Application for Automated Collection of Test Files  
 for CSS Class via HTTP and for Local Plagiarism Check:   
Testrek

**by**  
**Akshat Tandon**

Submitted in Partial Fulfillment of the  
Requirements for the Degree of  
Bachelor of Mechanical Engineering (Information and Automaton Technology)

in the  
Czech Technical University  
Faculty of Mechanical Engineering



© [Akshat Tandon]  
Czech Technical University  
Summer 2017

Copyright in this thesis rests with the author. Please ensure that any reproduction   
or re-use is done in accordance with the relevant national copyright legislation.

Ethics Statement

The author, whose name appears on the title page of this work, has conducted the research as a co-investigator, collaborator, or research assistant in a project approved in advance.

Abstract

This thesis focuses on analysing and automating the examination process for the ‘Computer Support for Study’ course taught at Faculty of Mechanical Engineering, Czech Technical University in Prague; and running the plagiarism check on answers from the test takers. An application called *Testrek* is built as a part for this thesis which should streamline much of the examination process for the CSS course. The application is written using Python programming language with support from external libraries which are discussed in the thesis. Testrek essentially checks and downloads the files from a publically accessible network location (public\_html folder), where answer files are uploaded by the test takers. The Plagiarism check is run using a library called fuzzywuzzy which in turn uses the Levenshtein distance to calculate the similarity between two strings. At the end, a technical walkthrough of the application is also provided for understanding of the use case.

In conclusion, the required automation of examination process has been achieved. Consequently, the analysis in this thesis open opportunity for further scope of automation and making the application even more dynamic to cover more courses.

**Keywords**: python, process automation, string comparison, HTML parsers, Testrek

Acknowledgements

I would like to express my deepest appreciation to Ing. Matouš Cejnek for all guidance that he has provided while preparation of this thesis work. Without his persistent help, motivation and immense knowledge in Python, this would not have been as doable as it had been.

I would like to thank Professor Ivo Bukovský for trusting in my ability to work on this thesis topic. The Python course taught by him provided a lot of help in familiarizing with many aspects of the programming language and its capabilities.

In addition, a big thanks to Kristyna Steidlova from Accenture sro for helping with producing the process procedures and for providing her valuable suggestions pertaining to different ways of approaching the automation from her experience and expertise.

Table of Contents

[Approval ii](#_Toc484728920)

[Ethics Statement iii](#_Toc484728921)

[Abstract iv](#_Toc484728922)

[Dedication v](#_Toc484728923)

[Acknowledgements vi](#_Toc484728924)

[Table of Contents vii](#_Toc484728925)

[List of Tables viii](#_Toc484728926)

[List of Figures ix](#_Toc484728927)

[List of Acronyms x](#_Toc484728928)

[Glossary xi](#_Toc484728929)

[Chapter 1. Introduction 1](#_Toc484728930)

[Chapter 2. Process Analysis 4](#_Toc484728931)

[2.1. Process Description 4](#_Toc484728932)

[Chapter 3. Proposed Solutions 5](#_Toc484728933)

[3.1. Standardization of the process 5](#_Toc484728934)

[3.2. Possible Solutions 5](#_Toc484728935)

[Chapter 4. Technologies and Methodologies 9](#_Toc484728936)

[4.1. Development Environment 9](#_Toc484728937)

[4.2. Programming Technologies 10](#_Toc484728938)

[4.2.2. Why Python? 15](#_Toc484728939)

[4.3. External Python libraries 15](#_Toc484728940)

[4.4. Development and Testing Methodology 18](#_Toc484728941)

[4.5. Plagiarism Check 20](#_Toc484728942)

[Chapter 5. Technical walkthrough 22](#_Toc484728943)

[Appendix A. An Example of an Appendix 25](#_Toc484728944)

[References 26](#_Toc484728945)

List of Acronyms

|  |  |
| --- | --- |
| CTU | Czech Technical University |
| CSS | Computer Support for Study |
| GUI | Graphical User Interface |
| IDE | Integrated Development Environment |
| PyPI | Python Package Index |
| TDD | Test Driven Development |
|  |  |
|  |  |

Glossary

|  |  |
| --- | --- |
| Python | An Object Oriented and interpreted Programming language |
| Script | Programs written for a special run-time environment |
| Open Source | The source code that is made available to the public under a license to study, change, and distribute the software to anyone and for any purpose |
| Database | a structured set of data held in a computer, |
| Hosting | A web hosting service is a type of Internet hosting service that allows individuals and organizations to make their website accessible via the World Wide Web |
| PEP8 | Python code style conventions[[1]](#endnote-1) |
| Object Oriented Programming | Computer programming in which we can define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure |
| Python Interactive Shell | Python’s command line utility |
| snippet | a small piece or brief code |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

The major motivation for this thesis comes from the fact that most of the work surrounding the examining of students is still done manually. This thesis introduces the need, scope and application that is built for the automation of the examination process for reducing the manual work. The software solution described and developed as a part of this thesis is tend to be designed in a way that it utilizes and requires the least resources and infrastructure for its operation. The application is made as dynamic as possible hence it can be used outside the set scope (i.e. for Computer Support for Study course), if the process of examination can be standardized for that course or subject. Different methods of approaching the automation are also described along with their benefits and shortcomings.

For the sake of simplicity, CSS course is used as an example all through this thesis, to analyze and demonstrate the challenges and solution proposals to different aspects of such an automation process.

When conducted manually, the whole examination process procedure is taken care of manually with the help of up to three or more people. From the preparation of question paper to reviewing the answers from students (most of the times in an unorganized way) and assigning grades. If a course or subject is registered by many students (which in fact is a case for the CSS course), for e.g. in order of hundreds then the whole process becomes even more cumbersome and hard to handle, leading to the slowdown of the review process and requirement of even more manual efforts.

The aim of this thesis is to provides a good understanding of the examination process where most of the work is done manually, along with ways to approach the automations of such nature. The problem that has been addressed in this thesis is mostly concerned with the organization and automation of the answer reviewing system and aid the plagiarism check. Also, a summary of how to achieve operational excellence by the deployment of small and cost effective automations is made part of this thesis.

The software solution ***Testrek***, developed mainly in Python as part of this thesis can help in organization of reviewing and running a preliminary plagiarism check on answers from the students taking part in the examination. The designing of the solution is dynamically approached so that the testing of any kind can use this solution if it can be standardized as mentioned in the 2nd Chapter. This thesis can also be used to conduct further research and development on automating the sub processes which are out of scope of this bachelor thesis.

There are information systems that already exists and can provide a potential solution for achieving the automation, if modifiable and extensible. For example, *Moodle*[[2]](#endnote-2) is a shared learning platform which is hosted by CTU to share study material with students. Although by default, there is no possibility for students to upload their answers during the examination or for teachers to run a plagiarism check, it can be extended in order to do similar tasks that Testrek would do but the efforts required for achieving the same results as Testrek would require more time, resources, expertise in other front and back end programming languages. Moreover, for integrating the extensible application to current *Moodle*, an extensive Black and White box testings will be required, ranging from regression and system tastings to integration testing. It is for all these reasons that it became necessary to write an autonomous application.

The information in this thesis is ordered in way so that one starts by understanding the aim and challenges faced due to the current process, followed by a description of the solution proposals and technologies used to develop the solution itself.

# Process Analysis

Process Analysis constitute an important part of this thesis as it helped in understanding the different aspects of the problem which are tried to be solved while designing the software solution described in *Chapter 3*.

## /Users/akshat/Documents/Tand_Labs/Testrek/_01_Designs/images/Thesis_Test_current.jpgProcess Description

When conducted manually, the process map of whole examination process is explained using a swim lane chart on the right. Next chapter contains further information about the scope of this thesis and a rough estimation of efforts required to automate the sub processes which are not included in the scope.

The chart shows the current process procedure, with the estimations of work and in some cases time required for a sub process to complete.

# Proposed Solutions

## Standardization of the process

Before designing any solution for automating the examination process, it was necessary to standardize it. This in fact proved to be crucial while searching for different ways to approach the solution. The standardization helped in deciding the best possible way to approach the designing of the software solution, in the limited time frame. To make the solution fail safe it was necessary to set many rules for different parts of the process, but the challenges was to still maintain its dynamical nature.

After a thorough analysis, the two most qualified solutions which can be build are listed below with the requirement for standardisation, complexity, maintainability, infrastructure and support.

## Possible Solutions

##### 1st Solution (Testrek) – Preferred

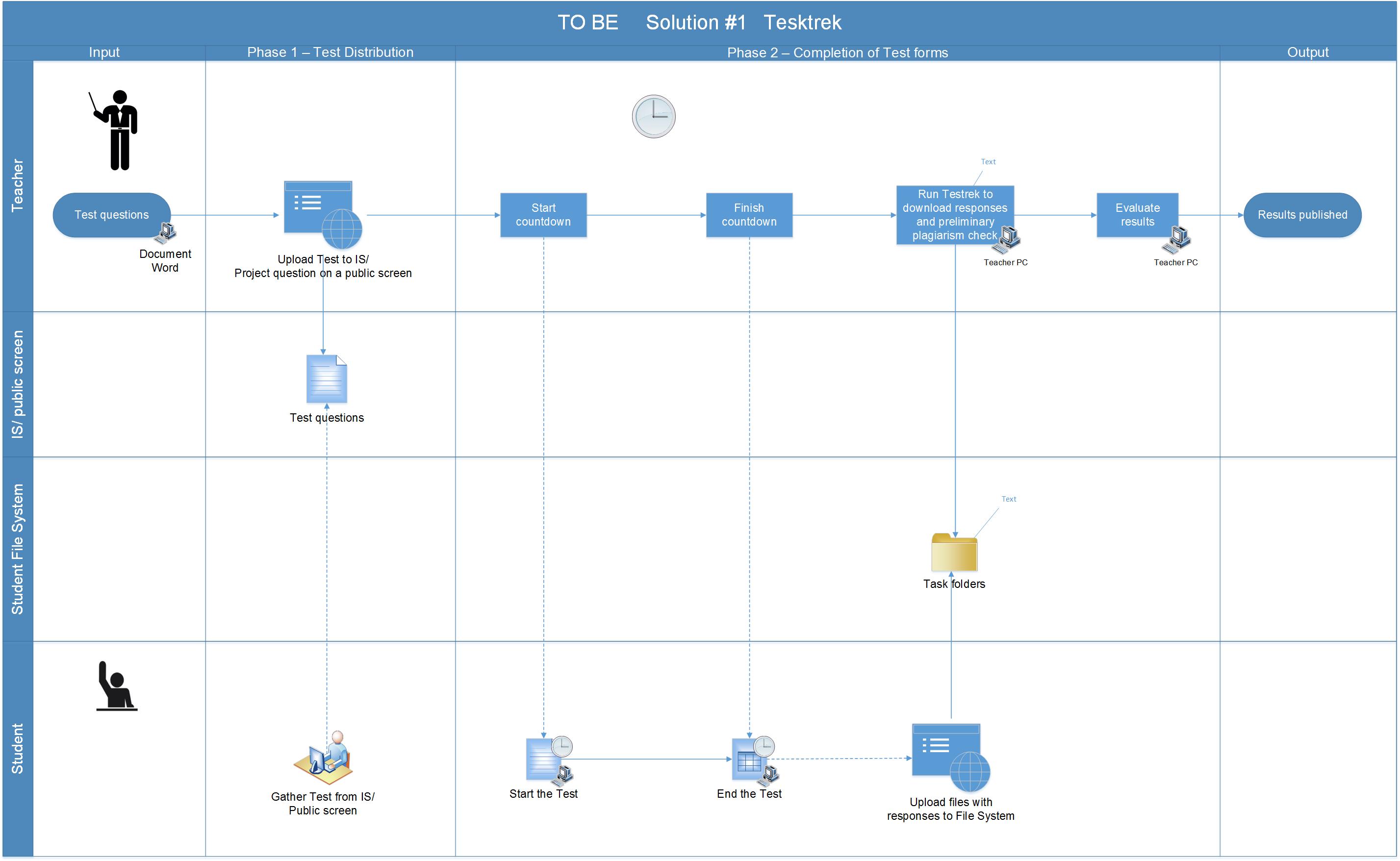
Basic Requirements for this solution to run:

1. Access to the internet
2. Disk space of about 30 Mega Bytes

This solution which is essentially a combination of scripts written in Python, is built as a part of this thesis. This solution was preferred because the requirements for running this application were very low and affordable. As the scripts are simple to understand, it would not require a lot of time to modify if the need be.

This solution also does not demand a lot of infrastructure hence it can run on most of the computers of today’s standard. Although it does not have a very user interaction to the of GUI, it is still easy to use and maintain. The challenging part for this solution is strictly standardizing many parts of the examination process, for example, where data is expected from the user.

The swim lane chart shows the process workflow if application like *Testrek* will be implemented for automation. A more thorough description of this solution can be found in the upcoming chapters.



##### 2nd Solution – Not preferred

As this solution will have to utilize a server, it will have a client side and the server side. Basic Requirements for this solution to run:

On client side:

1. Internet Connection and a modern web browser

On server side:

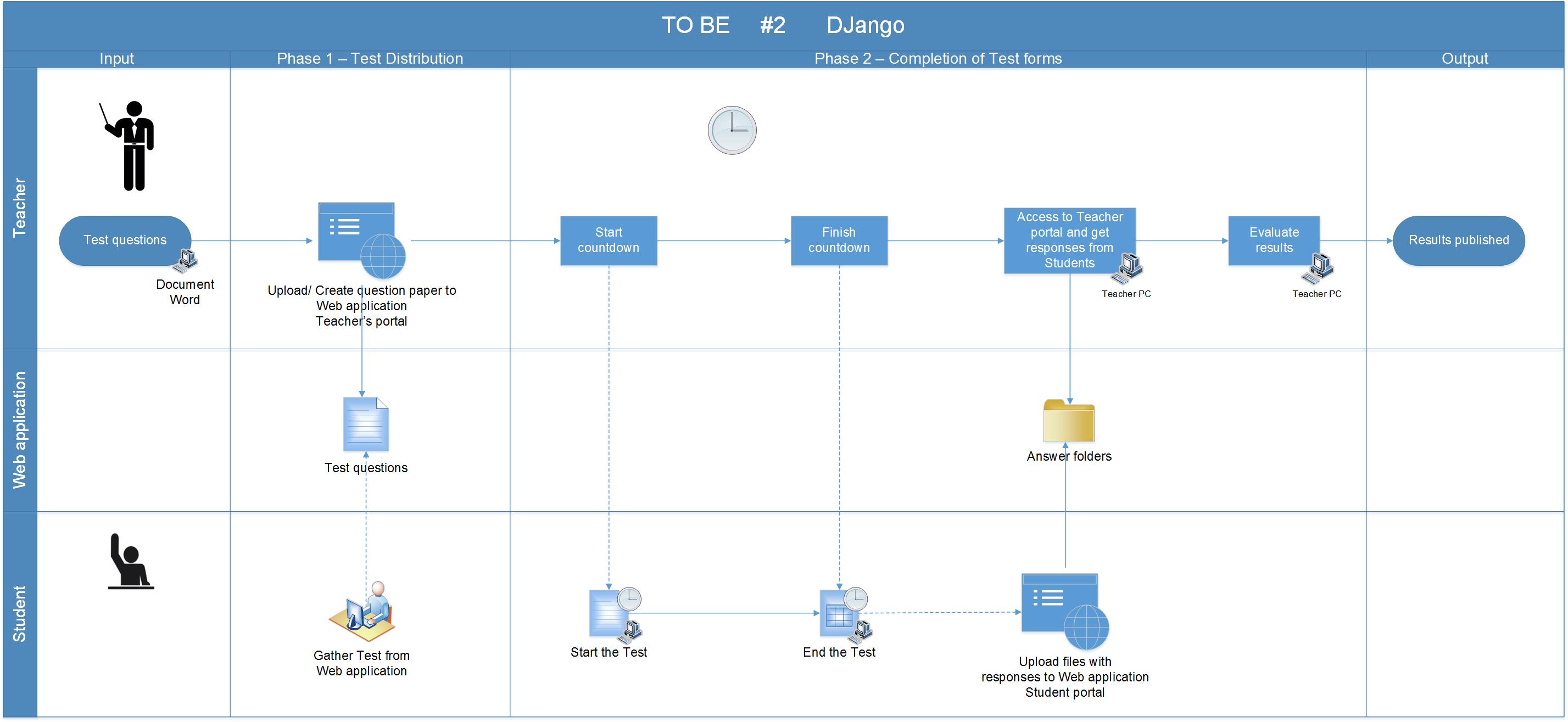
a. a database

1. a server supporting Python
2. an app hosting platform
3. maintenance and support

Building this application solution would require a robust web framework for Python, e.g. Flask, Django etc. It would utilize a database to store the information and a server to operate itself. Furthermore, the development process would require a lot more time and resources in comparison to the 1st solution.

Though the requirements are on the higher end when compared to the 1st solution, this solution which is essentially a web application will be more robust in its operation and delivery. This application would run on a web browser with an intensive GUI, hence will be more interactive for the users. One other essential characteristic of this solution would be the fact that it will not require as much standardization of the process as does the 1st Solution requires. Several validations can be run on the data as and when it is received, hence diminishing the need for much standardization.

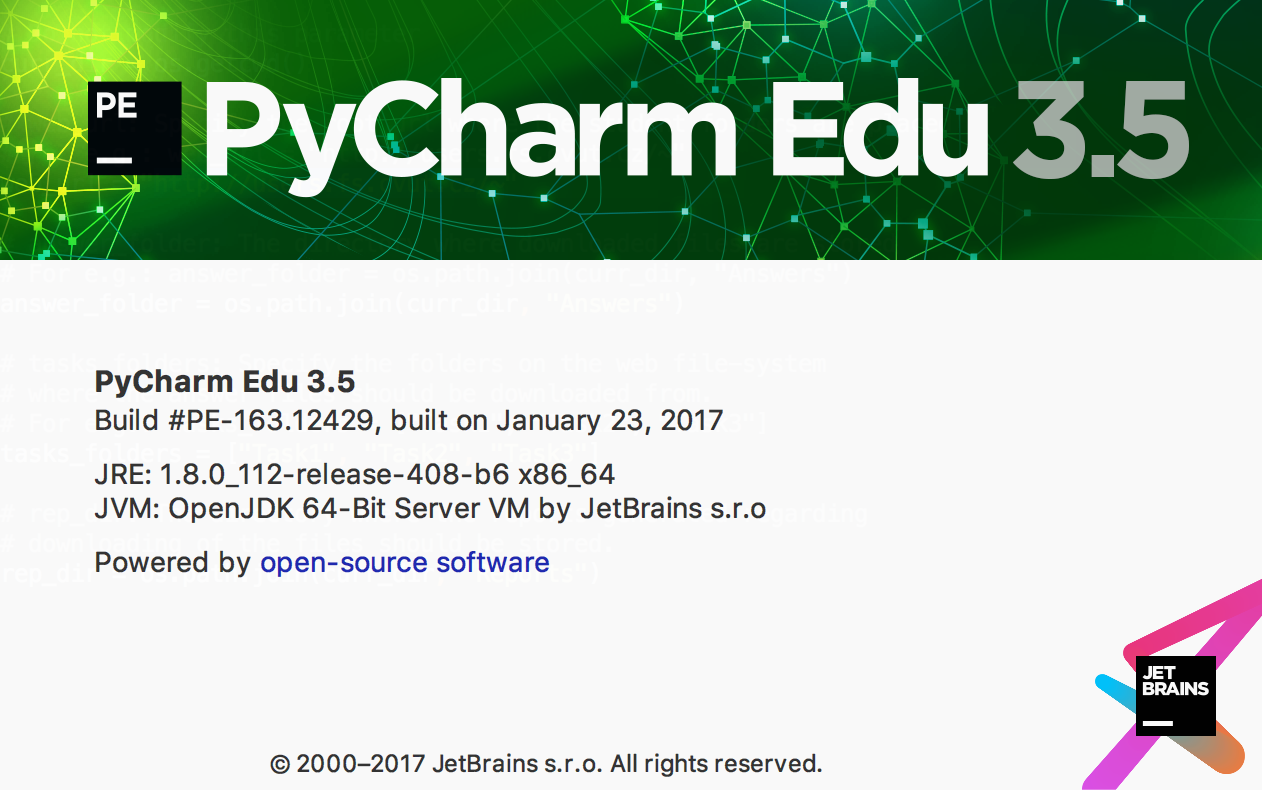
The swim lane chart below shows how the examination process would look like if a web application, written for example in python’s web framework Django, can be implemented to automate the examination process.



# Technologies and Methodologies

This chapter guides through different technologies and methodologies used, from development to the deployment of the application solution. The sub sections of this chapter consist of information about the platforms used for development, environment settings, programming technology and the libraries used.

## Development Environment

The solution had been built on a machine running MAC OS and the IDE or integrated Development Environment that was used to build the software solution is called PyCharm Edu 3.5[[3]](#endnote-3) provided by JetBrains s.r.o..

This IDE was selected as it has a great support for Python development and is an open-source software. It is best known for intelligent code completion, on-the-fly error checking and quick-fixes and easy project navigation. It helps keep quality under control with PEP8 checks, testing assistance, smart refactoring, and a host of inspections.

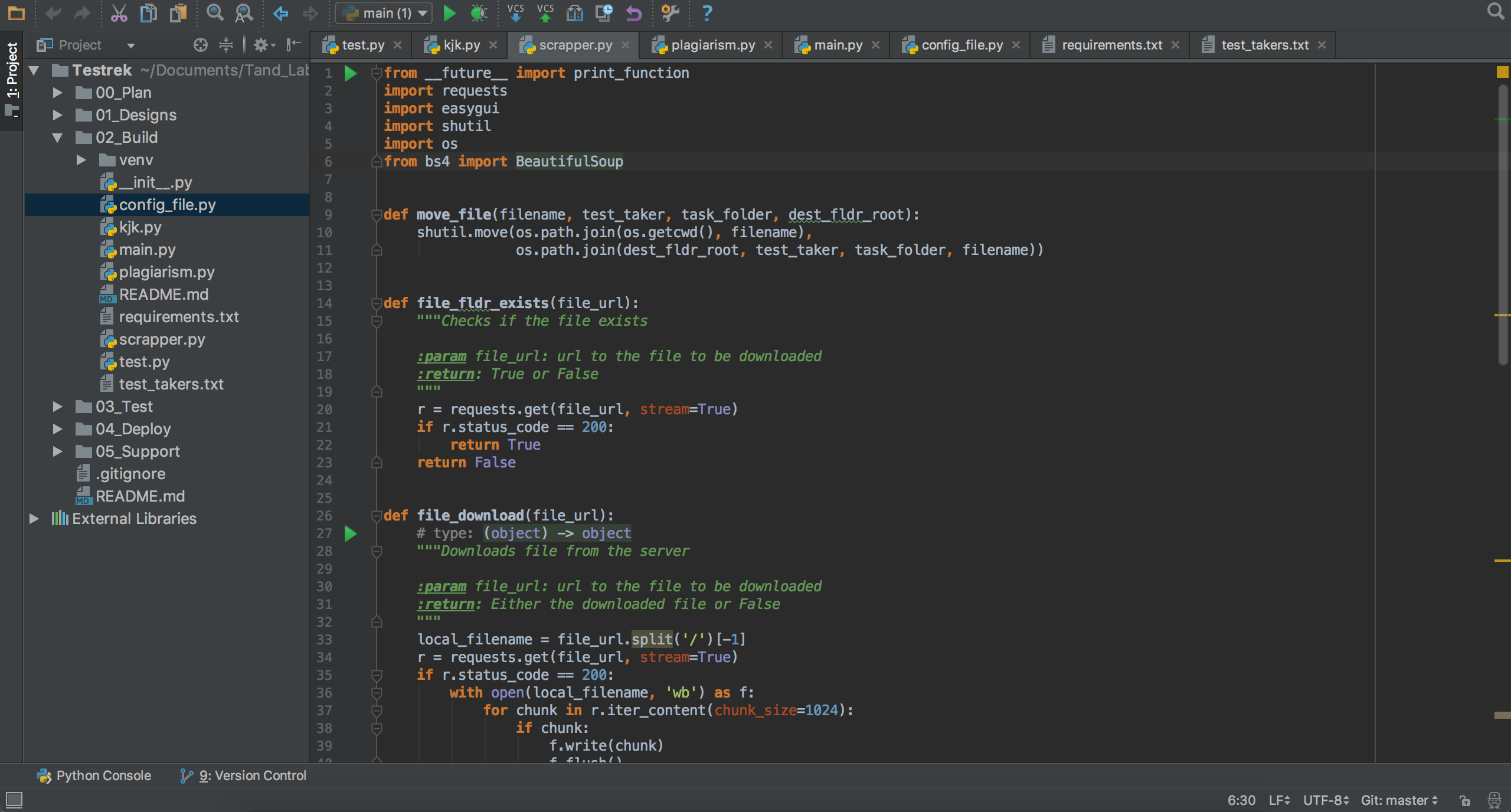


Figure 2 PyCharm Workspace

## Programming Technologies

The software solution *Testrek* is purely written in the programming language called Python. Python is an interpreted, multi-purpose programming language that can be used to write web applications, GUIs, scripts and much more. It is strongly and dynamically typed with focus given to its readability and productivity. With an immense support from the community around it which builds a great range of libraries, it has proved to be a powerful language for scientific use and mathematical modelling. It is a self-contained object oriented programming language that has an interactive shell, strong introspection, cross platform capabilities and a variant for specific use like CPython, JPython, IronPythnon etc.[[4]](#endnote-4)

The versions of Python used for building and testing the scripts for this thesis are Python 2.7.10 and Python 3.5.2.

##### Hello World in Python

Writing “Python” in the command line starts the Python interactive shell which can be used to write python commands.

|  |
| --- |
| #!/usr/bin/env python  print "Hello World!" |

##### Indentation is necessary

Unlike most other programming languages, Python cares about the indentation and structure of the code. A sample is shown below:

|  |
| --- |
| #!/usr/bin/env python  for i in range(1, 10):  print ("I am number " + str(i))  if i == 9:  print ("9, here again!") |

##### Comments

Comments in Python are written as shown below. Any string that is not assigned to a variable or function is regarded as a comment.

|  |
| --- |
| # One line comment  """  This is a multi-line  comment.  """  "Any string not assigned to a variable is a comment" |

##### Data Types

Python has built in support for primitive data type like strings, Numbers, Null, Booleans, Lists, Tuples and Dictionaries[[5]](#endnote-5). Python dynamically assign the data type after a variable is initialized, hence declaration of variable is not included in Python.

|  |
| --- |
| # Strings  address = "This is a string."  address\_long = """This is a  long string."""  # Numbers  # Integers  age = 9  year = int("2010")  #Float  pi = 3.14159  # Null  data = None  # Booleans  is\_Python = True  # Lists  # initialisation  names = ["Charlie", "Brown", "Chris", 59, True]  # Appending  names.append("Martin")  # Tuples  # Can't be changed after initialisation  names = ("This", "is", "final", true)  # Dictionaries  # initialisation  dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}  # Update  dict.update({  'Gender': 'Female',  'Hobby': 'Reading',  }) |

##### Basic Operators[[6]](#endnote-6)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Addition | a **+** b = 55 |
| - | Subtraction | a – b = 12 |
| \* | Multiplication | a \* b = 556 |
| / | Division | a / b = 45 |
| % | Modulus | b % a = 0 |
| \*\* | Exponent | a\*\*b =10 to the power 20 |
| // | Floor Division | 9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4 |
| == | equivalence | (a == b) is not true. |
| != | Not equal | (a != b) is true. |
| > | Greater than | (a > b) is not true. |
| < | Less than | (a < b) is true. |
| >= | Greater than equal to | (a >= b) is not true. |
| <= | Less than equal to | (a <= b) is true. |

##### Control Flow

Python has support for conditionals, for and while loops, and list comprehension. An example of list comprehension is mentioned below:

|  |
| --- |
| # List comprehension  div\_by\_two = {x for x in range(10) if (x%2 == 0)} |

##### Classes and Functions[[7]](#endnote-7)

Classes and Functions makes up for an important part of any object-oriented programming language. Classes can inherit from other classes and ultimately from “object” class which is the top-level class in Python from which all classes inherit. Functions can accept argument or not. In Python, a function can be defined as below:

|  |
| --- |
| # Funtion that excepts an argument  class Any\_name(object):  def foo(i):  """Function documentation"""  remainder = i % 5  if (remainder!= 0):  return remainder |

### Why Python?

Python is undoubtedly an easy to use programming language with a great community that generously promotes and supports it. Because of the reliability that frameworks like Django, Flasks and Pylons etc. provide, they are being used as a primary platform for development for many software products. Python has no interfaces or real scoping of functions and methods, which lets developer concentrate more of the logic of the application than the syntax of the code itself. Due to all the above reasons, Python proved to be the right choice for developing *Testrek*.

## External Python libraries

Python allows adding external modules (libraries) to a project. Packages are essentially a collection of dynamically written classes with variables and function, which can be re-used in another project. The keyword “import” is used to create a reference to these modules in a project and then the functions from these classes can be used.

|  |
| --- |
| # importing the whole datetime module and creating an alias dt  import datetime as dt  # importing only the classes timedelta and date from the datetime  from datetime import timedelta, date |

An exhaustive list of all the packages publicly available for Python is available on PyPI[[8]](#endnote-8).

Apart from the several packages that come by default with Python, a few other were also used while developing *Testrek*. I brief introduction to each one of them is mentioned in this section.

##### Requests (v. 2.14.2)

Requests is a HTTP library for Python, which send HTTP/1.1 requests without the need for much of work that is required to be taken care of while sending a HTTP request. For example, there is no need to manually add query strings the URLs or to form-encode the POST data.[[9]](#endnote-9) Connection pooling is also taken care of in the library itself which reduces the need to custom write the related code again and again.

***Testrek*** utilizes requests library to download files related to each task from the user’s filesystem (found under public\_html folder) on the University server.

##### Easygui (v. 0.98.1)

EasyGUI is a simple yet robust GUI written in Python. It is not event driven, instead all the GUI interactions are invoked by simple function calls.[[10]](#endnote-10) This GUI library is used to present dialogs with information during the runtime of ***Testrek***. It saves the user from knowing anything about tkinter, frames, widgets, callbacks or lambda, which are core to it. It runs smoothly on Python 2 and 3 and does not have any dependencies.

##### Fuzzywuzzy (v. 0.15.0)[[11]](#endnote-11)

Fuzzywuzzy is a package used for the string comparison. It uses Levenshtein Distance[[12]](#endnote-12) to calculate the differences between the sequence of strings. It is compatible with python 2.4 or higher. It utilizes difflib library that comes bundled with Python and uses the package python-Levenshtein to deliver results even faster. In ***Testrek*** python-Levenshtein package is used along with fuzzywuzzy in order to get results as quickly as possible.

##### Tqdm (v. 4.14.0)[[13]](#endnote-13)

Tqdm package is available for Python 2.6 and higher is used to create a progress bar from the number of iterations. For implementation, it must simply wrap with the iterable. The snippet below one can see the implementation in a real-time scenario.

|  |
| --- |
| from tqdm import tqdm  for i in tqdm(range(500)):  ... |

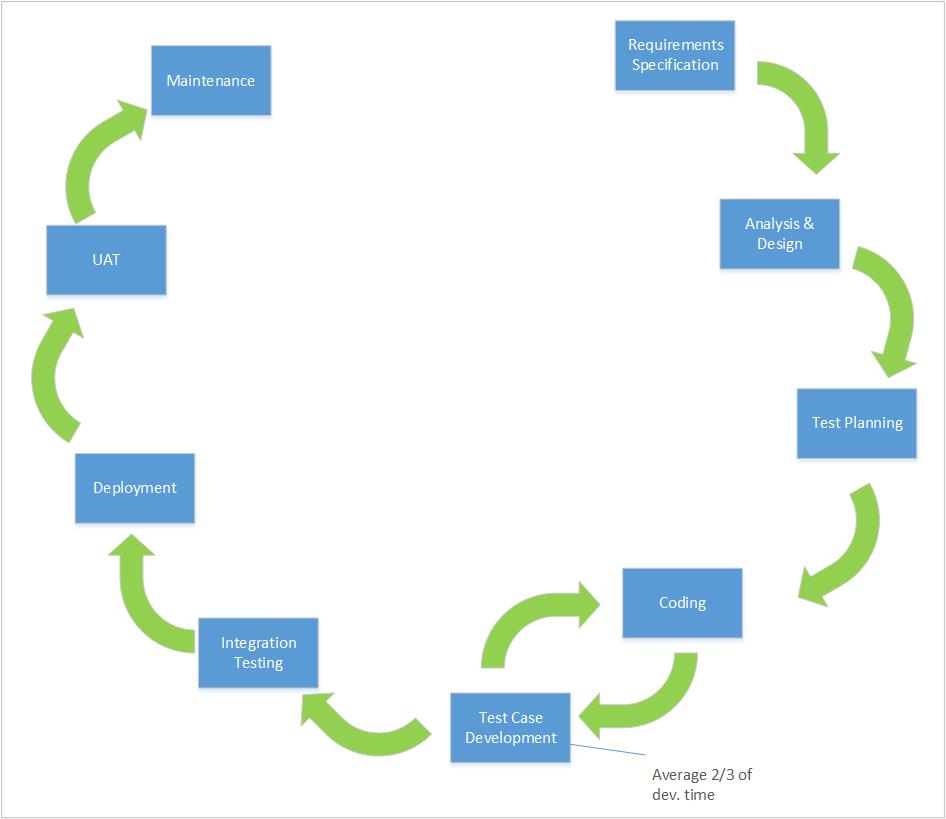
##### Beautifulsoup4 (v. 4.6.0)[[14]](#endnote-14)

Beautifulsoup is a an extensively used python package for parsing through a web page. Since its emergence in 2004[[15]](#endnote-15), it has been under constant development and the latest version provides some great features and robust runtime performance. It is built upon am HTML or XML parser, providing extensive features to iterate, search and modify the parsing tree.

In ***Testrek***, it is mainly used to check for the existence of the files on the web url before *requests* library can be used to download that file. This was necessary to be done while providing the right results in the download success report.

## Development and Testing Methodology

Most of the development work on ***Testrek***, was conducted by me and it was necessary to choose the right development techniques. The list of possible development methodologies[[16]](#endnote-16) was already shorten down to only a few because of this fact. The timeframe available for the development of the application was limited as well and it served as the second condition to limit the list down to one. For all these reasons, test-driven development was chosen. All through the development life cycle, it was necessary to design tests and then write the function definitions. Test driven development[[17]](#endnote-17) has its own challenges and but it did fit best for a small project like *Testrek.* The illustration below provides an overview of how the development was approached at different stages



The following sequence of steps are generally followed:

* Add a test
* Run all tests and see if the new one fails
* Write some code
* Run tests
* Refactor code
* Repeat

It is true that TDD slows down the development but once we get into the loop it becomes quite easy. It was important to produce better designs, allow easy and safe refactoring and slowly increase the test coverage. Another great benefit of TDD is that tests serve as a living documentation.

## Plagiarism Check

*Testrek* incorporates a feature to run plagiarism check on answers from every test taker against the answers from every other test taker. For this purpose, *fuzzywuzzy* package is used which can provide similarity ratios processed for different types of string comparisons.[[18]](#endnote-18)

##### Simple Ratio

|  |
| --- |
| >>> from fuzzywuzzy import fuzz  >>> from fuzzywuzzy import process  >>> fuzz.ratio("we are here, finally", "we are here, finally!")  97 |

##### Partial Ratio

|  |
| --- |
| >>> fuzz.partial\_ratio("this is a test", "this is a test!")  100 |

##### Token Sort Ratio

|  |
| --- |
| >>> fuzz.ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")  91  >>> fuzz.token\_sort\_ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")  100 |

##### Token Set Ratio

|  |
| --- |
| >>> from fuzzywuzzy import fuzz  >>> from fuzzywuzzy import process  >>> fuzz.token\_sort\_ratio("fuzzy was a bear", "fuzzy fuzzy was a bear")  84  >>> fuzz.token\_set\_ratio("fuzzy was a bear", "fuzzy fuzzy was a bear")  100 |

Fuzzywuzzy utilizes the ***Levenshtein Distance[[19]](#endnote-19)***to compute these ratios. Levenshtein distance (LD) is a measure of the similarity between two input strings. The distance is the number of deletions, insertions, or substitutions required to transform one stringinto another. The greater the Levenshtein distance, the more different the strings are.

Let’s take two identical strings. If x is "test" and y is "test", then LD(s,t) = 0, as no transformations are needed.

If s is "rent" and t is "rant", then LD(s,t) = 1, because one substitution (change "s" to "n") is sufficient to transform s into t.

Levenshtein distance are used in the following fields:

* Spell checking
* Speech recognition
* DNA analysis
* Plagiarism detection

# Technical walkthrough

This chapter describes the working of ***Testrek*** application, different standardized inputs that it takes and the output files it generates. As described in chapter 2, *Testrek* would need standardization certain parameters that it takes before starting the main process.

The file\_download function defined below is used to download the files from the server. The requests.get method returns the response from the URL that is provided as a parameter. The function proceeds if the URL can be reached by checking the status code of the response method. Below are some status codes that can be returned by a server:

|  |  |
| --- | --- |
| Code | Description |
| 100 | Continue |
| 200 | OK |
| 201 | Created |
| 202 | Accepted |
| 400 | Bad Request |
| 404 | Not Found |
| 500 | Internal Server Error |

|  |
| --- |
| def file\_download(file\_url):  # type: (object) -> object  """Downloads file from the server  :param file\_url: url to the file to be downloaded  :return: Either the downloaded file or False  """  local\_filename = file\_url.split('/')[-1]  r = requests.get(file\_url, stream=True)  if r.status\_code == 200:  with open(local\_filename, 'wb') as f:  for chunk in r.iter\_content(chunk\_size=1024):  if chunk:  f.write(chunk)  f.flush()  return local\_filename  pass  return 0 |

Appendix A.   
  
An Example of an Appendix

Appendices should be used for supplemental information that does not form part of the main research. Remember that figures and tables in appendices should not be listed in the List of Figures or List of Tables.

References

1. “[PEP 8 -- Style Guide for Python Code.](https://www.python.org/dev/peps/pep-0008/)” [↑](#endnote-ref-1)
2. “[ČVUT v Praze, Fakulta Strojní: Log in to the Site.](https://moodle.fs.cvut.cz/login/index.php)” [↑](#endnote-ref-2)
3. “[PyCharm](https://www.jetbrains.com/pycharm/).” [↑](#endnote-ref-3)
4. Nowell Strite, “[Introduction to Python.](https://www.slideshare.net/nowells/introduction-to-python-5182313)” [↑](#endnote-ref-4)
5. t[utorialspoint.com, “Python Dictionary.](https://www.tutorialspoint.com/python/python_dictionary.htm)” [↑](#endnote-ref-5)
6. [tutorialspoint.com, “Python Basic Operators.](https://www.tutorialspoint.com/python/python_basic_operators.htm)” [↑](#endnote-ref-6)
7. “[Learn Python the Hard Way.](https://learnpythonthehardway.org/book/ex40.html)” [↑](#endnote-ref-7)
8. “PyPI - the Python Package Index: [Python Package Index.](https://pypi.python.org/pypi)” [↑](#endnote-ref-8)
9. “Requests: HTTP for Humans — [Requests 2.17.3 Documentation.](http://docs.python-requests.org/en/master/)” [↑](#endnote-ref-9)
10. “[Easygui](https://github.com/robertlugg/easygui).” [↑](#endnote-ref-10)
11. “[Seatgeek/Fuzzywuzzy.](https://github.com/seatgeek/fuzzywuzzy)” [↑](#endnote-ref-11)
12. “[Levenshtein Distance.](https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm)” [↑](#endnote-ref-12)
13. “[developers, *Tqdm*](https://pypi.python.org/pypi/tqdm).” [↑](#endnote-ref-13)
14. [“Richardson, *beautifulsoup4*.](https://pypi.python.org/pypi/beautifulsoup4)” [↑](#endnote-ref-14)
15. “[Beautiful Soup](https://www.crummy.com/software/BeautifulSoup/): We Called Him Tortoise Because He Taught Us.” [↑](#endnote-ref-15)
16. “[Software Development Methodologies.](http://www.itinfo.am/eng/software-development-methodologies/)” [↑](#endnote-ref-16)
17. Farcic, “[Test Driven Development (TDD).](https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/)” [↑](#endnote-ref-17)
18. Marco, “[Fuzzy String Matching in Python.](https://marcobonzanini.com/2015/02/25/fuzzy-string-matching-in-python/)” [↑](#endnote-ref-18)
19. “[Levenshtein Distance.](https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm)” [↑](#endnote-ref-19)