0 Foreword

How are the tasks rated?

I want to say randomly, but they aren't. This is my own estimate of how I think my students will perform based on their level and of the quality of the lecture I've given previously. 5 stars are the exercises that will take them a significant amount of time and may require external knowledge/research. The one 1 stars easy and shouldn't take much effort at all. All the others lie somewhere inbetween on this range.

I'm stuck! What next?

Do let me know if some exercise made you seriously stumble. Take a break and go for a walk, maybe try solve it tomorrow, perhaps you are tired. If you can't crach that exercise to matter what be sure to inform me at the start of our next lesson, we'll go through it.

There is a builtin function (or a simple combination of those) that does exactly what the task asks me to do, can I use it (them)?

There for sure is and you may use it! Knowing your way around the standard library is very important, however, you should only use the functions if you are sure you could implement them yourself. You can `list(set([1, 1, 2]))` to keep only the unique elements, but can you do it with plain `for` loops? If that isn't the focus of the task you can take the shortcut, otheriwse reconsider.

Why is the document so exquisitely formatted?

I like it more that way. I am also practising LaTeX in general, it's really lovely.

Hey, are you sure execrise X in section Y is correct?

No I am not! Message me if you think there is a mistake since there very well might be one.

Contents

1 Exercises

1.1 Simple Stuff

Prerequisites: control flow (branching, iteration), IO, arithmetic, atomic types.

The Good Old Days ★☆☆☆☆

Input: An integer 4.
Output: The word `"Elephant"`.

In	Out
`4`	`Elephant`

Echos In The Well ★☆☆☆☆

Input: String S with no line breaks.
Output: Said string S.

In	Out
`Hello`	`Hello`

Equation of a Line ★☆☆☆☆

Input: Two integers k and b, $k \neq 0$.
Output: Such value x, that it satisfies the equation $kx + b = 0$.

Wait, what? ★★☆☆☆

Input: Two integers a and b.
Output: The product of a and b.
Note: You may not use the multiplication operation.

In	Out
`1` `0`	`0`
`7` `8`	`56`

Late'o'clock ★★☆☆☆

Input: An integer $0 \leq h < 24$. Hours on a clock.
Note: Convert the given time h to the 12-hour clock format.
Output: First the time h in 12-hour clock format, then `"am"` or `"pm"` depending on the time.

In	Out
`0`	`12am`
`8`	`8am`
`13`	`1pm`

Quadratic Equations ★★☆☆☆

Input: Three integers a, b and c.
Output: Find all values of x, such that $ax^2 + bx + c = 0$.
Note: If there are no possible values of x output `"NaN"` (not a number). The values should not be repeated.

In	Out

`1`
`-1`
`-6`
	`-2`
	`3`

Qubic Equation ★★★☆☆

Input: Four integers a, b, c and d.
Output: Find all values of x, such that $ax^3 + bx^2 + cx + d = 0$.
Note: If there are no possible values of x output `"NaN"` (not a number). The values should not be repeated.
Hint: use Cardano's formula.

Euclid Approves ★☆☆☆☆

Input: Two integers a and b, sides of a right angled triangle.
Output: The hypotenuse c of the aforementioned triangle.

In	Out
`3` `4`	`5`

Euclid Disapproves ★★☆☆☆

Input: Two integers a and b, sides of a right angled triangle and an integer angle θ (given in degrees) between them.
Output: The third side of the triangle.
Hint: You may use `import math` to get some functions you might want.

Everyone but Euclid Approves ★★★☆☆

Input: An integer n the amount of following lines, $3 \leq n \leq 100$. Each following line i contains a number $-100 \leq a_i \leq 100$, a component of the vector $\hat{v} = \{a_1, a_2, \ldots, a_n\}$.
Output: The length of a vector $\|\hat{v}\|$.

Minmaxed ★☆☆☆☆

Input: Two integers, a and b.
Output: Two integers, first the largest of them two, next the smallest.

TreE ★★★☆☆

Input: An integer h, the height of the christmass tree.
Output: A christmas tree with total height $h + 1$, 1 being the trunk of said tree and h all the result of it.

In	Out

`4`	`e`
	`a a`
	`e e e`
	`a a a a`
	`a`

Sigma for Sum ★★☆☆☆

Input: An integer a such that $1 \leq a \leq 10^{10^{10}}$.
Output: The sum all the integers $1 + 2 + \cdots + a$.
Hint: Loop isn't the only way to go.

Factor!al ★★☆☆☆

Input: An integer a such that $1 \leq b \leq 10^5$.
Output: The product all the integers $1 \times 2 \times \cdots \times b$.
Hint: Lookup the arguments for `range` in the official Python3.x documentation.

Minmaxed 2: The Sequel ★★★☆☆

Input: Two integers, a and b.
Output: Two integers, first the largest of them two, next the smallest.
Note: You may only use `min()` or `max()`, not both. You may not use branching.

Set Product ★★☆☆☆

Input: Two integers, a and b where $a > 0$ and $b > 0$. They create sets of values: $A = \{0, 1, \ldots, a-1\}$ and $B = \{0, 1, \ldots, b-1\}$.
Output: Print out the product of the two sets.
Note: A product of two sets is a mapping of every element of one set to every element of another, e.g. for sets $C = \{1, 2\}$ and $D = \{3, 4\}$ the product is $C \times D = \{(1,3), (1,4), (2,3), (2,4)\}$.

1.2 Drawings & Lists

Prerequisites: `turtle` `mod` arithmetic, lists, turtle.

Note: Use `inputs = list(map(int, input().split()))` to parse the list of numbers into a variable, no need to know it works for now, just assume it's magic. This transforms an input of `1 2 3 4 5 6` into `[1, 2, 3, 4, 5, 6]`.

Fair Square ★☆☆☆☆

Input: An integer A such that $10 \leq A \leq 100$.
Output: Using `from turtle import Turtle`'s methods like `forward` and `right` draw a square of length A.

Fair Ngon ★★☆☆☆

Input: Two integers, A such that $10 \leq A \leq 100$ and N such that $2 \leq N \leq 20$.
Output: Using `Turtle` draw a regular polygon (an N-gon) with N sides and side length $5A$. Ensure that the turtle finishes in the same position as it started in. The turtle shouldn't draw over itself at any point.
Hint: Loops are your friend.

Trigonometry BFF ★★★☆☆

Input: Two integers, a and b.
Output: Using `Turtle` draw a graph of the function $y = a * sin(\frac{\pi x}{10}) + b$. From 0 to 200 and a graph of the function $y = b$. Print the final position of the turtle.
Hint: You can get sin and π with `from math import pi, sin`, they are accurate enough for this purpose.

The Fair Ngon ★★★★★

Input: Two integers, A such that $10 \leq A \leq 100$ and N such that $2 \leq N \leq 20$.
Output: Using `Turtle` draw a regular polygon (an N-gon) with N sides and side length $10A$. Ensure that the turtle finishes in the same position as it started in. You are only allowed to control the turtle with `goto`.
Hint: Trigonometry might help.

Tick Space Tick Space Tick ★★☆☆☆

Input: Two integers, $10 \leq L \leq 100$ and $1 \leq N \leq 15$.
Output: Draw a horizontal dotted line of N segments. The length of each segment should be L. The space between two segments should also be L.
Note: The turtle should start and end the drawing with a filled segment.
Hint: Make use of `turtle.penup`, `turle.penup` and `turtle.isdown`.

Fib ★★☆☆☆

Input: An integer n, $n > 1$.
Output: All terms from 0th to nth (inclusive) of the Fibonacci sequence. The Fibonacci sequence is defined as follows.

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

Note: You can solve this with both loops and recursion, maybe try it both ways? You may use this website to check how correct your result is.

In	Out
14	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Blaise's Blessing ★★★★★

Input: An integer N, $N > 0$.
Output: N rows of the pascal triangle.

In	Out
5	1 1 1 1 2 1 1 3 3 1 1 4 6 4 1

Average ★☆☆☆☆

Input: A list of space-separated numbers a_i of length $n \geq 1$.
Output: An average of all numbers, $\frac{a_1+a_2+\cdots+a_n}{n}$.

In	Out
1 2 3 4 5 6	3.5
0	0
1 -1	0

Furthest Apart ★★☆☆☆

Input: A list of space-separated numbers a_i of length $n \geq 2$.
Output: The largest distance between two numbers.
Note: Say in the 1st example the most spread-apart numbers are 1 and 5, and the distance between them is 4, the answer. In the second all the numbers are the same, the distance between any number and itself is 0.

In	Out
1 2 3 4 5	4
3 3 3 3 3	0

Reversed ★★☆☆☆

Input: A list of space-separated numbers a_i of length $n \geq 0$.
Output: The same list in reverse order.
Note: Try and reverse the list in-place, without creating a new one to copy the elements into.

In	Out
1 4 9	9 4 1
1	1

ROT K ★★★☆☆

Input: First line is an integer $-2n \le K \le 2n$. The next line is a list of space-separated numbers a_i of length $n \ge 0$.
Output: The same list with all of its elements shifted by K, to the right if K is positive and to the left if negative. When $K = 0$ the list should stay intact.
Note: Try and do this in-place, without creating a copy of a list!

In	Out
2 1 2 3 4 5	4 5 1 2 3
-1 1 2 3 4 5	2 3 4 5 1
-6 1 2 3 4 5	2 3 4 5 1
0 1 2 3 4 5	1 2 3 4 5

Shufflepuff ★☆☆☆☆

Input: A list of space-separated numbers a_i of length $n \ge 1$.
Output: The same list shuffled in any way you want. The shuffle of the same list must be the same across program runs (i.e. when given a list the output will always be the same, no matter the time of day, weather or else).

In	Out
1 2 3 4 5	3 4 5 1 2

Just Like In The Code! ★★★☆☆

Input: A list of space-separated words not containing any special symbols (i.e. newlines, carriage returns, bells), single or double quotes.
Output: A list, formatted like it is written in python. See the examples below.

In	Out
hi, oh dear world of sunshine	["hi,", "oh", "dear", "world", "of", "sunshine"]

Strong Neighbour ★★☆☆☆

Input: A list of space-separated numbers a_i of length $n \ge 2$.
Output: A list of maximum elements from each **adjecent triplet** of numbers.

In	Out
8 1 9 3 5 1 0 -8	9 9 9 5 5 1

Transpose ★★★☆☆

Input: A square matrix of numbers of size $n \times n$.
Output: A transpose of that matrix. The same matrix flipped across its main diagonal.

In	Out
1 4 7 2 5 8 3 6 9	1 2 3 4 5 6 7 8 9

Slice up ★★☆☆☆

Input: First line is an integer S, the second line is a list L of numbers containing S.
Output: Two parts of the list, before and after S, both not including S.

In	Out
0 3 6 9 0 2 4 8	3 6 9 2 4 8
0 0 88 77 66	 88 77 66

1.3 Dictionaries & Sets

1.4 Graphs & Networks

2 Labs

2.1 Hangman

This lab work is focused on implementing a digital variant of a classical paper-and-pencil game "Hangman". Create a small game that formulates a word and accepts user input. If the player input is a single letter the program should reveal all the copies of that letter in the word. If the word is entirely revealed the player gets one win. Every player is assigned a certain number of attempts per formulated words. If the letter is not in the word, a single attempt is taken away. If there are no more attempts left the player gets a loss. At the end of each round (independent of the win/loss result) the user is shown

Ext. Quiet Quitting

When encountering a keybord interrupt the app should silently close.

Ext. Save Files

Wins and losses can be tracked across program instances in a file. When starting without a save file such should be created and updated upon every win and loss.

Ext. Dictionary

The words to use should be picked from a dictionary file which can be specified by the user as the program's first command line argument.