PyAutoMark

An Application for Computer Science Teachers

Daniel M. Palmarin[*]

Contents

[*]E-mail: dan.palmarin@gmail.com

1 SOFTWARE INSTALLATION

1. Download the `PyAutoMark_Installer.exe` at:
   https://github.com/DanPalmarin/PyAutoMark

   Alternatively, you can download the entire repository as a zip file and run the application as a `.py` (or you could run the installer). **Note that this application is open-source**. If you make any significant changes, please initiate a pull request so that those changes can potentially be incorporated into the main repository.

2. Launch `PyAutoMark_Installer.exe`.

3. Complete the installation process by following the installation prompts. Note that the default installation directory is `C:\Program Files\PyAutoMark\`. You can change this if you wish.

2 USING THE APPLICATION

2.1 Intended use

This program is intended to be used to mark computer science coding assignments, where the API is Python 3. The following assumptions are important to keep in mind before using the program:

- Each assignment is assumed to contain multiple questions.
- Each question must correspond to a python script that, when executed, produces an output. This is typically done using the `print` function.
- Questions can accept user input using the `input` function, but this is not required.

2.2 Functionality

First, it is best to conceptualize assignments as folders (directories) in Windows File Explorer, Each assignment question corresponds to a python script located inside the assignment folder. Before sending the assignment for marking, each student compresses the assignment folder as a `zip` file. Starting with many such compressed `zip` folders, the functionality of PyAutoMark is outlined below:

- The user is prompted to specify the location to an assignment key (written as a python script).
- The user is prompted to select any number of student `zip` files (student assignment submissions).
- The user runs the program. PyAutoMark then performs the following actions on each `zip` file:
  - Unzip the `zip` file (student submission).
  - Execute each python script contained in the student submissions and compare output to the solution key. It is possible to open certain scripts multiple times to check various inputs.
  - A text file is produced with a summary of the results.

2.3 Example

Consider the following basic introduction to python assignment, which consists of the following three questions:

1. **Problem:** Assign the word "Hello" to a variable. Then print this variable to the console.

   Example output:
   ```
   Hello
   ```

2. **Problem:** Make a program that reads in a value of time (in hours) and prints that value in minutes and seconds. The program should be able to accept a decimal (float) for input.

   Example input:
   ```
   1.0
   ```

   Example output:
   ```
   60.0
   3600.0
   ```

3. **Problem:** Given a quadratic equation: $ax^2 + bx + c = 0$, the *discriminant* is given by $b^2 - 4ac$. Create a program that accepts the variables a, b, and c as inputs. Then compute the discriminant as a float and print it to the console.

   Example input:
   ```
   1
   4
   2.0
   ```

   Example output:
   ```
   8.0
   ```

The solution key should be saved as a `.py` file. Here is one possible solution key for the above questions:

```
assignment_num = 1

Q1 = "Hello"
Q2 = {
    1.0 : "60.0\n3600.0",
    5 : "300.0\n18000.0"
    }
Q3 = {
    (1,4,2.0) : 8.0,
    (2.0,3.0,1.0) : 1.0
    }

Q_weight = [1,2,2]
Q_all = [Q1,Q2,Q3]
```

The above format **must be followed** for PyAutoMark to function properly. That is, the variables `assignment_num`, `Q_weight`, and `Q_all` must be defined. If the question only has an output (no inputs), follow the syntax of `Q1`. Note that multiple outputs are indicated by separating each with `\n` (see the output of `Q2`). If a question requires both an input and an output, this is defined using a python dictionary. A question that requires only one user input should follow the syntax of `Q2`. A question that requires multiple user inputs should follow the syntax of `Q3`. The multiple inputs are stored as a tuple.

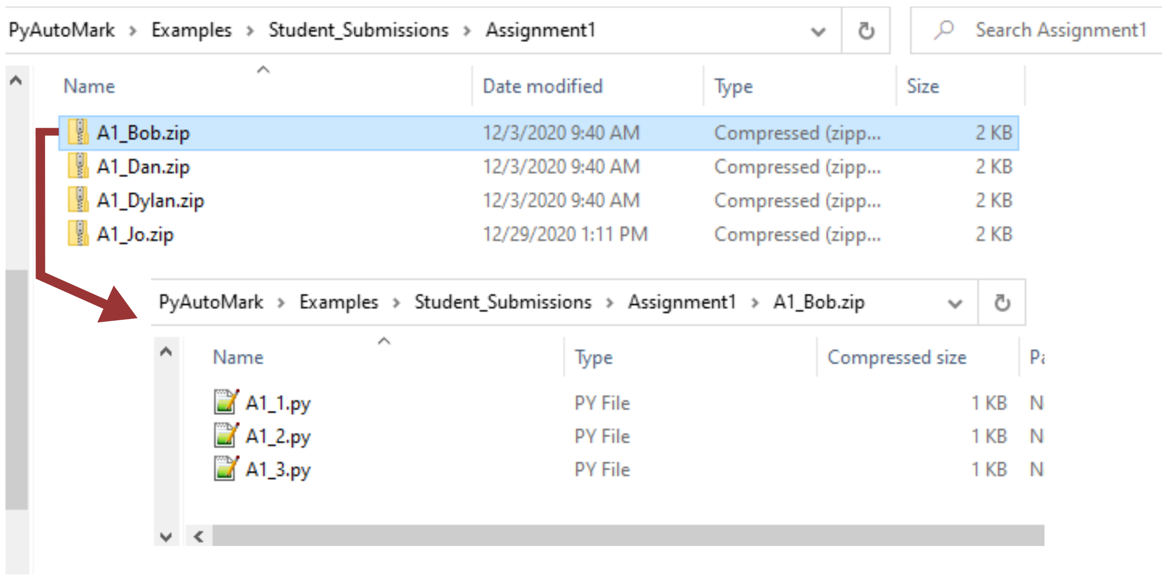
In the solution key above, `Q2` and `Q3` instructs PyAutoMark to run those scripts twice as they both define two sets of inputs and outputs. There is no limit to the number of times a script is run. This is useful for testing a student's code against many different inputs/outputs to ensure that the code is functioning properly.

Finally, the solution above has the weight of each assignment question defined in the variable `Q_weight`. Here, `Q1` is out of 1 mark and both `Q2` and `Q3` are out of 2 marks. Hence, this assignment is out of 5 marks. Note that there is no partial mark functionality (a student cannot score 1 out of 2 on `Q2` for example).

In order to stay organized and precise, PyAutoMark requires the variable `assignment_num` to be defined as a positive integer. **Students must adhere to the following format when naming zip and python files**:

Assignment folder name: `A(num)_(name)`
Assignment question name: `A(num)_(question num)`

In the example above, `num=assignment_num=1`. See the following figure for more clarity.



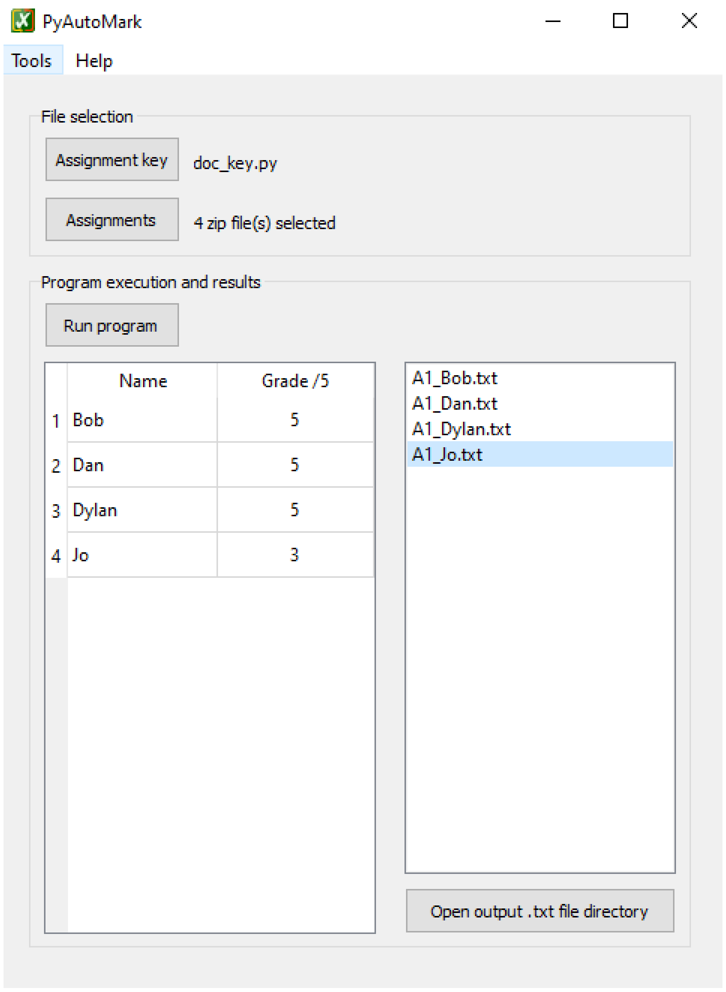
PyAutoMark › Examples › Student_Submissions › Assignment1

Name	Date modified	Type	Size
A1_Bob.zip	12/3/2020 9:40 AM	Compressed (zipp...	2 KB
A1_Dan.zip	12/3/2020 9:40 AM	Compressed (zipp...	2 KB
A1_Dylan.zip	12/3/2020 9:40 AM	Compressed (zipp...	2 KB
A1_Jo.zip	12/29/2020 1:11 PM	Compressed (zipp...	2 KB

PyAutoMark › Examples › Student_Submissions › Assignment1 › A1_Bob.zip

Name	Type	Compressed size	Pa
A1_1.py	PY File	1 KB	N
A1_2.py	PY File	1 KB	N
A1_3.py	PY File	1 KB	N

Note that the question and folder names can have additional words or numbers. For both, any additions should be placed *in front* of `A(num)_(name)` or `A(num)_(question num)`. Selecting the four students above in PyAutoMark, the result should look as follows:



doc_key.py

4 zip file(s) selected

	Name	Grade /5
1	Bob	5
2	Dan	5
3	Dylan	5
4	Jo	3

A1_Bob.txt
A1_Dan.txt
A1_Dylan.txt
A1_Jo.txt

Here, Jo scored 3/5. On question 2, he incorrectly converted hours into seconds, as seen in his output text file, `A1_Jo.txt`:

```
1. Correct

2. Incorrect
    -Test 1: failure
       Input: 1.0
       Desired output: 60.0 3600.0
       Your output: 60.0 360.0

    -Test 2: failure
       Input: 5
       Desired output: 300.0 18000.0
       Your output: 300.0 1800.0

3. Correct
    -Test 1: success
    -Test 2: success
```