Techniques for Improving Software Productivity - Project

Project Explanation:

We decided that we want to make one of the tools more accessible, and more usable & friendly.
We've figured that if someone writes a python code, he does not want to re-write his code in Dafny in order to check validation.

Our project was to create a website, with a friendly UI/UX where you can write python code, the code will be run & checked by a python interpreter, and then translated to Dafny and checked by Dafny verifier.

We've minimized our scope to basic python/Dafny commands, and in order to make it useful, added examples, and a code skeleton from the HW of the course "General Introduction to Programming and CS" ("אשנב למדעי המחשב") (Specifically question 8 from HW1 and Questions 5a and 5b from HW2).

We've used a very basic translation techniques (string manipulation rather than the full cycle of lexer/parser etc') due to the desired scope of the project (defined by Mooly ☺)

What we did / what the source code does:

We separated the project into a few major parts.

1. Build a friendly UI/UX interface (website)
2. Build an http server capable of:
    a. hosting the website
    b. receiving requests with python code (with added Dafny annotations)
    c. check the python code
    d. translate the python code to Dafny
    e. run Dafny Verifier on the translated code
    f. return the result (python prints/errors + dafny status/errors)

In order to translate the code, we've also started adding our own methods to expand Dafny, to support some operations that exists in python such as appending/removing items from a python list (…Dafny's array).

What we learned:

We've learnt how to setup Dafny on our local computers, and how to interact with it from a python code, we've learnt Dafny's 'annotations' in order to add them to our Python To Dafny translator.

We've also added our own Methods, written in Dafny

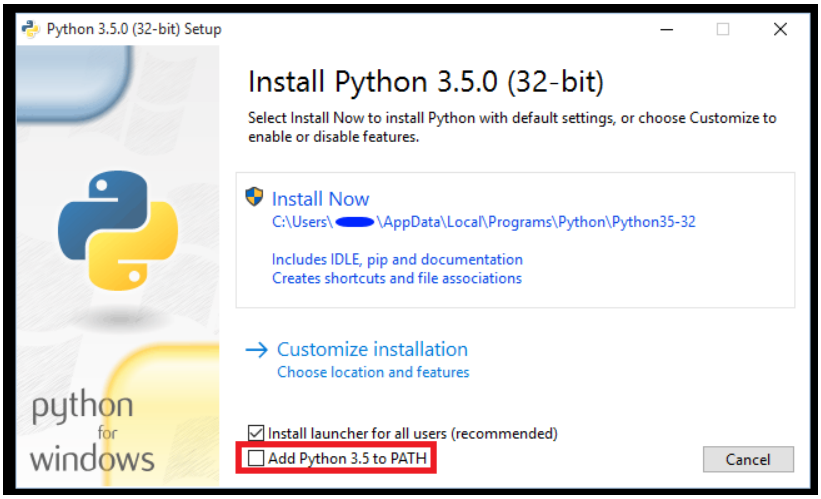
Besides that, (which is less the point of the course, but we've learnt nonetheless ☺) is how to setup and write a Python web server (Flask), and how to have the Python run/check other python code.
[We also had to refresh our python skills☺]

How to run it:

Hosting A local Python Server:

Requirements:

- Windows computer
- Python 3 (we used 3.6) installed and on the PATH variable
  (can be found as part of the install file for python…)



- Internet connection on first-time install (Pythons package manager [PIP] downloads *Flask* on 1st time run

Installation:

1. Make sure you have Python installed and on Windows' Path Variable
2. Extract the zip into a folder of your choice.
3. Run "Install.bat"
   Note: it may take a few minutes.
4. *If it did not open automatically* – open your web browser and go to http://localhost:5000/index

Source Code Explanation:

The zip includes 2 folders, and 3 files.

App - the folder that holds our source code, the webserver, the html, and pretty much everything

Static - a folder used by Flask (the webserver) to serve static content (a JS add-on we used for syntax highlight, and images used in the UI

Templates - holds the .html file – that contains the html/js/css of the UI/UX

__init__.py – holds the definition of our package (we've used this in order to organize our python code as a "package" and holds the initial run of Flask)

dafnyUtils.dfy - methods that we're adding when we send the translated code to Dafny's verifier

pyToDafTranslator.py - the code responsible of translating the Python code to Dafny

utils.py - a bunch of python functions that we wrote as utilities for the webserver

views.py - the entry point of the Flask webserver, it is responsible for receiving and sending http requests/responses. It's also incharge of the "flow" of the application

Dafny - downloaded from Dafny's website. This is version 1.9.8 download from https://github.com/Microsoft/dafny/releases

go.bat - a batch file to simply run "run.py" and launch the user's web browser with the url "localhost:5000\index"

install.bat - a bunch of commands to install everything else needed to run the project, it creates 2 "virtual python environments" [explained below], downloads Flask with PIP, and run "go.bat"

run.py - the python commands needed to start the Flask server

After running Install.bat, it will also create 2 more folders, and 2 specific files that we need.

CheckEnv - a virtual python environment to check the python code received on the webserver, we've added this as an extra layer of protection, in case someone tries to run malicious code. We've also added another level of security by not allowing the use of "import, __*__, this, eval, and exec"

CodeToCheck - a folder we've manually added to the virtual environment

input.py – will hold the code received by the user in order to run it on the virtual environment.

input.dfy – will hold the translated Python code to be used by Dafny

flask a virtual python environment holding "Flask" – used simply to "not mess" with the user other python installation and to avoid conflicting with other addons / configurations