

# SESSION 1 – PYTHON AND AEROSPACE INTRODUCTION

## Objectives

The objectives of this session are:

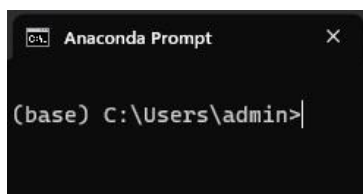
1. Install Anaconda Python manager, Python 3.10, and VS code.
2. Create environment in Anaconda with python 3.10.
3. Install Poliastro python packages using conda install.
4. Open VS code and choose the 3.10 interpreter. Write a program in python to do the following:
  - a. Input the input variables from a text file and from the user.
  - b. Implement mathematical operations on the inputs.
  - c. Output the output variables using print and in a text file.
5. Create a GitHub account and upload your code and a pdf explanation of what it is in a GitHub repository.

Note: The variables and mathematical operations here are related to the calculations of orbits. This session forms the foundation of using python for doing aerospace calculations. References to the equations and underlying physics will be provided for further learning.

## Chapter 1: Anaconda and Python

Python is the most common programming language due to its ease of use and loads of inbuilt functions and libraries. Unlike other languages, it would be quite easy to find something similar to what you are trying to do already existing in python. If not, there are always AI models that can write code quickly. This course is quite complicated, so it is highly encouraged to use all the tools at your disposal to write code. Be it finding existing libraries or asking AI models. Simplify the process for yourself now and learn from the code you have and the reports you submit by revisiting them later.

Anaconda is an environment for python which helps manage various packages you have for running python. It also makes it easier to download libraries.




The installation of poliastro python packages is done through this anaconda prompt.

## Downloads and installation

Here are the steps to follow to install it and get libraries:

1. Download and install Anaconda from the [Anaconda Website](#).
2. Download and install python version 3.10 from [python website](#).



Python version	Maintenance status	First released	End of support	Release schedule
3.13	prerelease	2024-10-01 (planned)	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569

3. Go to Conda Prompt and install the packages under poliaastro using [this link](#).

poliaastro requires a number of Python packages, notably:

- [Astropy](#), for physical units and time handling
- [NumPy](#), for basic numerical routines
- [jplephem](#), for the planetary ephemerides using SPICE kernels
- [matplotlib](#), for static orbit plotting
- [numba](#) (when using CPython), for accelerating the code
- [Plotly](#), for interactive orbit plotting
- [SciPy](#), for root finding and numerical propagation

poliaastro is supported on Linux, macOS and Windows on Python 3.8 to 3.10.

### Using conda

The easiest and fastest way to get the package up and running is to install poliaastro using [conda](#):

```
$ conda install -c conda-forge poliaastro
```

or, better yet, using [mamba](#), which is a super fast replacement for `conda`:

```
$ conda install -c conda-forge mamba
$ mamba install -c conda-forge poliaastro
```

If all cannot be installed, at least install NumPy and Plotly. Note that these only work with 3.10.

Please use youtube and AI models to ask how to install these things.

4. Download and install VScode from the [Visual Studio Code Website](#).

VScode is the actual code interface we use. The rest of the stuff is just the background to make it work the way we want.

# Chapter 2: Implementing Python for Aerospace

## Introduction

To learn something and get the hang of it, it must be used for something. So, instead of focusing on python learning itself, we just start by solving some calculation and code related problems with python and discover the code requirements in the process.

To get an introduction to python however, go through [this python for beginners playlist](#) on YouTube. When trying to write code, always ask the AI models for help as well.

## Aerospace calculations

We will do the preliminary calculations for a satellite in orbit and get various orbit parameters like speed, time period, time in horizon, etc using python. This is to get used to Python and you need to submit the code and results from this.

We divide this code setup into 3 sections: inputs, code, outputs. Inputs and outputs are text files where the values are in plain text and the code uses the input text file to do calculations and make the output text file.

## Inputs

The input text file contains the values we need for calculations that we need to search online. These are:

Variable name	Value	Units	Meaning	Link
Rearth	6378	km	Radius of the earth	<a href="#">Radius wiki pg</a>
Gconst	6.674E-11	SI units: $m^3kg^{-1}s^{-2}$	Universal Gravitational constant	<a href="#">G wiki pg</a>
Mearth	5.792E+24	kg	Mass of the earth	<a href="#">Mass wiki pg</a>

The input.txt file will contain:

Rearth (km): 6378

Gconst (SI): 6.674E-11

Mearth (kg): 5.792E+24

From the next session onward, all text files will be shared separately and only the explanation will be in the problem statement.

For this session, we have one input which we get from the user. We get the height above the earth in km (orbit height) from the user while the program is running. So input **Horbit** from the user.

## Equations

### Orbital Radius:

$$R = R_{earth} + R_{orbit} \quad \text{in km}$$

Convert that to meters (x1000) to put in subsequent equations.

### Speed: [https://en.wikipedia.org/wiki/Orbital\\_speed](https://en.wikipedia.org/wiki/Orbital_speed)

$$v = \sqrt{\frac{GM}{R}}$$

Note that R here is the radius of the orbit and not that of earth alone.

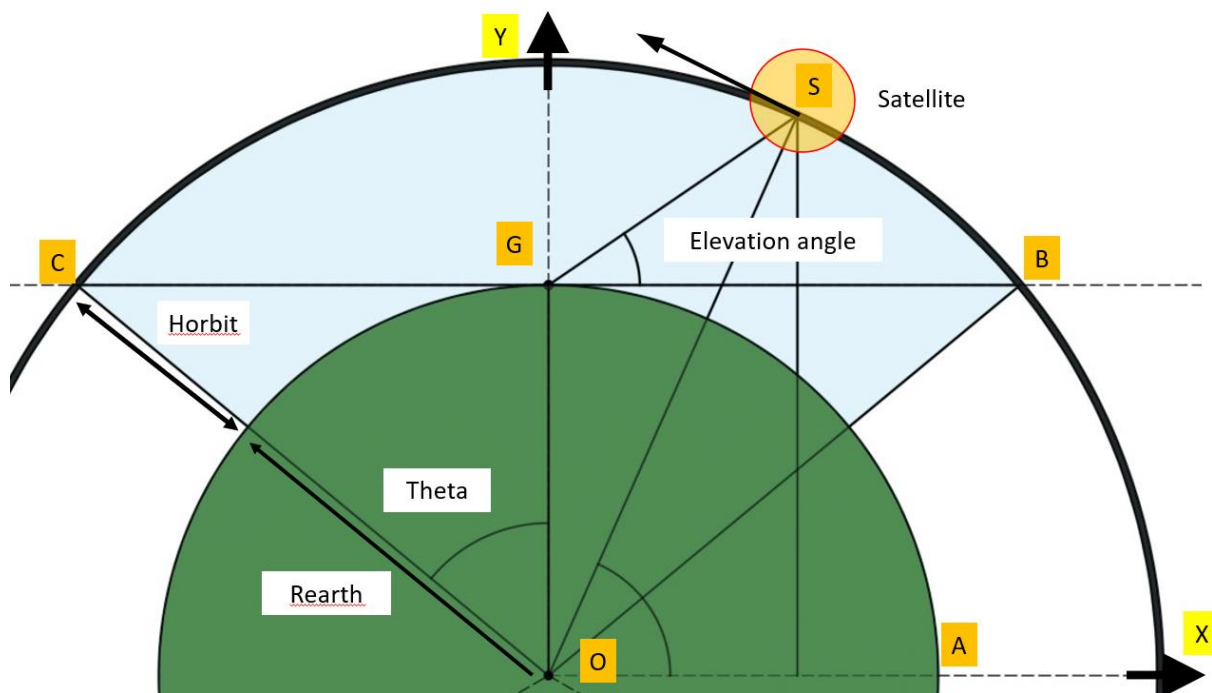
### Time period: [https://en.wikipedia.org/wiki/Orbital\\_period](https://en.wikipedia.org/wiki/Orbital_period)

$$T = \frac{2\pi R^{3/2}}{\sqrt{GM}}$$

### Angle in horizon:

$$\theta = \arccos\left(\frac{R_{earth}}{R}\right)$$

This is my own derivation from the figure shown below.



The satellite (S) is in the horizon over the point G (ground station) from when it is between point B and point C. The elevation from the horizontal is given by the elevation angle. The angle for which the satellite stays in the horizon is COB or  $2 \times \text{COG}$  or  $2\text{Theta}$ . OG is Rearth so theta is given by the above equation.

### Time in horizon:

$$t_{horizon} = \theta * \frac{T}{\pi}$$

This follows from the total time period being taken to go through an angle of  $2\pi$ , so to go through  $2\theta$  radians, we need proportionate amount of time.

## Code

This section just converts the equations and inputs into the outputs using python.

The code for this session is pasted here.

```
import numpy as np
import math

inpdat = open("inputs.txt",'r') # open reading file
line = inpdat.readline() # get the line
row = line.split(":") # get row as an array
Rearth = float(row[1])
line = inpdat.readline() # get the next line
row = line.split(":")
Gconst = float(row[1])
line = inpdat.readline() # get the next line
row = line.split(":")
Mearth = float(row[1])
inpdat.close() # close the file after done reading

Horbit = float(input("Enter orbit height in km: ")) # input from user

# Calculations
R = (Rearth + Horbit)*1000.0 # in meters
v = np.sqrt((Gconst*Mearth)/R) # in m/s
T = (2*np.pi*math.pow(R,1.5))/(np.sqrt(Gconst*Mearth)) # in seconds
Hang = np.arccos((Rearth*1000.0)/R) # angle in horizon in radians # Rearth
multiplied by 1000 to get in meters
Thor = (Hang/np.pi)*T # total time in horizon in seconds

outdat = open("outputs.txt", 'w') # open new file and truncate all data in it
outdat.write("Rorbit(km) \t Vorbit (km/s) \t Torbit(min) \t Theta(deg) \t\n"
Thorizon(min)\n")
# converting units for variables
Rorbit = round(R/1000.0,16)
Vorbit = round(v/1000.0,4)
Torbit = round(T/60.0,3)
Theta = round((Hang*180)/np.pi,3)
Thorizon = round(Thor/60.0,3)
# need to round off the values to make them easier to read

# writing in output file
outdat.write(str(Rorbit) + "\t\t" + str(Vorbit) + "\t\t" + str(Torbit) +
"\t\t" + str(Theta) + "\t\t" + str(Thorizon) + "\n")
```

Next session onward, the code will be put in a separate solution folder along with a dedicated solution file. For this session, both problem statement and solution are clubbed into one file for an easier start to the process.

## Outputs

This is the output file for a 600km high orbit:

Rorbit(km)	Vorbit (km/s)	Torbit(min)	Theta(deg)	Thorizon(min)
6971.0	7.4466	98.031	23.946	13.041

Note that some values make more sense in different units compared to what is best for calculation. So, we give the velocity in km/s, the orbital period and time in horizon in minutes, and the angle in horizon in degrees. So, we convert from the original units before outputting the values using print and write.

## Chapter 3: GitHub

GitHub is an online platform to showcase your programming portfolio. Most of the files on GitHub are for code and its explanation, but sometimes, other items like CAD are also put there.

Follow the following steps for the GitHub account and accessing the assignment:

1. Create a GitHub account on the GitHub website.
2. Find my repository on [my GitHub profile](#) for this course – [Python4Aerospace](#).
3. Download all files for [session 1 of the course](#).

Check out the format of the submission:

1. Code folder containing the code files and the input and output text files.
2. Pdf explaining what you did in the code for each chapter in your own words especially your conclusion and key learnings.

Note that for this session, the problem and solution are in the same file so there is not much input needed from your side. From the next session onward, the problem statement will be given a week before the solution file and code. You try to solve for 1 week based on the instructions given.

# Conclusion and Key Learnings

## Learnings

- Understand the basics of python, anaconda, and vscode installation.
- Understand the basics of python coding, functions, read, and write.
- Implement aerospace related equations in python to see how coding is used in space!

## Course completion instructions

- From the next session onward, the output text will be given 1 week after the session so you can try the solution once yourself.
- To complete the course, you must submit the solutions (input, code, output) and explanation pdf to a new repository on GitHub for each session.