ENGG1811: Computing For Engineers

2025 Term 2

More on NumPy

Week 8: Friday 25th July, 2025

Friday 16:00 - 18:00 | OboeME304

Today

Reminders

More NumPy Functions

Lab Tips

Reminders

Reminders

- Assignment 2 is out!
  - Due 5pm Friday Week 10 (08/08/2025)
  - Check WebCMS3 for details
  - Same advice as last time:
    start early and seek help when needed
- Self-directed lab 2 on Matlab
  - Due 5pm Friday Week 10 (08/08/2025)
    - Note that this is due very close assignment two! More incentive to start early so you're not rushing to complete both tasks

More NumPy Functions

NumPy: Arange

- `np.arange(start, stop, step)`
  - NumPy equivalent of the `range()` function
    - `start` is the starting value
    - `stop` is the 'ending' value — remember it will stop right before this actual value
    - `step` is the step-size — which is now allowed to be a float
- Usage:
  - `np.arange(6) = array([0, 1, 2, 3, 4, 5])`
  - `np.arange(1, 6) = array([1, 2, 3, 4, 5])`
  - `np.arange(4, 6, 0.5) = array([4, 4.5, 5, 5.5])`

NumPy: Zeros

- `np.zeros(shape, dtype = "float")`
  - Creates an array of a certain shape filled only with zeros
    - `shape` is the desired # of rows and columns you want for the resulting array
    - If you only specify one value for `shape`, then the resulting array will be one-dimensional
  - **Question:** Why would this even be useful?
- Usage:
  - `np.zeros(5) = arr([0., 0., 0., 0., 0.])`
  - `np.zeros((1, 5)) = arr([[0., 0., 0., 0., 0.]])`.
  - **Question:** What is the difference between the above two results? Why?

(N.B. `arr` is shorthand for `array`)

NumPy: Zeros-Like

- `np.zeros_like(array, dtype=None)`
  - Creates an array with a shape of the provided array, and then fills it only with zeros
    - `array` is the reference array; NumPy will copy its shape
    - **Question:** What does the `dtype=None` mean?
- Usage:
  - `array1 = np.array([1, 2, 3, 4, 5])`
  - `np.zeros_like(array1) = array([0, 0, 0, 0, 0])`
  - `np.zeros_like(array1, dtype = float) = array([0., 0., 0., 0., 0.])`
  - **Question:** What is the difference between these two outputs?

NumPy: Max-Min & Argmax-Argmin

- `np.max(array, axis = None)`
  `np.min(array, axis = None)`
  - Finds the maximum or minimum **value** of an array.
    - `array` is the given array in which we want to find the maximum/minimum value
- `np.argmax(array, axis = None)`
  `np.argmin(array, axis = None)`
  - Find the **index** of the maximum or minimum value
    - `array` is the given array in which we want to find the maximum/minimum value
- Let `array2 = np.array([4, 7, -2, 6, 9])`
  - **Question:** What is `np.max()`, `np.min()`, `np.argmax()`, `np.argmin()` for the above array?

NumPy: Max-Min & Argmax-Argmin (cont)

- Consider the array below, called `table`

4	10	5	2
6	9	12	3
11	8	7	1

- Here we would have
  - `np.min(table, axis=0) == array([4, 8, 5, 1])`
  - `np.max(table, axis=1) == array([10, 12, 11])`
  - `np.argmax(table, axis=0) == array([2, 0, 1, 1], dtype=int64)`
  - `np.argmin(table, axis=1) == array([3, 3, 3], dtype=int64)`

NumPy: Where

1	2	3
4	5	6

- `np.where(boolean_condition, x, y)`
  - Acts like a search function — finds the indices where a boolean condition is true.
    - `boolean_condition` is the logical condition we are searching for
    - `x, y` are *optional* arguments — wherever the `boolean_condition` is true, it will be replaced by `x`, otherwise it will be replaced by `y`
- Usage:
  - Let `arr1` be the array at the top-right.
    - `np.where(arr1 > 3) == (array([1, 1, 1], dtype=int64), array([0, 1, 2], dtype=int64))`
  - Let `arr2 = np.array([4, 7, -2])`
    - `np.where(arr2 < 0) == (array([2], dtype=int64))`

Lab Tips

Lab Tips

- Part A:
  - If you are not getting the same shaped graph as is shown in the lab-spec, double-check your equations carefully
- Part B & C:
  - Part B and C use the same logic, Part B is just to help you understand what you need to do for Part C on a smaller scale, and is directly applicable
- Part D:
  - The starter code provides `landing_pos_array` and `landing_time_array` for a range of angles. Recall that the specification allows you to assume a strawberry lands successfully to feed the alien if its distance is between 89.5 and 90.5 metres. Our goal is to find the launch angle that feeds the alien in the shortest time.

Feedback

Feel free to provide anonymous feedback about the lab!

Feedback Form