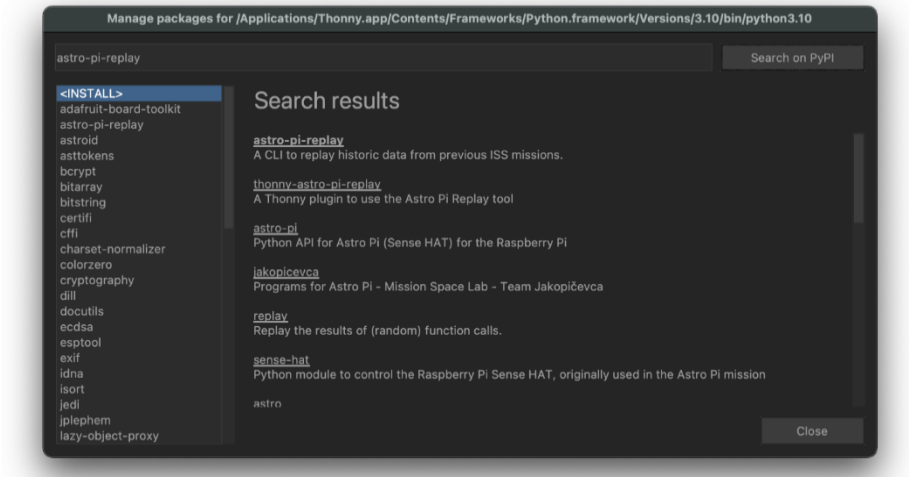
Se utilizzi Raspberry Pi OS Bookworm, segui le istruzioni su come configurare Thonny per utilizzare un ambiente virtuale sul sito raspberry pi prima di procedere con le istruzioni seguenti.

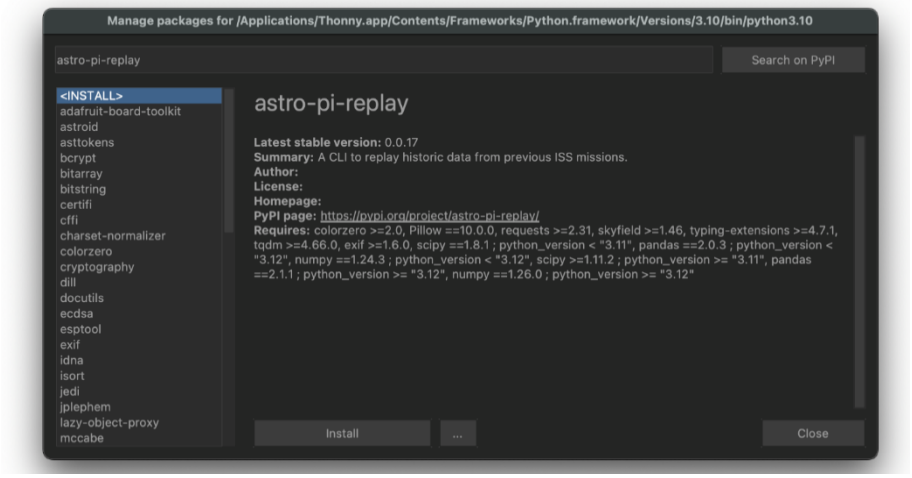
Per installare lo strumento Astro Pi Replay, apri Thonny, quindi fai clic su **Strumenti > Gestisci plug-in...** e cerca thonny-astro-pi-replay. Selezionare il plug-in corretto, quindi premere **Installa**.





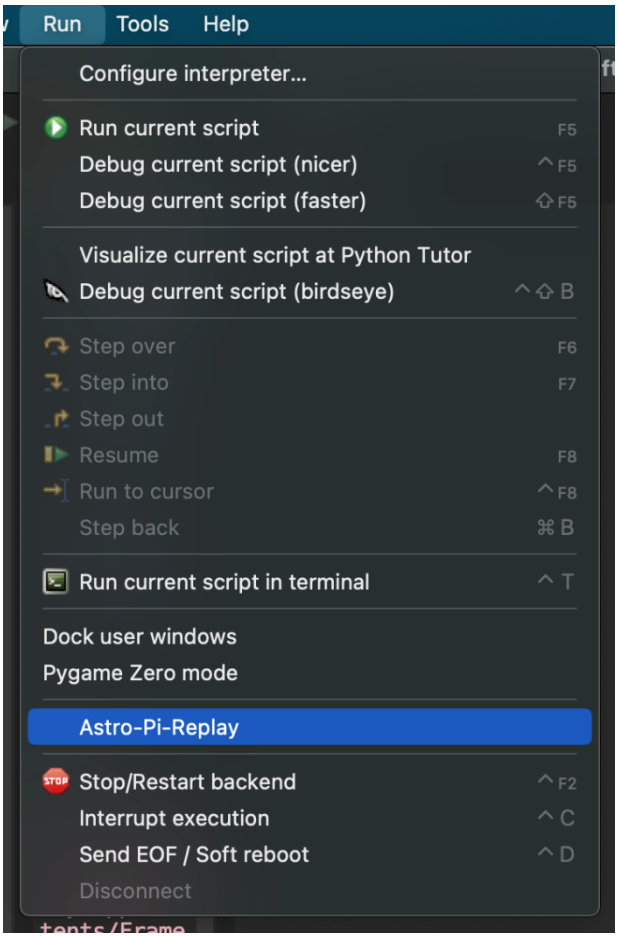
Quindi, fai clic su **Strumenti > Gestisci pacchetti**... e cerca astro-pi-replay. Seleziona il pacchetto corretto, quindi premi **Installa**.





Se hai già partecipato a Mission Space Lab e hai già scaricato lo strumento Astro Pi Replay, dovresti reinstallare la libreria astro-pi-replay per assicurarti di avere l'ultima versione. Per fare ciò, rimuovi la

directory `~/.astro_pi_replay` nella tua cartella home (ad esempio usando il comando rm -rf `~/.astro_pi_replay` in una finestra di Terminale) e poi segui le istruzioni sopra, come se non avessi mai installato `astro-pi-replay` prima.



Sarà necessario chiudere e riavviare Thonny per completare l'installazione.

Lo strumento Astro Pi Replay funziona riproducendo una serie di vecchie foto scattate sulla ISS. Quando il codice passa a scattare una foto, invece di accedere all'hardware della fotocamera, la libreria seleziona un'immagine da riprodurre e si comporta come se fosse stata appena acquisita "dal vivo".

Come utilizzare il plug-in Astro Pi Replay

Per eseguire il codice utilizzando il plug-in Astro Pi Replay, non premere il pulsante verde Esegui. Invece, apri il menu Esegui, quindi fai clic su Astro-Pi-Replay. Questo eseguirà il tuo codice come se fosse in esecuzione sull'hardware Astro Pi.

Nota: Sebbene siano disponibili tutte le funzioni della libreria `picamera-zero`, molte delle impostazioni e dei parametri `picamera-zero` che normalmente comporterebbero l'acquisizione di un'immagine diversa vengono ignorati in silenzio quando il codice viene eseguito utilizzando Astro Pi Replay. Inoltre, la maggior parte degli attributi dell'oggetto `Camera` viene ignorata. Ad esempio, l'impostazione dell'attributo resolution su un valore diverso da (`4056,3040`) non ha alcun effetto quando viene simulato su Astro Pi Replay, ma cambia la risoluzione quando viene eseguito su un Astro Pi nello spazio.

Calcolo con dati storici

Potresti voler iniziare imparando a scrivere un programma che stima la velocità della ISS utilizzando le foto con la nostra guida al progetto Calcola la velocità della ISS (in inglese) utilizzando le foto. Una volta scritto un programma, puoi provarlo utilizzando diverse immagini o set di dati per migliorare l'accuratezza della tua stima. Di seguito sono riportati alcuni esempi di immagini e dati che è possibile utilizzare:

1. Foto del laboratorio spaziale della missione Astro Pi 2022/23
2. Dati Astro Pi Mission Space Lab 2022/23

Non dimenticate che quest'anno potrete utilizzare solo la fotocamera a luce visuale sulla ISS.

Simula l'esecuzione del tuo programma in tempo reale

Potresti preferire iniziare utilizzando le librerie sense_hat e picamera-zero e simulando l'esecuzione del tuo programma in tempo reale. Per simulare la lettura dei dati dal Sense HAT e l'acquisizione di foto dalla fotocamera, utilizzerai lo strumento Astro Pi Replay online o con Thonny.

Effettuare misurazioni con il Sense HAT

Per calcolare la velocità della ISS, è possibile raccogliere dati dai sensori sul Sense HAT. Dai un'occhiata alla nostra guida introduttiva al progetto Sense HAT per scoprire come farlo.

Scattare foto con la fotocamera

Potresti anche voler utilizzare la fotocamera per scattare foto della Terra da utilizzare nel tuo programma. Puoi utilizzare la nostra guida introduttiva al progetto Camera Module (in inglese) per scoprire come farlo. Tuttavia, se non disponi di un Raspberry Pi e di una fotocamera di alta qualità su cui testare il tuo codice, puoi comunque eseguire lo stesso codice utilizzando lo strumento di riproduzione di Astro Pi.

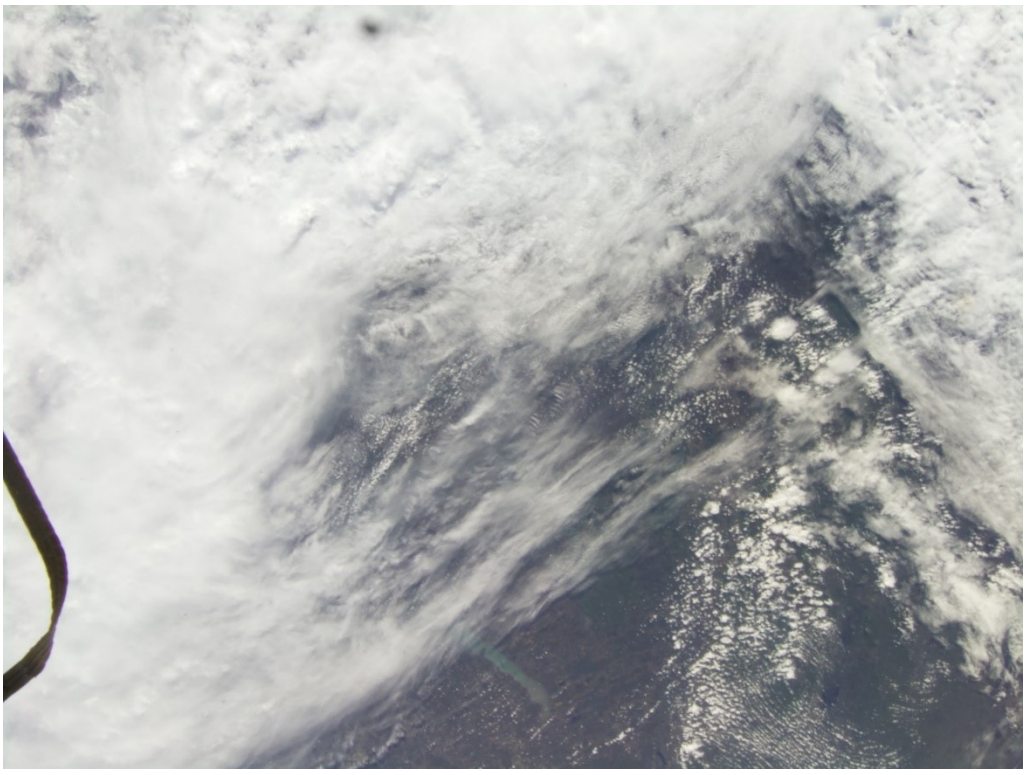
Ecco un esempio di un semplice programma per testare il plug-in Astro Pi Replay, se si utilizza la versione offline in Thonny:

```
# Import the Camera class from the picamera-zero module
from picamzero import Camera

# Create an instance of the Camera class
cam = Camera()

# Capture an image
cam.take_photo("image1.jpg")
```

In questo modo si simula lo scatto di una foto sulla ISS e la si salva in un file chiamato image1.jpg. Se apri questo file, dovresti vedere la foto esatta qui sotto.



La libreria picamera-zero supporta una varietà di funzioni e impostazioni della fotocamera. Puoi vedere alcuni esempi andando alla pagina "Ricette" sul sito web di picamera-zero, ma tieni presente che se il tuo codice viene eseguito sulla ISS, scatterà foto di una varietà di condizioni meteorologiche con una gamma di nuvole, paesaggi e illuminazione. Tuttavia, il tuo programma è sempre garantito per essere eseguito alla luce del giorno.

Mentre tutte le funzionalità della libreria picamera-zero saranno disponibili sull'Astro Pi nello spazio, non tutte possono essere simulate dallo strumento di riproduzione Astro Pi.

Acquisizione di sequenze

Utilizzando picamera-zero è molto semplice scattare una sequenza di foto chiamando la funzione capture_sequence. L'esempio seguente scatta tre foto in successione, con un intervallo di 3 secondi tra una foto e l'altra.

Crea un nuovo file chiamato camera-sequence.py e in esso digita le seguenti righe:

```
# Import the Camera class from the picamzero (picamera-zero) module
from picamzero import Camera

# Create an instance of the Camera class
cam = Camera()

cam.capture_sequence("sequence", num_images=3, interval=3)
```

Esegui questo codice utilizzando Astro Pi Replay online) o con il plug-in Thonny facendo clic su **Esegui > Astro-Pi-Replay.**

Piani di numerazione per immagini e file

Quando si ha a che fare con molti file dello stesso tipo, è una buona idea seguire una convenzione di denominazione. Nell'esempio precedente, utilizziamo un numero di sequenza ovvio - `image1.png`, `image2.png`, ecc. - per mantenere organizzati i nostri file.

Se hai bisogno di ulteriore aiuto con l'uso della fotocamera, dai un'occhiata al passaggio "Scatta foto con il codice Python" nella nostra guida al progetto "Guida introduttiva al modulo fotocamera".

Aggiorna il tuo file `main.py` per acquisire immagini o dati Sense HAT in tempo reale.

Trovare la posizione della ISS

Potrai scaricare fino a 42 foto che scatterai sulla ISS. Può essere utile sapere dove è stata scattata esattamente un'immagine e lo puoi fare facilmente con le librerie `astro_pi_orbit` ed `exif` disponibili sugli Astro Pi.

Di seguito è riportato un esempio di un programma che, quando viene eseguito utilizzando lo strumento di riproduzione di Astro Pi, crea una nuova immagine chiamata `gps_image1.jpg`. La libreria `picamzero` avrà impostato i metadati Exif per l'immagine in modo che includano la latitudine e la longitudine correnti della ISS.

```
from astro_pi_orbit import ISS
from picamzero import Camera

iss = ISS()

def get_gps_coordinates(iss):
    """
    Returns a tuple of latitude and longitude coordinates expressed
    in signed degrees minutes seconds.
    """
    point = iss.coordinates()
    return (point.latitude.signed_dms(), point.longitude.signed_dms())

cam = Camera()
cam.take_photo("gps_image1.jpg", gps_coordinates=get_gps_coordinates(iss))
```

Dovrai utilizzare lo strumento Astro Pi Replay per eseguire questo frammento.

Si noti che la latitudine e la longitudine sono oggetti Angle mentre l'elevazione è una Distance. La documentazione di Skyfield descrive come passare da una rappresentazione angolare all'altra e come esprimere la distanza in unità diverse (in inglese).

Apprendimento automatico con l'acceleratore Coral

Se hai accesso a un acceleratore di machine learning Coral, consulta la nostra guida al progetto Classificazione delle immagini con Google Coral (in inglese). Potrai consultare un processo passo passo per capire un modello di apprendimento automatico per classificare le immagini e sperimenterai l'utilizzo della libreria TensorFlow Lite. È quindi possibile utilizzare un approccio simile per classificare le immagini riprodotte quando si esegue il programma utilizzando lo strumento di riproduzione Astro Pi o sulla ISS.

Una volta completato questo progetto, potresti voler esaminare la pagina degli esempi di Coral (in inglese) e questa pagina GitHub per trovare ispirazione su come applicare le tecniche di apprendimento automatico al tuo esperimento.

Scrittura del file dei risultati

Affinché la tua presentazione superi i test da parte del Controllo Missione Astro Pi, il tuo programma deve scrivere un file chiamato result.txt che contiene la tua stima della velocità della ISS. Questo file deve essere in formato file di testo (.txt) e conterrà la stima fino a un massimo di cinque cifre significative. Si prega di non includere altri dati in questo file, comprese le unità di misura, ad esempio km/s.

```
7.1235
```

Esempio result.txt per una stima della velocità media.

Di seguito è riportato un esempio di un programma che scriverà un file .txt chiamato result.txt con un valore di velocità stimato in chilometri al secondo (km/s) a 5 cifre significative. Dovrai adattare questo codice per adattarlo al tuo programma particolare.

Aggiorna il file main.py in modo che scriva un file denominato result.txt quando viene eseguito.

Assicurati di controllare il regolamento del Mission Space Lab (in inglese) per le regole sui file e sui nomi dei file.

```
estimate_kmps = 7.1234567890  # Replace with your estimate

# Format the estimate_kmps to have a precision
# of 5 significant figures
estimate_kmps_formatted = "{:.4f}".format(estimate_kmps)

# Create a string to write to the file
output_string = estimate_kmps_formatted

# Write to the file
file_path = "result.txt"  # Replace with your desired file path
with open(file_path, 'w') as file:
    file.write(output_string)

print("Data written to", file_path)
```

Ottimizzazione del programma per la ISS

Per gli astronauti, lavorare nello spazio significa lavorare sotto alcuni vincoli molto rigidi, e lo stesso vale per te. Questa sezione illustra come garantire che il codice si comporti come previsto durante l'esecuzione sulla ISS e come gestire elementi come risorse ed errori.

Esecuzione di un esperimento per 10 minuti

Ogni programma eseguito sugli Astro Pi ha una fascia oraria di 10 minuti alla luce del giorno per stimare la velocità della ISS. Il tuo programma dovrà tenere traccia del tempo e spegnersi normalmente prima che siano trascorsi i 10 minuti per assicurarsi che nessun dato venga perso.

Un modo per interrompere un programma Python dopo un determinato periodo di tempo consiste nell'utilizzare la libreria Python datetime. Questa libreria semplifica il lavoro con i tempi e il loro confronto.

Registrando e memorizzando l'ora all'inizio dell'esperimento, possiamo quindi controllare ripetutamente per vedere se l'ora corrente è maggiore di quell'ora di inizio più un certo numero di minuti, secondi o ore. Nel programma seguente, questo viene utilizzato per stampare "Hello from the ISS" ogni secondo per 1 minuto:

```
from datetime import datetime, timedelta
from time import sleep

# Create a variable to store the start time
start_time = datetime.now()
# Create a variable to store the current time
# (these will be almost the same at the start)
now_time = datetime.now()
# Run a loop for 1 minute
while (now_time < start_time + timedelta(minutes=1)):
    print("Hello from the ISS")
    sleep(1)
    # Update the current time
    now_time = datetime.now()
# Out of the loop - stopping
```

Aggiorna il file `main.py` per utilizzare la libreria datetime per arrestare il programma prima del termine dell'intervallo di tempo di 10 minuti.

Nota: quando decidi il runtime per il tuo programma, assicurati di tenere conto del tempo necessario al ciclo per completare un ciclo. Ad esempio, se si desidera utilizzare l'intero intervallo di 10 minuti disponibile, ma il completamento di ogni ciclo del codice richiede 2 minuti, il `timedelta` deve essere 10-2 = `8` minuti, per garantire che il programma venga terminato prima che siano trascorsi 10 minuti.

Utilizzo di percorsi relativi

Il tuo programma verrà archiviato in una posizione diversa quando verrà distribuito sulla ISS, quindi è davvero importante evitare di utilizzare percorsi di file assoluti quando scrivi il tuo file `result.txt` (o qualsiasi altro file che potresti voler scrivere). Utilizza il codice seguente per capire in quale cartella è attualmente archiviato il file `main.py`, che si chiama `base_folder`:

```
from pathlib import Path
base_folder = Path(__file__).parent.resolve()
```

Quindi puoi salvare i tuoi dati in un file sotto questo `base_folder`:

```
data_file = base_folder / "data.csv"
for i in range(10):
    with open(data_file, "w", buffering=1) as f:
        f.write(f"Some data: {i}")
```

Assicurati di controllare il regolamento del Mission Space Lab per le regole sui file e sui nomi dei file.

Chiusura delle risorse

Al termine dell'esperimento, è consigliabile chiudere tutte le risorse aperte. Ad esempio, chiudi tutti i file aperti:

```
file = open(file)
file.close()
```

Esamina il file main.py e aggiornalo in modo che chiuda tutte le risorse in modo appropriato.

Prepararsi agli imprevisti

Un programma può fallire per molte ragioni, ma con un po' di lungimiranza e pianificazione, è possibile che il tuo programma affronti questi problemi invece di bloccarsi e perdere la possibilità di acquisire dati e immagini a bordo della ISS. In questa sezione, cercherai di trovare modi per migliorare il tuo programma in modo che abbia le migliori possibilità di funzionare come previsto se accade qualcosa di imprevisto.

Gestione delle eccezioni

Un'eccezione è quando accade qualcosa mentre un programma è in esecuzione che non sa come gestire. Ciò può causare l'arresto anomalo del programma, a meno che non disponga di una procedura da seguire nel caso in cui qualcosa vada storto.

Visita Ada Computer Science per saperne di più sulla gestione delle eccezioni (in inglese).

Buffering dei file

Quando si scrive su un file utilizzando la funzione open, Python normalmente non salva immediatamente il file su disco. Invece, mantiene il contenuto del file da salvare in un'area di archiviazione temporanea nella memoria del computer chiamata buffer. Python lo fa in modo da poter scegliere il momento migliore per scrivere sul disco, qualcosa che normalmente non ci importa.

```
with open("some_file.txt", "w", buffering=1) as f:
    f.write("example data")
```

Tuttavia, mentre i dati si trovano nel buffer e non sono ancora stati salvati sul disco, è possibile che vadano persi se si verifica un errore. Per evitare che ciò accada, possiamo dire a Python di salvare il buffer su disco alla fine di ogni riga di testo impostando l'argomento `buffering` su `1`:

Nota: Se stai scrivendo byte su un file (con argomento "`wb`"), allora dovresti dire a Python di non utilizzare affatto un buffer e di scrivere immediatamente i dati su disco. A tale scopo, è possibile impostare l'argomento `buffering` su `0`.

Esamina il tuo programma e valuta se è necessario impostare la modalità di buffering durante la scrittura su un file.

Se il tuo programma fallisce, è sempre utile avere una registrazione di ciò che è successo, in modo da poterlo correggere per la prossima volta. La libreria Python `logzero` (documentazione qui in inglese) rende facile prendere appunti su ciò che sta accadendo nel tuo programma. È possibile registrare molte informazioni su ciò che accade nel programma - ogni iterazione del ciclo, ogni volta che viene chiamata una funzione importante - e se nel programma sono presenti dei condizionali, `logzero` registrerà il percorso seguito dal programma (`if` o `else`). Ma ricorda che non puoi scaricare più di 250 MB di dati dalla ISS.

Di seguito è riportato un esempio di base di come `logzero` può essere utilizzato per tenere traccia delle iterazioni del ciclo:

```
from logzero import logger, logfile
from time import sleep

logfile("events.log")

for i in range(10):
    logger.info(f"Loop number {i+1} started")
    ...
    sleep(60)
```

I due tipi principali di voce di log che è possibile utilizzare sono `logger.info()` per registrare le informazioni e `logger.error()` quando il programma riscontra un errore imprevisto o gestisce un'eccezione. Ci sono anche `logger.warning()` e `logger.debug()`.

È consigliabile usare sempre la libreria `logzero` (per registrare eventi importanti che si verificano durante l'esperimento), anche se si scrivono anche i dati del sensore in un file.

Una volta che hai finito di scrivere il tuo programma e ritieni che fornisca la stima della velocità ISS nel formato corretto e segua le migliori pratiche come la registrazione e la gestione degli errori, è fondamentale testare a fondo il tuo programma utilizzando lo strumento di riproduzione Astro Pi.

Migliorare l'accuratezza del programma

La velocità media della ISS non è un segreto, ma la velocità della ISS non è realmente costante e ci sono diversi fattori che possono influenzarla, ad esempio l'altitudine. Se l'altitudine è cambiata, allora la ISS deve aver sparato i suoi booster e quindi la velocità deve essere cambiata. Per aumentare le tue possibilità di far funzionare il tuo programma sulla ISS, chiediti se il tuo programma è abbastanza sensibile ai sottili cambiamenti che influenzeranno la velocità a cui viaggia la ISS.

Sul sito web N2YO.com, è possibile vedere i dati in tempo reale della ISS che mostrano come cambia la sua altitudine durante l'orbita. Puoi anche vedere quando la ISS passerà la prossima volta sopra la tua posizione.

Medie

Se il programma calcola più stime per la velocità della ISS (ad esempio, calcolando la velocità da sequenze di due foto), sarà necessario decidere come ridurre queste stime in un unico numero quando si scrive il file result.txt. Se si utilizzasse una media semplice, si potrebbe esplorare l'accuratezza di altre misure statistiche, come la mediana e altri percentili?

C'è molto spazio per essere creativi quando si migliora l'accuratezza della stima. Un metodo consiste nell'essere selettivi su quali foto o dati utilizzare per calcolare la stima. Se è possibile determinare che una specifica sequenza di dati è la più affidabile, è possibile ponderare maggiormente questi dati nella stima finale.

Fai attenzione quando alleni il tuo programma in modo che sia ipersensibile all'esatta sequenza mostrata quando usi Astro Pi Replay: la sequenza sulla ISS sarà diversa e tu vorrai che il tuo programma sia accurato sulla ISS più di ogni altra cosa!

Se il tuo metodo di calcolo della velocità si basa sul progetto Calcola la velocità della ISS (in inglese) usando le foto, allora forse potresti utilizzare tecniche di visione artificiale o apprendimento automatico per classificare le foto da cui è più facile stimare. Ad esempio, è possibile eseguire l'inferenza dell'apprendimento automatico in tempo reale per valutare l'accuratezza della stima o le condizioni al di sotto dell'ISS.

Testare il programma

Ora dovresti testare il tuo programma utilizzando Astro Pi Replay. In questo modo si ottengono le migliori possibilità di successo e di assicurarsi che funzioni a bordo della ISS. Quando il Controllo Missione Astro Pi riceve il tuo programma, verrà testato e valutato utilizzando Astro Pi Replay e, in caso di successo, su un Astro Pi sulla Terra. Centinaia di team inviano programmi a Mission Space Lab ogni anno e, sfortunatamente, non c'è abbastanza tempo per verificare la presenza di errori o errori di codice di debug. Se il tuo programma presenta errori durante il test, il tuo programma non sarà idoneo per l'esecuzione sulla ISS.

Se hai seguito questa guida fin dall'inizio, dovresti aver già visto lo strumento Astro Pi Replay Tool online o aver installato la versione offline come plug-in Thonny. Le istruzioni per l'installazione sono disponibili in precedenza in questa guida e nella documentazione di Astro Pi Replay.

Per testare il tuo programma e simularlo in esecuzione a bordo della ISS, vai allo strumento Astro Pi Replay Tool e invia il tuo file di programma. Se stai utilizzando il plug-in Astro Pi Replay con Thonny, esegui il tuo codice `main.py` tramite il plug-in Astro Pi Replay aprendo il menu Esegui e facendo clic su Astro-Pi-Replay.

Il codice dovrebbe essere completato entro 10 minuti.

Al termine, verifica che sia stato creato un file `result.txt` nella cartella del progetto con una struttura valida. Inoltre, osserva eventuali altri file di output creati dal progetto. Controlla che i file salvati non abbiano superato il limite di 250 MB o che includano tipi di file consentiti dalle regole. Infine, controlla i tuoi registri per eventuali errori.

Se stai utilizzando la versione online di Astro Pi Replay, puoi scaricare un file zip dell'output del tuo programma.

Se vedi degli errori, o il programma non fa quello che ti aspettavi, dovrai risolverli prima di inviare il tuo codice, per assicurarti di avere le migliori possibilità di raggiungere lo "stato di volo". È possibile eseguire nuovamente l'esperimento con lo strumento Astro Pi Replay tutte le volte che è necessario fino a quando non si è sicuri che il programma funzioni.

Testa il tuo programma con Astro Pi Replay Tool e controlla l'output per eventuali problemi o comportamenti imprevisti.

Checklist del programma

Anche il tuo programma verrà analizzato per assicurarsi che aderisca al regolamento. Prenditi il tempo ora per leggerlo e rivedere il tuo programma per assicurarti che sia adatto.

Controlla il tuo programma rispetto al regolamento del Mission Space Lab (in inglese).

Molti team del Mission Space Lab non hanno eseguito i loro programmi sulla ISS a causa di alcuni errori comuni o errori nei loro programmi. Di seguito troverai un elenco di errori comuni con descrizioni dei motivi per cui influenzano i programmi in esecuzione sulla ISS

Errori comuni

Apertura e chiusura ripetute della fotocamera

Se si creano più oggetti `Camera`, ad esempio in un ciclo, è probabile che il Raspberry Pi esaurisca la memoria e che il programma non venga accettato dal Mission Control di Astro Pi.

Non utilizzare immagini a risoluzione massima

Attenzione che la distanza di campionamento del suolo, o GSD, può cambiare con la risoluzione finale dell'immagine.

Il valore utilizzato nella guida del progetto Calcola la velocità della ISS utilizzando le foto (in inglese) è valido per le immagini a piena risoluzione `(4056x3040)`, ma non necessariamente per risoluzioni inferiori. Per questo motivo, si consiglia di acquisire immagini a piena risoluzione.

Memorizzare più di 42 immagini

Il programma non è autorizzato a conservare più di 42 immagini alla fine dei 10 minuti, anche se può memorizzarne di più mentre è in esecuzione.

Input dell'utente

Il tuo programma non può fare affidamento sull'interazione con un astronauta per funzionare.

Utilizzare la matrice LED

Il programma non è autorizzato a utilizzare la matrice LED.

Scarsa documentazione

Quando hai creato un software utile e vuoi condividerlo con altre persone, un passaggio cruciale è la creazione di una documentazione che aiuti le persone a capire cosa fa il programma, come funziona e come possono usarlo. Assicurati che il tuo programma contenga commenti e che il metodo utilizzato per determinare la velocità della ISS sia ben spiegato.

La guida al progetto (in inglese) mostra il modo consigliato per aggiungere commenti utili al programma.

Nota: qualsiasi tentativo di nascondere o rendere difficile la comprensione delle operazioni eseguite da un pezzo di codice può comportare la squalifica. E, naturalmente, non ci dovrebbe essere un linguaggio scurrile o maleducato nel tuo codice.

Overfitting ai dati riprodotti

Il codice deve rispondere alle immagini e ai dati dei sensori sulla ISS e non fare affidamento su punti di riferimento o aree geografiche specifiche mostrate nelle sequenze utilizzate in Astro Pi Replay.

Utilizzare percorsi di file assoluti

Assicurarsi di non utilizzare percorsi specifici per i file di dati. Utilizzare la variabile `__file__`.

Non salvare i dati immediatamente

Assicurarsi che tutti i dati sperimentali vengano scritti in un file non appena vengono registrati. Evitare di salvare i dati in un elenco interno o in un dizionario man mano che si procede e di scrivere tutto in un file alla fine dell'esperimento, perché se l'esperimento termina bruscamente a causa di un errore o del superamento del limite di tempo di 10 minuti, non si otterrà alcun dato.

Esaurimento dello spazio

È possibile produrre fino a 250 MB di dati. Ricorda che la dimensione di un file immagine dipenderà non solo dalla risoluzione, ma anche dalla quantità di dettagli presenti nell'immagine: una foto di un muro bianco vuoto sarà più piccola di una foto di un paesaggio. L'uso di Astro Pi Replay ti darà una buona idea di quante foto sarai in grado di scattare.

Dimenticando di chiamare la funzione

Abbiamo visto casi in cui i team hanno scritto una funzione ma si sono dimenticati di chiamarla nel loro programma `main.py`: attenzione!

Salvare in directory che non esistono

Un certo numero di team desidera organizzare i propri dati in directory come dati, immagini, ecc. Questo di per sé è davvero una buona cosa, ma è facile dimenticare di creare queste directory prima di scriverci.

Networking

Per motivi di sicurezza, il programma non è autorizzato ad accedere alla rete sulla ISS. Non deve tentare di aprire una presa, accedere a Internet o stabilire una connessione di rete di alcun tipo. Ciò include le connessioni di rete locali all'Astro Pi stesso.

Tentativo di eseguire un altro programma

Oltre a non essere in grado di utilizzare alcuna rete, il tuo programma non è autorizzato a eseguire un altro programma o qualsiasi comando che normalmente digiteresti nella finestra del terminale del Raspberry Pi, come vcgencmd.

Thread multipli

Se è necessario eseguire più di un'operazione alla volta, è possibile utilizzare un processo multithreading. Esistono diverse librerie Python che consentono di includere questo tipo di multitasking nel codice. Tuttavia, per fare ciò sugli Astro Pi, è consentito utilizzare solo la libreria di threading.

Utilizzare la libreria di threading solo se assolutamente necessario. La gestione dei thread può essere complicata e, poiché il programma verrà eseguito come parte di una sequenza di molti altri programmi, è necessario assicurarsi che il precedente sia terminato senza problemi prima di avviare il successivo. I thread non autorizzati possono comportarsi in modo imprevisto e occupare troppe risorse di sistema. Se si utilizzano thread nel codice, è necessario assicurarsi che siano tutti gestiti con cura e chiusi in modo pulito alla fine del programma. È inoltre necessario assicurarsi che i commenti nel codice spieghino chiaramente come si ottiene questo risultato.

Impostazione del tempo di esecuzione del programma troppo breve

Alcuni team impostano il tempo di esecuzione del programma su un valore basso (ad esempio 1 minuto) per il test e poi dimenticano di riportarlo a un valore appropriato. Assicurati di utilizzare il più possibile la fascia oraria assegnata.

Rivedi di nuovo il tuo programma. Riesci a individuare qualcuno degli errori comuni nel tuo programma?

Invio del tuo lavoro

Se sei soddisfatto del tuo programma e lo hai eseguito utilizzando Astro Pi Replay, hai letto il regolamento di Mission Space Lab sul sito web di Astro Pi e hai controllato e ricontrollato la lista di controllo del programma Mission Space Lab, tutto ciò che resta da fare è chiudere il tuo lavoro e chiedere al mentore del tuo team di inviarcelo.

Preparazione di un file zip

Se non sai come comprimere la cartella del tuo progetto in un file zip, parla con il tuo tutor, che sarà in grado di dirti il processo corretto per il tuo sistema operativo.

Rimozione di file non necessari

Il file zip non deve avere una dimensione superiore a 3 MB, a meno che non includa un modello .tflite, nel qual caso può avere una dimensione massima di 7 MB per adattarsi alle dimensioni del modello. Ciò significa, ad esempio, che non è possibile fornire i propri file effemeridi (ad esempio de421.bsp o de440s.bsp), poiché superano il limite di dimensione.

Entrambi i file de421.bsp e de440s.bsp sono disponibili sugli Astro Pi. Se il tuo programma ne ha bisogno, puoi accedervi tramite la libreria astro_pi_orbit inclusa, come mostrato nel seguente codice:

```
from astro_pi_orbit import de421, de440s
print(de421)
print(de440s)
```

Se non stai utilizzando un Raspberry Pi, dovrai eseguire questo frammento utilizzando lo strumento Astro Pi Replay per accedere alla libreria astro_pi_orbit.

Genera un file zip per il tuo progetto e chiedi al tuo tutor di inviarcelo prima della scadenza.

Altre risorse

Ci sono una vasta gamma di risorse disponibili per aiutarti ad avere successo in ogni fase del tuo viaggio in Astro Pi.

1. I progetti di esempio Coral e gli esempi di fotocamera Coral offrono una buona panoramica di ciò che è possibile fare con l'acceleratore di apprendimento automatico Coral.
2. I tutorial Python di OpenCV spiegano come fare ogni sorta di cose interessanti come l'apprendimento automatico, il rilevamento dei bordi e il tracciamento degli oggetti.
3. La galleria di esempi di EarthPy illustra come elaborare le immagini satellitari.
4. Questa pagina GitHub contiene un elenco curato di risorse relative allo spazio.
5. Questo elenco di set di dati satellitari annotati può essere utilizzato per addestrare qualsiasi modello di apprendimento automatico.
6. I team che desiderano una sfida più grande troveranno qui una risorsa completa sull'utilizzo dell'apprendimento automatico con immagini aeree e satellitari: tieni presente che è piuttosto avanzato.

Infine, non dimenticare che i collegamenti alla documentazione per ciascuna libreria sono disponibili nella sezione Ambiente Python di Astro Pi se hai bisogno di informazioni specifiche su come utilizzare una particolare libreria.

Buona Fortuna!