

# Constraint-Based Python Tutor:

## A Platform for Instructors

Bennett DenBleyker

December 08, 2023

Seth Poulsen's CS5890

### **Abstract**

The tool proposed in this essay aims to provide a platform for instructors to teach Python skills to their students in a standard manner. Using Django for the web interface and RestrictedPython for a sandboxed environment, the platform includes features such as multiple tabs for coding, saving code locally, running code, requesting hints, and testing code against instructor-written constraints. Preliminary testing has used an implementation of a FizzBuzz problem to demonstrate its utility in guiding users with basic proficiency towards the correct answer.

### **Problem**

Programming languages are difficult to teach and evaluate. For those new to programming, understanding what exactly the instructor wants is often a huge stumbling block. When friends ask me for help on a coding project, the issue is usually a fundamental misunderstanding of either a programming construct (such as functions or classes) or what the instructor wants and expects.

### **Background**

Most tools online for learning Python teach in a sequential manner, which isn't very useful as a classroom aid, nor does it require the learner to branch out. They are useful for learning the basics of Python, but not how to solve more complex problems. These tools include Codecademy, [learnpython.org](https://learnpython.org), and others.

Another tutor for Python found online is Python Tutor. This acts as a visual debugger for Python code. It also allows the user to ask ChatGPT questions, which will answer in context of the code, with the option to “guide me to come up with answers on my own.”

## Approach

My approach to solving this problem was to create a platform: a platform which requires the instructor to give the problem in the format of

1. In a single Python file, `fizzbuzz.py`:
  1. Write a function `fizzbuzz(n: int) -> str` that:
    - Returns "Fizz" for any multiple of 3
    - Returns "Buzz" for any multiple of 5
    - Returns "FizzBuzz" for any multiple of both
    - Returns the number as a string for any multiple of neither
  2. When run:
    - Prints the output of `fizzbuzz(n)` for `n` in (1, 100), separated by newline characters
  3. When imported:
    - Prints nothing
    - Exposes only the `fizzbuzz(n)` function

Figure 1: An example of how Design Requirements are displayed to the learner

design requirements, and gives the user most of the tools of a normal IDE. On top of this, the instructor is given a unified user interface for creating their problems, and for writing tests for them.

The tool uses Django to host the web interface, and RestrictedPython to run the code within a sandboxed environment.

When the learner opens a problem, it opens with the starter code set by the instructor. The

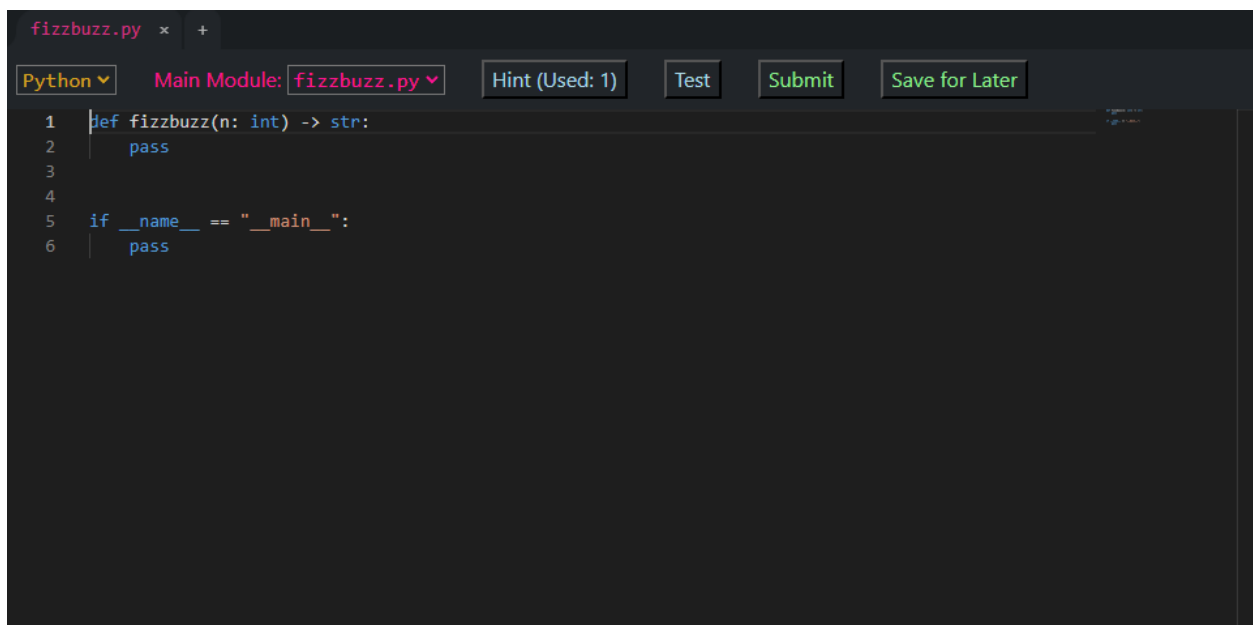
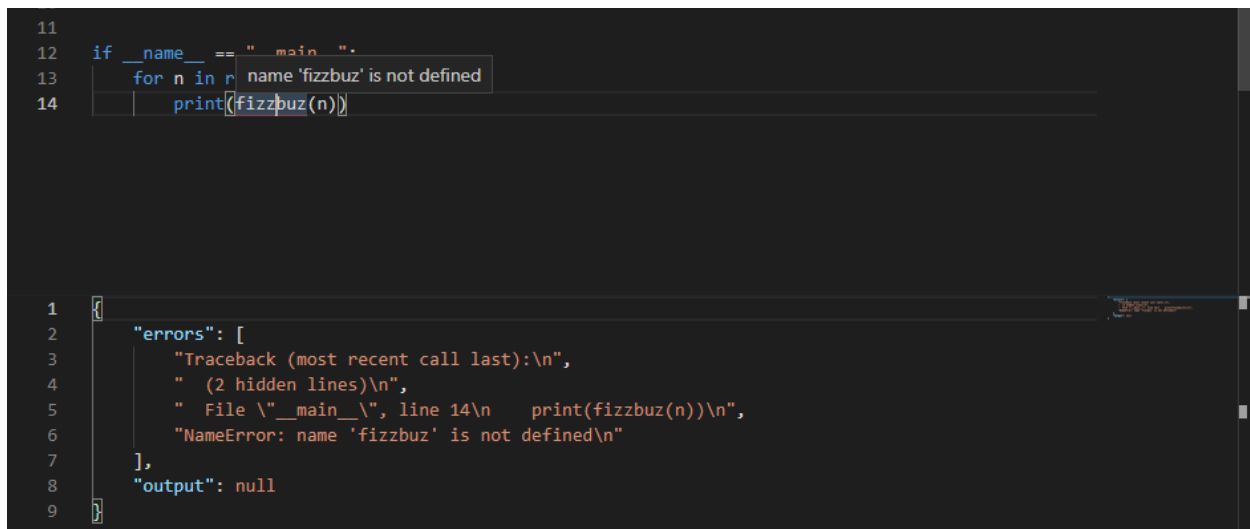


Figure 2: User interface at first load

instructor has the option to create one or multiple tabs, and allow the user to create more, if they like. Each file can be simply imported by other files as if they were in the same folder in a normal Python package.

As the learner writes code, they can take one of four actions at any time. The first is to save their code; the second is to run the code and see the output; the third is to request a hint; and the final action is to submit the code for grading. When saved, the code is saved to the user's local storage, meaning that it is accessible on the same browser in future sessions, so long as the user does not clear their cache. In each instance, the instructor has a chance to review the code with constraint-based tests, and give the learner feedback. This feedback can be in the form of a red underline (or other in-text formatting), a hover message, or Markdown at the bottom.



```
11
12 if __name__ == "__main__":
13     for n in range(1, 101):
14         print(fizzbuzz(n))

{
  "errors": [
    "Traceback (most recent call last):\n",
    "  (2 hidden lines)\n",
    "    File \"__main__\", line 14, in <module>: print(fizzbuzz(n))\n",
    "NameError: name 'fizzbuzz' is not defined\n"
  ],
  "output": null
}
```

Figure 3: User interface on error

## Write a function fizzbuzz(n: int) -> str

Your code doesn't return a value

```
{
  "value_received": null,
  "value_sent": 122
}
```

Figure 4: Markdown output on error

## Results

As it stands, it implements a rudimentary structure for evaluating a student's FizzBuzz program. It does not yet give enough hints to help a user who knows nothing about Python, but to a user with basic proficiency, it is enough to guide them to the correct answer. Once implemented, though, many of these tests can be easily copied and adapted to different problems/use cases. The main advantage of this system is that it was clear to every single 'test subject' what was expected – even those who had little to no programming knowledge. Though they struggled with a few of the underlying concepts (though this could be remedied with more comprehensive tests and hints), the hinting, testing, and error-underlining helped to ensure that the output looked and worked exactly as intended.

## **Next Steps**

There are many different areas of possible improvement for this tool. The first has to do with ease-of-use: saving the language of tabs (and allowing for the rendering of Markdown files), and automatically setting it depending on the file extension; saving the file and running tests when keystrokes haven't occurred in a while; saving code into a database (with sign-in), not just to local storage; and adding a custom autocomplete handler. These would increase the usability of the tool, and thereby the comfort of the learner and the utility to the instructor.

The second group of improvements would have to do with better explanation of the design requirements: leveraging ChatGPT to help answer the learner's questions of design requirements; including pre-baked explanations of new knowledge components used in the problem, and allowing the same question-answer system to help with these knowledge components. The user could even be allowed to search StackOverflow, and may be limited to certain articles, at the instructor's discretion.

Finally, auto-grading still needs to be implemented, and problem states would then be saved into the database.