University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

FINAL-YEAR PROJECT REPORT
BSc IN COMPUTER SCIENCE

Python Package for Visualisation of Sets

Nathan John Crowley

Department of Computer Science

Supervised by
Dr Rosane Minghim
Department of Computer Science

April 2022

Abstract

The project involves creating a Python Package to be used by students or academic professionals with the aim of visualising sets, using InteractiVenn's approach to Set Visualisation, to allow a greater understanding of given sets, and their unions and intersections. The Python package developed should allow users to input up to six sets of data to be visualised as an interactive Venn Diagram with functionality that:

1. Allow users to visualise the union and intersections of the given complex input sets.
2. Allow users to hover over set unions and present the elements of a given set.

The developed Python package should take into account that the users may have little or no technical skills, this should direct the design towards being as easy as possible to use for the target audience. The documentation of the Python package must have clear and concise information on its functionality, along with concrete examples displaying to the user the operations that can be performed.

The final year project includes:
1. Developing a Python package for Set Visualisation.
2. Experimental research conducted on previously established similar Set Visualisation systems; *InteractiVenn [1] and UpSet [16]*.
3. Evaluation of the Python package created in contrast to the earlier specified established system; *UpSet [16]* to highlight the advantages and disadvantages of each package given a number of examples.

- **crisply summarises the project in a single paragraph**
- **used to assess if a document is worthy of reading**
- ***appears on its own page***
- ***positioned verticalled on the page beginning roughly ⅓ down from the top***
- ***labelled with the word "Abstract" (bold face)***
- ***between 50-200 words***

Declaration of Originality

This is to certify that the work I am submitting is my own and has been done by me solely and not in consultation with anyone else. Neither I nor anyone else have submitted this work for assessment, either at University College Cork or elsewhere. I have read and understood University College Cork's exam regulations, plagiarism policy and Code of Honour. I understand that breaches of this declaration are serious issues and can incur penalties.

____

Signature of Student: Nathan Crowley
Date: Monday 25th April, 2022

- ***signed declaration that the work submitted is the fruits of your own individual effort.***
- ***report not accepted without***

Acknowledgements

This report and research would not have been possible without the guidance and support of my supervisor Dr Rosane Minghim. Dr.Minghim's supervision has been critical from the beginning of the project to the conclusive report. Her knowledge and management showcased in bi-weekly meetings, manifested in meeting reports as well as a continuous logbook of the project's progress, helped keep the project and myself on track to achieve the goals set for the project.

I would like to thank Frank Boehme, the administrator of the Final Year Project process for the BSc Computer Science degree program. Mr.Boehme's lecture at the beginning of the academic calendar assisted in many aspects of the report, such as report structure, key dates and preferred language, along with the information necessary for submission in a timely manner.

Contents

1. Introduction

1.1. Overview of Python Package

A Python package is a reusable set of Python instructions that a user can download and import into their project, allowing a user to take advantage of the created functionality.

This allows users of all technical levels the ability to quickly as well as easily import code to their project that may have either been; too time-costly to implement or above their technical abilities.

The Python package is suitable for the project as Python is widely regarded as an entry-level programming language [2] taking into account that the users may have little or no technical skills.

1.2. Project Aims

The goal of the project is to create a Python package to be used by students or academic professionals with the aim of visualising sets.

InteractiVenn [1] is an existing Venn Diagram visualisation tool which provides interactive venn diagrams that mirrors certain aspects of the functionality required. As a result, *InteractiVenn*. will serve as a benchmark for the Python package generated. Specifically the package will include the ability to allow users to input up to six sets of data to be visualised as an interactive Venn Diagram with functionality that:

1. Allow users to visualise the union and intersections of the given complex input sets.
2. Allow users to interact with the generated output of the Python package.

The Python package aims to provide users with a more accessible approach to set visualisation.

1.3. Deliverables

The project was completed by achieving specific goals at set intervals over the project's lifecycle. The specific goals include the *Outline Document,Extended Abstract* deliverables expected by University College Cork. Along with continuous development with bi-weekly meetings with my project supervisor, Dr. Rosanne Minghim.

1.3.1. Outline Document - October 2021.

A short analysis of the project and a broad plan of the steps to complete the work, ensuring that the student understood the assigned project brief along with building a calculated structure of work needed to complete the project.

1.3.2. Extended Abstract - February 2022.

A summary of the essential work completed over the project's lifecycle. Ensuring that the student had gained valuable experience and technical knowledge as a result of the work completed on the project.

2. Analysis (no code yet)

2.1. What are Venn Diagrams:

2.2. Venn Diagram history:

2.3. Venn Diagram Examples:

2.4. What is a Python API

3. Design (no code!!!!)

3.1. Project structure

The structure of the Python package created was developed continuously over the Python packages lifecycle. The Python package structure can be found at the public *Github repository for the Final Year Project [7]*. The base directory "/FianlYearProject" contains the source code and metadata belonging to the Python package created. The contents of the Python package structure is listed below:

1. **Deliverables**: base directory regarding the expected work from *University College Cork [8]* completed throughout the project's development. The expected deliverables can be found at Dr.Frank Boehme's *University College Cork's final year project website [9]*. This base directory "/Deliverables" contains four subdirectories, ;
   1.1. **Extended_Abstract**:
   1.2. **Final_Project_Report**:
   1.3. **Open_Day**:
   1.4. **OutlineDocument**:

2. **FYP_Drive:** directory containing detailed summaries of bi-weekly meetings hosted on *Microsoft Teams [10]* with the final year project's supervisor, Dr. Rosanne Minghim. Furthermore a 'Deliverables_What_is_Success' file contains the discussion between the student and the projects supervisor on what is expected from the project to be considered a successful final year project.

3. **Report_Tips:** directory containing constructive advice from the project's supervisor. This includes files supplied by the projects supervisor, such as "WritingTips_v8.pdf", in conjunction with advantageous report writing information in regards to the final year project deliverable. Additionally the reports completed during the third year *"Work Placement" CS3300* module were consulted as a reference for report writing.

4. **Research:** working directory of all experimental research completed over the final year project's development. This includes the source folder for the developed Python package along with sub-directories:
   4.1. **API_research**: directory containing research conducted on *Python web API [11]* examples implemented using the *Python web framework Flask [12]* , to implement and test a web API in Python. Following the research we concluded that the Python web API would not be a feasible technology to achieve success in the final year project.
   4.2. **Library_research**: directory containing research conducted on *Python Package [13]* as a technology for the final year project development. Included are sub-directories:

        4.2.1. **Examples**: directory containing two simple calculator Python packages used to experiment with the implementation as well as the download of a created Python package.
        4.2.2. **FYPlibrary**: directory containing an experimental Python package structure that later influenced the final year project's Python package structure.
    4.3. **PDF**: storage of back-up *Adobe PDF [14]* files that were used through the Python package's development.
    4.4. **Upset**: directory for the experimental installation and testing of the Python package "*pyupset"[15],* used as a Python implementation of the *UpSet API [16]*. UpSet API is described in the final year project's project brief as a means to evaluate the Python package created in contrast to a previously established set visualisation system.
    4.5. **VennDiagram_research**: directory containing more advanced experimental research into a Python package for set visualisation. Contains experimental sub-directories along with the source directory for the final year project Python package created:
        4.5.1. **API**: source directory for the Python package created, *DrawVennDiagram [17]*. The source directory is expanded on in chapter 3.2.
        4.5.2. **VennDiagramLibrary**: Includes earlier development versions of the Python package created. During the experimental development in "/VennDiagramLibrary" a fundamental error occurred regarding Python package uploads to *PyPI [3].* Concluding in a separate source directory "/API" being created.

5. **TestAPI**: working test directory for the Python package created. Contains a *jupyter notebook [18]* testing file, 'FYP_TESTING.ipynb' used to import, from PyPI, the Python package created, to test the outputs generated and evaluate them against expected output. Additionally included is the sample biological data file retrieved from the *InteractiVenn web API [1]*, *"test_model.ivenn"* ,used to test and evaluate the Python package created.

6. **venv**: directory for the *Python virtual environment [19], venv,.* The virtual environment allows developers to isolate their working environment, enabling developers to install packages locally in the working environment, escaping the need to globally install packages that may cause system issues.

7. **ReadME.md**: *markdown [20]* file containing instructions on how to interact with a project, including concrete examples followed by expected outputs. A brief *abstract* [21] is included giving a short overview of the project.

8. **test_set.ivenn**: sample biological data file retrieved from the *InteractiVenn web API [1]*, *"test_model.ivenn"* ,used to test and evaluate the Python package created conducted in the *jupyter notebook [18]* test file, found at "./TestAPI/'FYP_TESTING.ipynb".

3.2. Source Folder (/API) & File structure - __init__.py (no code!!!!)

The Python package created source directory can be found at the public *Github repository for the Final Year Project [7]*. The *base directory "/API"[22]* contains all source code and metadata. The *source directory "/DrawVennDiagram" [23]* can be found at "/FinalYearProject/Research/VennDiagram_reseach/API". The *source file "__init__.py" [24]* contains the Python code for the package.

3.2.1. Base directory "/API":

3.2.1.1. **build**: directory created when the Python package is created and built. Contains the source code for the Python package created.

3.2.1.2. **dist**: directory containing subdirectories of the numerous version releases of the Python package.

3.2.1.3. **setup.py**: Python file that is used to configure and build the Python package. States the *PyPI classifiers [5]*, containing information to categorise each Python package release which is needed to upload a built Python package to *PyPI [3]*. States the PyPI set up data such as:

- Python package name.
- Python package version number.
- Short description of the package.
- Python package author and author's email address.
- Python package licence.
- Installation requirements for the Python package.

3.2.1.4. **MANIFEST**.txt: Text file consisting of commands, one per line, used in the building of the Python package which states what file types to include or exclude in the "/dist/" directory.

3.2.1.5. **LICENSES**.txt: text file confirming the copyright licences rights of the author for the Python package created.

3.2.2. Source directory "/DrawVennDiagram":

3.2.2.1. **__init__.py**: Source file for the Python package created. Contains the Python code of the package, this is expanded on in chapter 3.2.3.

3.2.2.2. **requirements.txt**: text file inside the source directory that allows the Python package to automatically install any dependencies required for the running of the Python package. This is a crucial step that will allow users of all programming skills to interact with the Python package created.

3.2.3. Source File:

The source file "__init__.py" contains the Python code for the developed Python package. It comprises of several sections which are listed below:

3.2.3.1. **Dependency imports**: at the beginning of the source file, the necessary dependency imports are made.

3.2.3.2. **constants**: constant values for the variables seen later in the source code. Variables which may not be set by the user are automatically set here.

3.2.3.3. **drawVenn**: Python *Object-Oriented Programming [25]* class contains all the Python code in a *"drawVenn"* object that includes all necessary functions and encapsulation.

3.2.3.3.1. **__init__**: initialising function necessary in all Python Object-Oriented Programming classes to initialise the object. The parameters passed into the initialising function include the fundamental information needed for the Python package to operate.

3.2.3.3.2. **__str__**: overridden the Python string function to allow more control over the output of the Python package. The overridden string function automatically outputs the correct sized *Venn Diagram [26]* based on the length of the object's labels variable.

3.2.3.3.3. **getters & setters**: Getter function returns the value of a given variable. The setter function allows the user to adjust the object's variables through a function.

3.2.3.4. **read_data**: function built into the Python package that allows users to read in biological data files, with a similar format to the "*test_model.ivenn"* used for the testing of the Python package. The Python package initially only accepted integer arguments as set elements. This became an issue when the biological data set elements consisted of both integers as well as strings. The *"read_data"* function solves this issue by treating each set element as a string, then uniquely mapping each string to an integer representation. This unique integer representation is then used as an argument to the Python package therefore allowing the visualisation of biological sets.s

3.3. Technologies (no code !!! code in implementation)

3.3.1. Programming Language - Python:

Python is a high level, general purpose programming language. Python is widely regarded as an accessible programming language to users of all programming skill levels. Python is therefore suitable for this final year project as the aim is to create a simple and user-friendly Python package for set visualisation.

3.3.2. Python Package Hosting - PyPI / Test PyPI:

PyPi: Python Package Index [3], offers developers the ability to host their Python package The Python package is uploaded to PyPi by the developer, using the tools mentioned below in chapter 3.3.5 . The user of the project can import the Python package using the python Pip package installer for Python [4] or from the PyPi website [3]. Test PyPI [6] is a separate instance of PyPi. Test PyPi used to privately upload project versions and test importation prior to releasing a given public version on PyPi.

3.3.3. Project Tools:

- Project structure hosting - GitHub

  GitHub [26] is a version control system that allows developers to track the evolution of projects over time, throughout different versions using *Git[27]*. GitHub allowed the storing of the project structure, mentioned in chapter 3.1, on the cloud in a public repository. This is advantageous as the project's life cycle was over the span of six months therefore uploading iterations of the project to *GitHub* using *Git* ensured that progress would not be lost. GitHub's version control also allowed free development of the Python package with the protection that if a direction of development led to system issues, then the project could be easily reverted to an earlier state.

- Virtual Environment - venv:

  The *Python virtual environment [19], venv,.* allows developers to isolate their working environment, enabling developers to install packages locally in the working environment, escaping the need to globally install packages that may cause system issues.

3.3.4. Project Dependencies:

3.3.4.1. Matplotlib

Matplotlib[28] is an object-oriented Python package for generating and outputting visual representations of data.

3.3.4.2. Matplotlib-venn

Matplotlib-venn[29] is a Python package that builds on the foundations of *Matplotlib* allowing developers to generate Venn Diagram outputs.

3.3.5. Python Package Upload & Build:

3.3.5.1. Build

- wheel

Wheel [30] is a Python package used for installation and management of Python packages. *Wheel* possesses similarities to the *ZIP [31]* format however it is more applicable to Python projects.

- setuptools

Setuptools [32] is a Python package that allows developers to easily build and deploy their packages.

3.3.5.2. Upload - twine

Twine [33] is a tool for developers to publish their Python packages to *PyPI [3]*. Twine takes as an argument the Python package version from the "/dist" directory mentioned in chapter 3.2.1.2 , uploading the given verison to PyPI.

4. Implementation (code!!!)

4.1. Software implementation

4.1.1. What are the inputs

4.1.2. Who are the users

4.1.3. Api documentation

4.1.4. What makes a good api design

4.2. How is the Python package hosted (code !!! theory in design Technologies)

4.2.1. PyPI - how to upload

4.2.2. Test PyPI

4.2.3. Github

4.3. Technologies and file/folder structure CODE!!!

4.4. Python package documentation

4.4.1.

4.5. Errors???

4.5.1. Started with integer inputs so when it came to biology string inputs, a mapping function had to be created to convert biological strings to unique identifying integers.

4.5.2. 6 sets challenge??

4.5.3. interactivity??

5. Evaluation & Results (GRAPHS & OUTPUTS) (((Test API jupyter notebook)))

5.1. Testing the package

5.2. Appraisal of Completed System (does package meet the description)

5.3. Incomplete Work

“next steps for project”-extended abstract

“The future project state should extend the development from the previous chapter. The goals of future development include:

1) Adding functionality to the API to allow the user to position their mouse over the generated venn diagram output and if a venn diagram section is selected, output the elements in this given section of the venn diagram.

2) Current state of the API is limited to inputs of two or three sets, future states of the project should allow users to input up to six sets to the API.

Documentation must be created describing the resources, functionality available. that the user can implement. The documentation created should be accessible to users of all levels of technical experience. Concrete working examples will guide the user on how to interact with the API.

The Python API created must be evaluated in comparison to an existing set visualisation technology, UpSet. The evaluation should display the key benefits and disadvantages of each technology, through concrete examples given to both. Continued development of the Final Project Report deliverable, showcasing all development and evaluation completed for the project's lifecycle.
”

5.4. Does the api produce the expected outputs from the fyp abstract

5.4.1. Output vent diagram using python ?

5.4.2. Biology inputs?

5.4.3. 6 sets?

5.4.4. Interactive?

5.5. Compare api and Upset and Interactivenn

5.5.1. What is Upset good and bad

5.5.2. What is interactivenn good and bad

5.5.3. What is api good and bad

6. Conclusion

6.1. results of the project (overview)

6.2. How the project was conducted

6.2.1. What are my findings

6.3. What have i learned

6.4. Future work

6.5. Reflection

7. References

[1] InteractiVenn http://www.interactivenn.net/
[2] “Python is widely considered one of the easiest programming languages for a beginner to learn” https://brainstation.io/career-guides/how-long-does-it-take-to-learn-python#:~:text=Is%20it%20Hard%20to%20Learn,lot%20of%20practice%20and%20patience.
[3] PyPi, Python Package Index (PyPI) is a repository of software for the Python programming language.Developed and maintained by the Python community, for the Python community. (2022), https://pypi.org/
[4] Pip package installer for Python, https://pypi.org/project/pip/
[5] PyPi classifiers, https://pypi.org/classifiers/
[6] Test PyPI, https://test.pypi.org/
[7] Github repository for the Final Year Project, https://github.com/NathanCrowley/FinalYearProject
[8] University College Cork, https://www.ucc.ie/en/
[9] *University College Cork's final year project website*. https://project.cs.ucc.ie/
[10] *Microsoft Teams, https://www.microsoft.com/en-ie/microsoft-teams/log-in*
[11] Python web API, https://realpython.com/api-integration-in-python/
[12] *Python web framework Flask , https://flask.palletsprojects.com/en/2.0.x/quickstart/*
[13] *Python Package, https://www.udacity.com/blog/2021/01/what-is-a-python-package.html*
[14] *Adobe PDF, https://en.wikipedia.org/wiki/PDF*
[15] *pyupset, https://pypi.org/project/pyupset/*
[16] UpSet API, https://upset.app/
[17] Final Year Project python package created, DrawVennDiagram, https://pypi.org/project/DrawVennDiagram/
[18] *jupyter notebook, https://jupyter.org/*
[19] *Python virtual environment, https://docs.python.org/3/library/venv.html*
[20] *markdown files,* https://www.markdownguide.org/
[21] ReadME.md abstract, https://en.wikipedia.org/wiki/Abstract_(summary)
[22] Github Python package base directory “/API, https://github.com/NathanCrowley/FinalYearProject/tree/main/Research/VennDiagram_research/API
[23] Github Python package source directory “/DrawVennDiagram”, https://github.com/NathanCrowley/FinalYearProject/tree/main/Research/VennDiagram_research/API/build/lib/DrawVennDiagram
[24] Github Python package source file “__init__.py”, https://github.com/NathanCrowley/FinalYearProject/blob/main/Research/VennDiagram_research/API/build/lib/DrawVennDiagram/__init__.py
[25] *Object-Oriented Programming, https://www.geeksforgeeks.org/python-oops-concepts/#:~:text=In%20Python%2C%20object-oriented%20Programming,in%20the%20programming*.
[26] GitHub home, https://github.com/
[27] Git, https://github.com/
[28] Matplotlib, https://matplotlib.org/
[29] Matplotlib-venn, https://pypi.org/project/matplotlib-venn/
[30] *Python Wheel, https://pypi.org/project/wheel/*
[31] ZIP archive, https://en.wikipedia.org/wiki/ZIP_(file_format)
[32] Python Setuptools, https://pypi.org/project/setuptools/

[33] Twine, https://twine.readthedocs.io/en/stable/

“Include an entry that includes the publication details of the work in question in the References section:
[1] Jeremy Butterfield, ed. Fowler’s Dictionary of Modern English Usage.
4th. Oxford University Press, 2015. isbn: 978-0199661350.

[2] University College Cork. UCC Plagiarism Policy. url: www.ucc.ie/
en/exams/procedures-regulations/.

[3] Thomas H. Cormen et al. Introduction to Algorithms. 2nd. McGraw-
Hill Higher Education, 2001. isbn: 0070131511.

[4] Charles Darwin. On the origin of species. (Available online: www.gutenberg.
org/ebooks/22764). London: John Murray, 1859.

[5] Marc van Dongen. LATEX and Friends. url: csweb.ucc.ie/ ~dongen/
LAF/LAF.html.

—---

2 Expectations

The project should commence with research into which technologies should be used and what API's are best for outputting interactive models to the user through Python. The design of the Python API should have consideration for users coming from any Operating System, with minimal importing of dependencies needed, allowing for users of any technical skill level to operate.
Technologies that need to be decided on:

- Which technology for displaying interactive outputs through Python.
- How the documentation of the Python API will be presented to the user.
- Which file types the user will be inputting and relevant error handling measures.

Once the technologies have been chosen, the implementation may begin. This will involve the development of the Python API using the chosen technologies from the initial definition phase of the project. The implementation involves:

1) Creating a directory to store the Python library.
2) Creating a virtual environment for the directory.
3) Creating the content of the Python library.
4) Building the Python library.

Development of the documentation, to allow the users to understand the functionality of the API, may commence. Once the implementation and documentation have been completed to a satisfactory level, the final stage of the project may begin.

The final stage of the project will involve testing the Python API extensively, to discard unforeseen errors or missing functionality. The Python API will be evaluated by itself to assure it satisfies the requirements of the project. The Python API will also be evaluated against the existing *UpSet Visualisation* API to compare the two projects. The documentation will be finalised and completed. The project closure will involve public and peer testing of the developed Python API by the target audience.

"Most projects involve the development of some piece of software, though there are other possibilities as listed below. Expectations and the assessment yardsticks applied vary between different types of projects, reflecting the underlying diversity of the projects themselves. Clearly what may be appropriate in the case of a software development project may be pointless and irrelevant in the case of an algorithm design project.
Software development project The most common category of project involves the design and implementation of some substantial piece of software or perhaps a hardware-software system.
Research project (empirical) Other projects typically involve the experimental evaluation of some idea (algorithm, heuristic etc.). This may involve the bench-marking of some software package or a rigorous comparison of a number of competing algorithms or protocols.
Research project (investigative) Projects in this category might involve the development of a new algorithm for some problem or an enhancement of an existing algorithm."

3 Organisation and Structure

"This section specifies the standard organisation of a project report. It focuses mainly on software development projects but also touches the other types. Subsection 4.1 discusses the heart of the report and the sections into which should be subdivided. Subsection 4.2 specifies the format and sequenching of the other elements (title page, table of contents and so on)."

1. Main Sections
   1.1. Introduction

   "Set the scene - paint a picture - basically why and how?.

   Why?:

   - **Importance, relevance and interest to (the most important & relevant of):**
     - **humanity, science, computer science, department, yourself.**

   How?:

   - **this part of the introduction should be brief (< 1 page) - only give only a a general outline of approach to give direction to the thesis and whet the reader's appetite - don't write the entire thesis in the intro - else it's finished"**

   1.2. Analysis

   "How comprehensive is the student's literature review? Is the review analytic rather than descriptive? Are the strengths and weaknesses of existing work identified?"

   1.3. Design

   "Is the research methodology the right one? Are the experiments well designed? Are the experimental materials (e.g. prototype systems, etc.) well designed? Can the hypotheses be verified?"

   1.4. Implementation

   "Were the experimental materials properly constructed? Were the experiments properly conducted?"

   1.5. Evaluation

   *** have a documentation for API use ***

   My API vs UpSet:

   - **advantages**
   - **disadvantages**
   - **suitable utility for each**

   "

   - **Collect and run the API's of the two tools on collected data.**
   - **Evaluate the new API by itself and in comparison with UpSet.**

   "

“Are the results well reported (e.g. using tables, graphs, etc.)? Are the results analysed deeply and properly explained? Is there both quantitative and qualitative analysis of the results?”

1.6. Conclusion

1.7. Support elements

1.8. Back matter

6 Appendix

“You must acknowledge the sources the any published material that you make use of in your report. Failure to do so can have very adverse con-sequences [10]. The details of each source (author name, title of the work in question, publisher and so on) should be recorded in the References section at the back of the report.”

1. **Some useful resources**

   “For sound advice on (non-technical) writing in general, Strunk and White’s slim classic Elements of Style [12], a stalwart on US campuses for generations, is well worth reading.
   For a reliable source on spelling and related matters, you should refer to The Oxford English Dictionary [1]. There are copies in the library and the on-line version is quite useful. For guidance on punctuation and grammar, the authoritative Fowler’s Dictionary of Modern English Usage [9] is a good source. There is also a (limited) on-line version.
   The IEEE Computer Society has a useful style guide [11]. It is aimed at authors intending to publish in the organization’s technical journals and offers guidance on technical (specifically computer-related) writing.”

“All students are bound by UCC’s Plagiarism Policy [2] and are expected to familiarize themselves with the provisions (and penalties) of that policy. The principles set out below interpret this policy in the context of final year projects in Computer Science.
The overarching principle is that the presentation of the project, encom-passing both the project report and the underlying body of technical work, should be accurate and honest in describing what has been accomplished and in specifying the precise individual contribution of the examinee.”