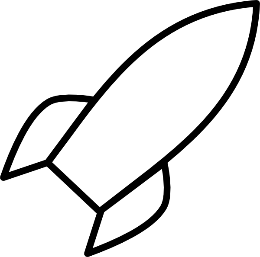
**Handout Workshop *National Open Day***  
Aerospace Engineering, TU Delft

Welcome to the workshop! Today, we are going to simulate the **first 500 metres** of a simplified rocket launch in Python. From this simulation, we will extract data and plot it afterwards. It is allowed to work together with your neighbours.

1. First of all, listen carefully to the instructions and the Python crash course. You will need this information during the workshop. In case you would have forgotten some of the Python syntax, you can have a look at the Appendix.
2. Open the file rocketsimplot.py in Spyder, which is the Integrated Development Environment (IDE) that we will be using during this workshop. At the left side, you will see the script in which you have to work. At the bottom right side, you find the IPython console that executes the code. In order to run your code, press F5.
3. Generally, there are 3 main parts in the script:

* **The preparation:** In this part, the simulation is initialised and the required parameters, constants and initial conditions are defined. It is not necessary to work in here but make sure to have a look at it!
* **The simulation:** This is the main and only part in which you have to work. It contains the actual simulation. You will need the parameters, constants and initial conditions from the preparation.
* **The results:** This part contains the code that shows the rocket launch characteristics after the simulation. It is not necessary to work in this part.

1. Assuming that all the forces start from the centre of gravity, let’s have a look at the following Free Body Diagram to understand the fundamentals of a rocket launch:



**Fthr**

**Fdrag**

**Fgrav**

**h**

**+**

With Fthr the thrust force from the engines, Fgrav the force due to earth’s gravitational force and Fdrag the drag force caused by the atmosphere.

1. Using the **parameters, constants and initial conditions** from the preparation part, we can now find the formulas for the three main forces. Keep in mind that the **thrust force** is given **per engine** and that we have **multiple engines**! As you did not yet learn the formula for Fdrag, the **aerodynamic drag force** is defined as **half the drag coefficient times the air density times the vertical velocity squared times the frontal surface area**.

* Fgrav = ……………………………………………………………………...…[EQ 1]
* Fthr = ……………………………………………………………………...…...[EQ 2]
* Fdrag = ………………………………………………………………………...[EQ 3]

1. If you have found the 3 equations, you can code them in the Python script. You will recognise them in the simulation part on **line 64, 65 and 66**. Make sure to use the **correct variable names** for the parameters, constants and initial conditions from the preparation part! Otherwise, Python won’t recognise them.
2. Now that we have Equation 1, 2 and 3, it is possible to combine them and calculate the total force on the rocket Ftot. Keep the **correct signs** with regard to the **coordinate system** of the Free Body Diagram in mind:

* Ftot = …………………………………………………………...…………...…[EQ 4]

If you have found Equation 4, you can code it in Python on **line 69**.

1. The total force is now defined in Ftot. Together with the current mass of the rocket, the vertical acceleration can be calculated using **Newton’s second law**. This law states that the total force is equal to the total mass multiplied by the acceleration. With this information, the vertical acceleration ay can be calculated:

* ay = …………………………………………………………………………..…[EQ 5]

If you have found Equation 5, you can code it in Python on **line 72**.

1. Lastly, we need to update the mass m of the rocket. This cannot be assumed to be constant. It should be calculated numerically and needs to be updated every loop. Python wise, generally the following can be said:

* m = m – Δm [EQ 6]

Mathematically, this does not make any sense. However for Python, it totally does! In fact, this can be translated as ‘**assign the current mass in variable m, which is the old value m from previous loop minus the difference between these two loops**’. This means that in order to update mass m, Δm should be calculated first. This can be calculated using the total fuelflow from the preparation part and the **already defined** time step dt from the simulation part (**line 75**).

* Δm = ……………………………………………………………………………[EQ 7]

Filling in Equation 7 (Δm) in Equation 6, we can finally calculate the new mass:

* m = ……………………………………………………………………………...[EQ 8]

If you have found Equation 8, you can write it in Python on **line 80**.

1. If you have finished the previous tasks, you should be able to run your code by **pressing F5**. First you will see a Pygame simulation screen (**the screen opens minimised**), visualising the rocket launch you coded. The simulation ends when the rocket reaches an altitude of 500 metres. After that, the simulation screen will close and the **rocket launch characteristic data** will be plotted in the IPython console.

If you are ready, feel free to play around with the parameters, constants and initial conditions from the preparation part and see what happens…

**Appendix *National Open Day***  
Aerospace Engineering, TU Delft

**Python syntax cheat sheet**

* Assign a variable in the memory of the computer
  + Variable assignment: x = 4
* Basic calculations
  + Summation: a + 3
  + Subtraction: a - 3
  + Division: a / 3
  + Multiplication: a \* 3
  + Exponentiation: a \*\* 3
  + Remainder: a % 3
* Conditions
  + Equals: x == 4
  + Does not equal: x != 4
  + Less than: x < 4
  + Less than or equal to: x <= 4
  + Greater than: x > 4
  + Greater than or equal to: x >= 4

**General tips and tricks**

* In Python, comment lines start with a ‘#’.
* Keep in mind than indentations matter in Python!
* To run the code, press ‘F5’.
* To close the pygame visualisation window, close the window or press ‘ESC’. If it would be stuck, type drawing.pg.quit() in the IPython console and press ‘enter’.
* **Use the assigned variable name instead of the value itself!**