THE UNIVERSITY of EDINBURGH

Find out more about integrating Noteable into your courses:
noteable.edina.ac.uk
information-services.ed.ac.uk/

Teaching and learning tools for code as course-integrated SaaS.
Featuring extensions and plugins

Noteable™ by EDINA

a cloud-hosted digital platform developed in ISG that revolutionises assessment and feedback practices in higher education.

Designed to integrate with academic curricula and virtual learning environments, Noteable offers educators robust tools for creating equitable, accessible coding assignments. The focus on digital inclusivity supports a diverse range of student needs, fostering fair environments for code assignments in courses.

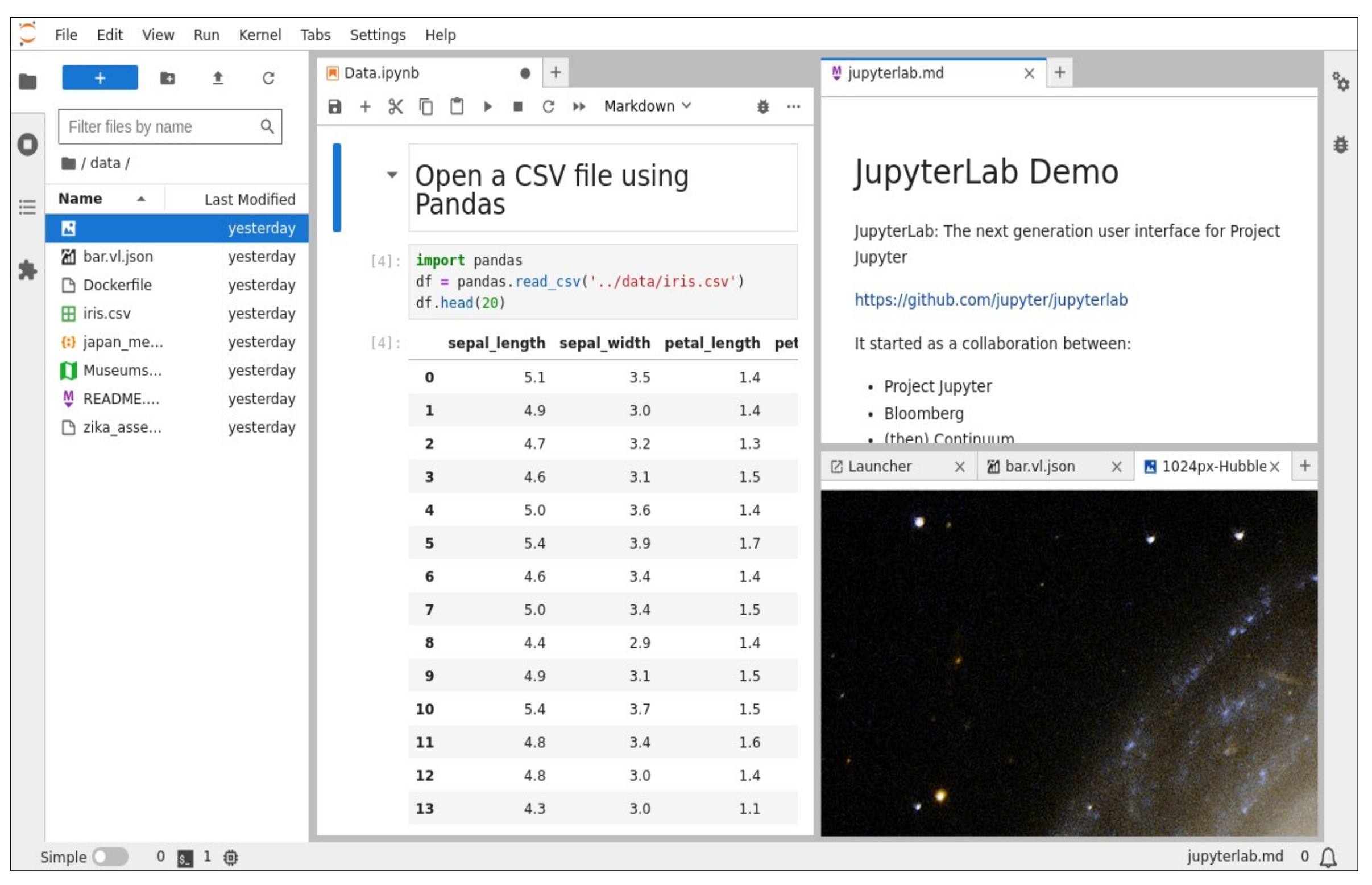
By integrating Noteable, educators enhance student engagement and teaching efficiency through with auto-grading and real-world applications. Discover how Noteable can elevate your educational methods.

Why Noteable?

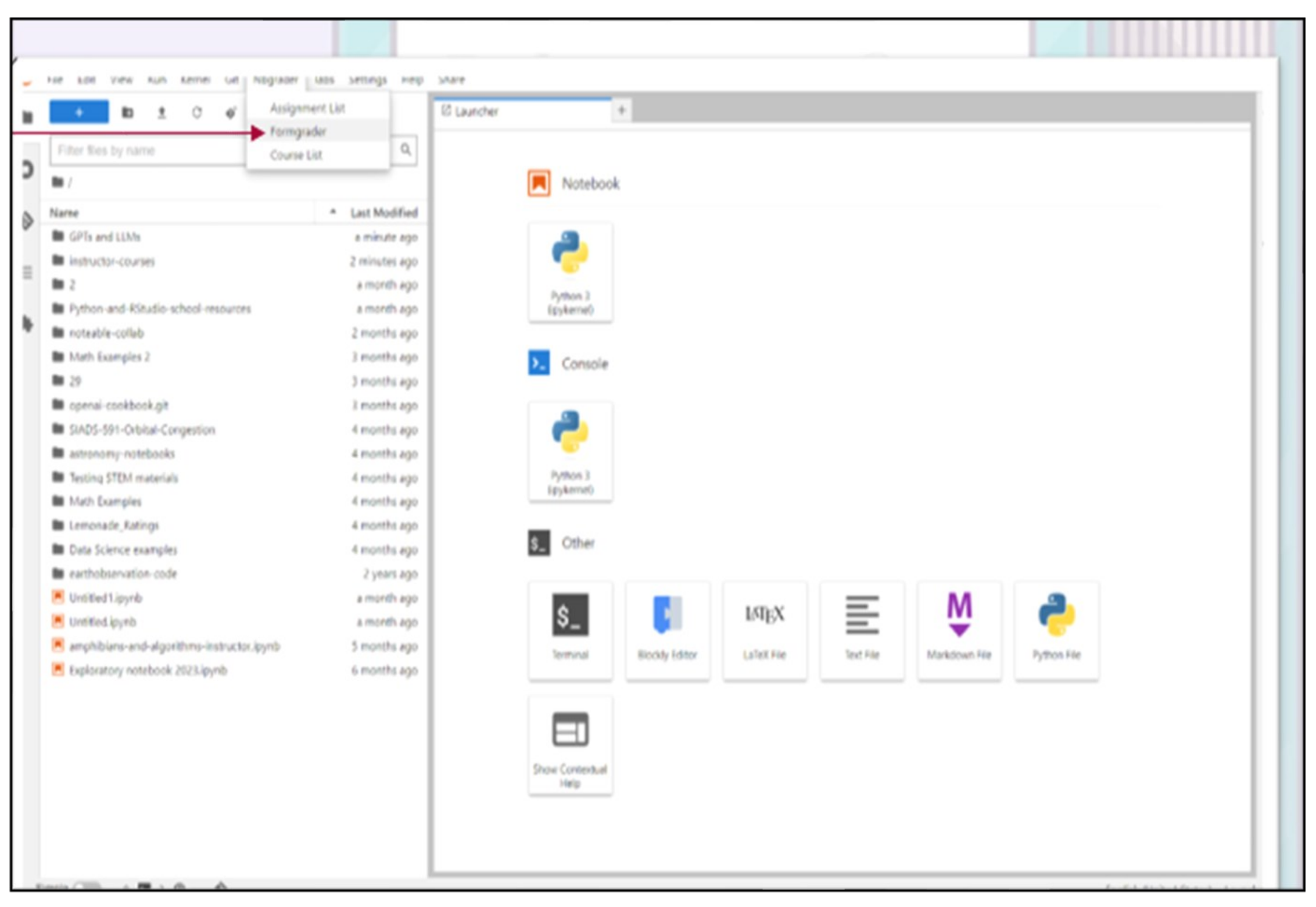
Aligning with Teaching and Learning Priorities

•**Inclusive Assessment and Feedback**: Noteable's tools cater to diverse student abilities and backgrounds, promoting equity in education.

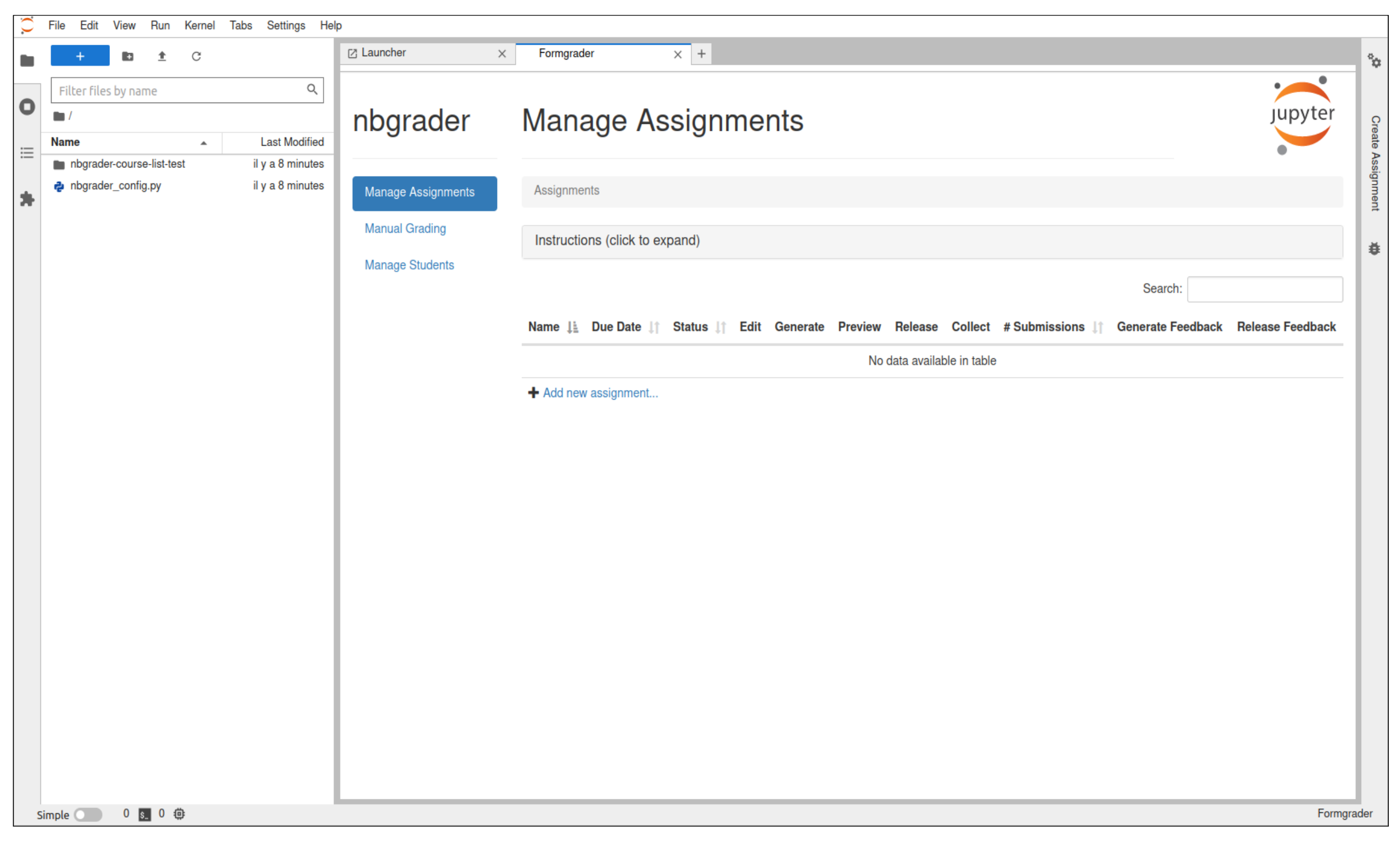
•**Innovation and Technological Opportunities**: Leverages cutting-edge technology to transform traditional assessment models, encouraging authentic learning experiences.



based on open-source Jupyter notebooks ecosystem



Code grading Features



manage course-level assignments and data in one place

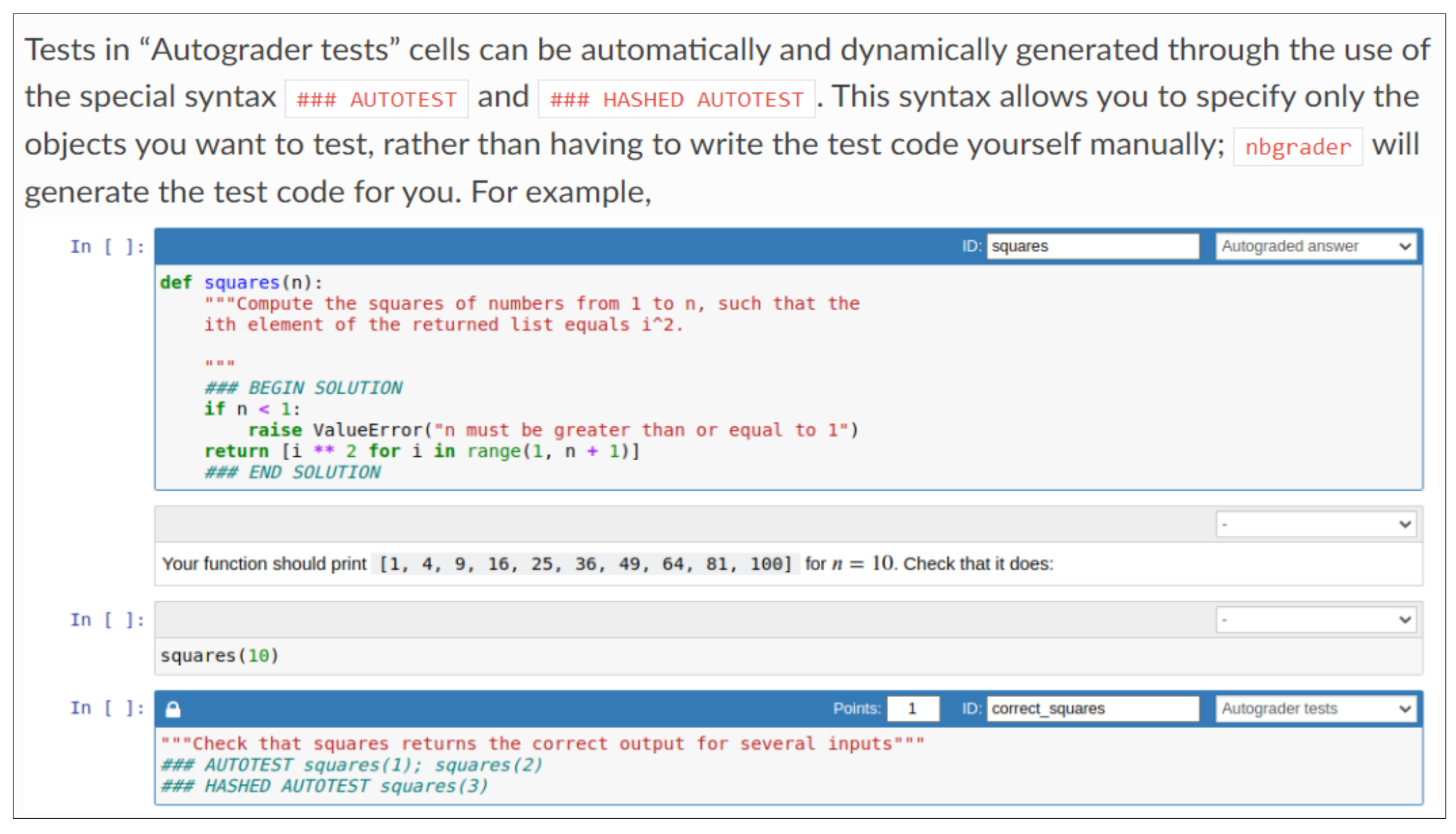
Efficiency: Automates the grading process to provide instant feedback, allowing educators to focus on qualitative assessment.

Consistency: Ensures uniform evaluation criteria across all coding assignments, enhancing fairness.

Scalability: Accommodates large classes without compromising the quality of feedback or the assessment process.

Engage, Learn, Innovate:

Discover how Noteable can elevate your coding assessments. Join thousands of users across institutions in pushing the boundaries of traditional education through innovative, inclusive, and technologically advanced methodologies. One integration for multiple instructors to benefit from code (auto)grading in courses.



Tests in "Autograder tests" cells can be automatically and dynamically generated through the use of the special syntax `### AUTOTEST` and `### HASHED AUTOTEST`. This syntax allows you to specify only the objects you want to test, rather than having to write the test code yourself manually; `nbgrader` will generate the test code for you. For example,

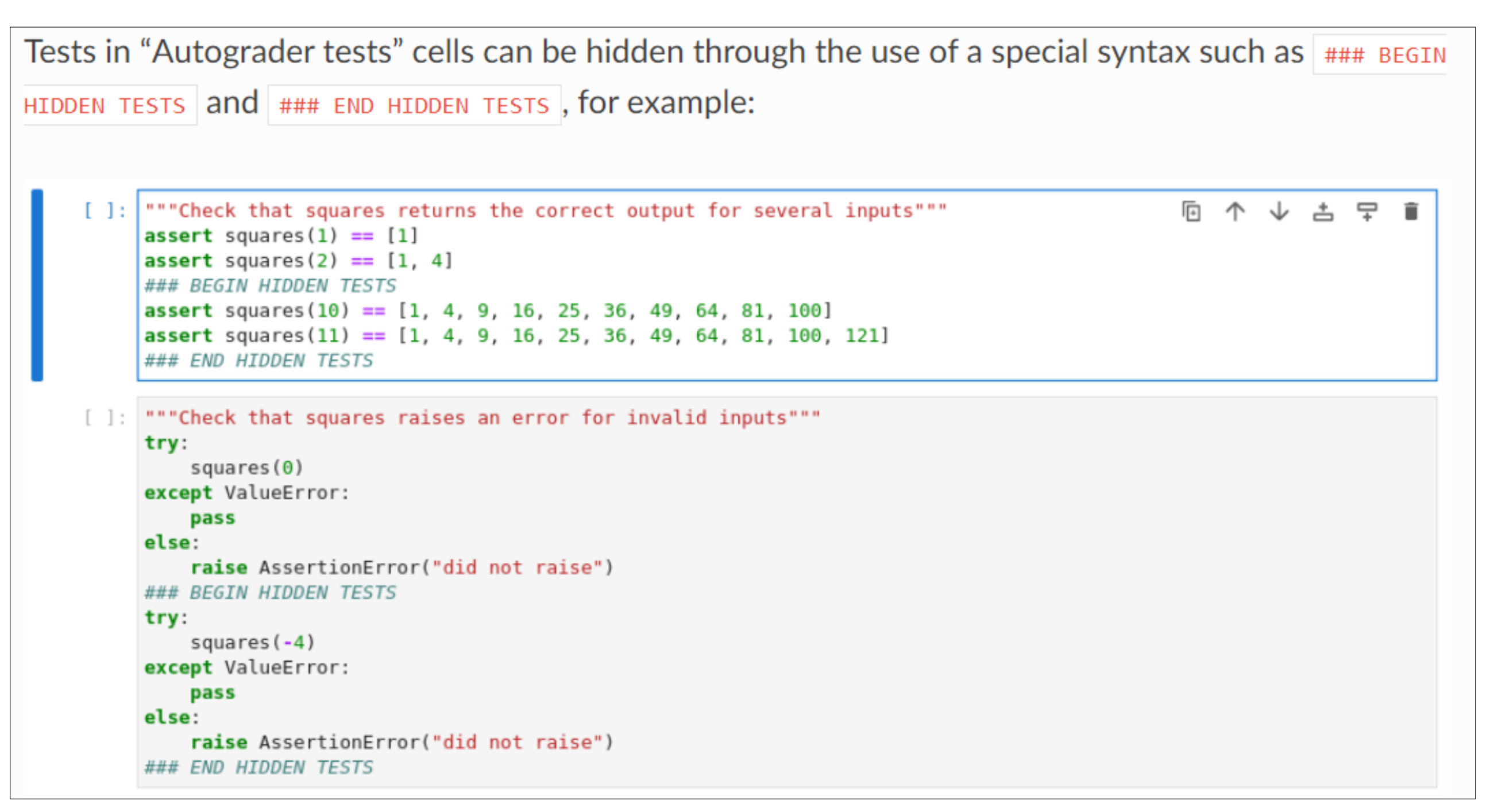
```
In [ ]:                                    ID: squares   Autograded answer
def squares(n):
    """Compute the squares of numbers from 1 to n, such that the
    ith element of the returned list equals i^2.

    """
    ### BEGIN SOLUTION
    if n < 1:
        raise ValueError("n must be greater than or equal to 1")
    return [i ** 2 for i in range(1, n + 1)]
    ### END SOLUTION
```

Your function should print `[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]` for $n = 10$. Check that it does:

```
In [ ]:
squares(10)
```

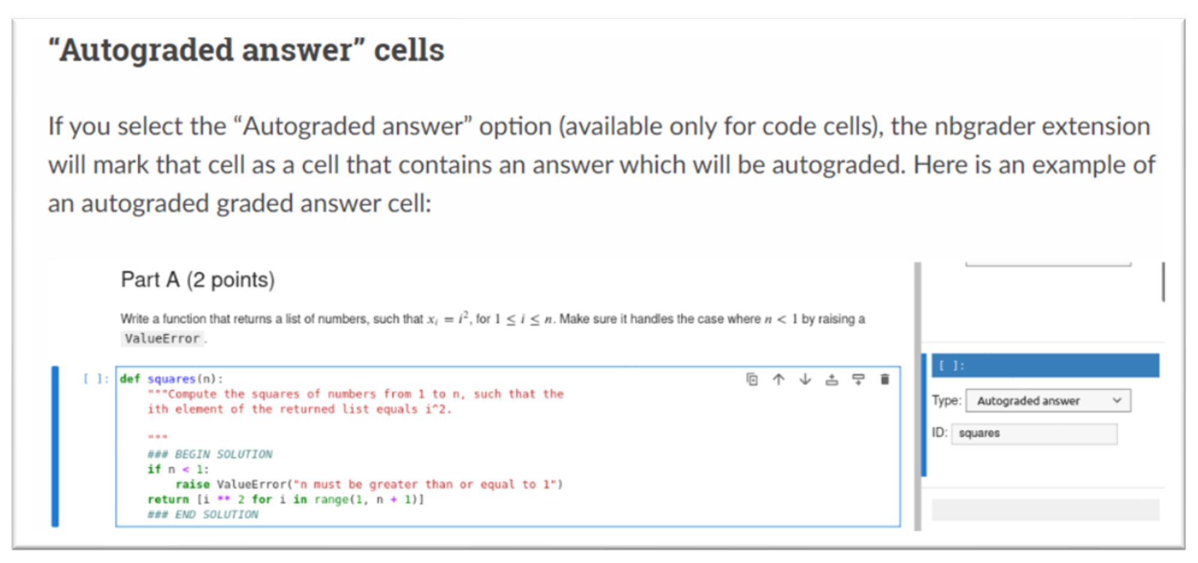
```
In [ ]:                  Points: 1   ID: correct_squares   Autograder tests
"""Check that squares returns the correct output for several inputs"""
### AUTOTEST squares(1); squares(2)
### HASHED AUTOTEST squares(3)
```



Tests in "Autograder tests" cells can be hidden through the use of a special syntax such as `### BEGIN HIDDEN TESTS` and `### END HIDDEN TESTS`, for example:

```
[ ]:
"""Check that squares returns the correct output for several inputs"""
assert squares(1) == [1]
assert squares(2) == [1, 4]
### BEGIN HIDDEN TESTS
assert squares(10) == [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
assert squares(11) == [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
### END HIDDEN TESTS
```

```
[ ]:
"""Check that squares raises an error for invalid inputs"""
try:
    squares(0)
except ValueError:
    pass
else:
    raise AssertionError("did not raise")
### BEGIN HIDDEN TESTS
try:
    squares(-4)
except ValueError:
    pass
else:
    raise AssertionError("did not raise")
### END HIDDEN TESTS
```



"Autograded answer" cells

If you select the "Autograded answer" option (available only for code cells), the nbgrader extension will mark that cell as a cell that contains an answer which will be autograded. Here is an example of an autograded graded answer cell:

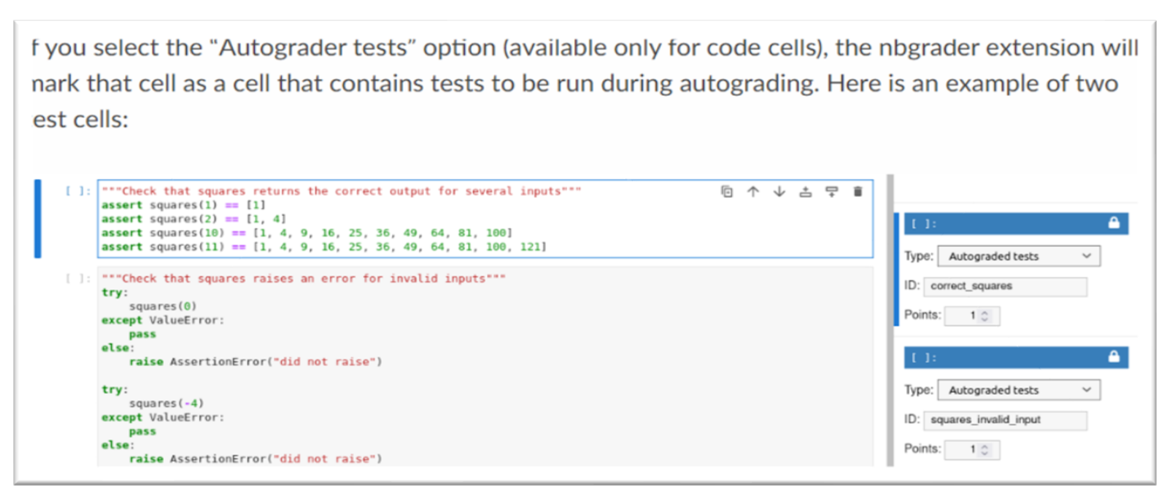
Part A (2 points)

Write a function that returns a list of numbers, such that $x_i = i^2$, for $1 \leq i \leq n$. Make sure it handles the case where $n < 1$ by raising a `ValueError`.

```
[ ]:
def squares(n):
    """Compute the squares of numbers from 1 to n, such that the
    ith element of the returned list equals i^2.

    """
    ### BEGIN SOLUTION
    if n < 1:
        raise ValueError("n must be greater than or equal to 1")
    return [i ** 2 for i in range(1, n + 1)]
    ### END SOLUTION
```

Type: Autograded answer
ID: squares



f you select the "Autograder tests" option (available only for code cells), the nbgrader extension will nark that cell as a cell that contains tests to be run during autograding. Here is an example of two est cells:

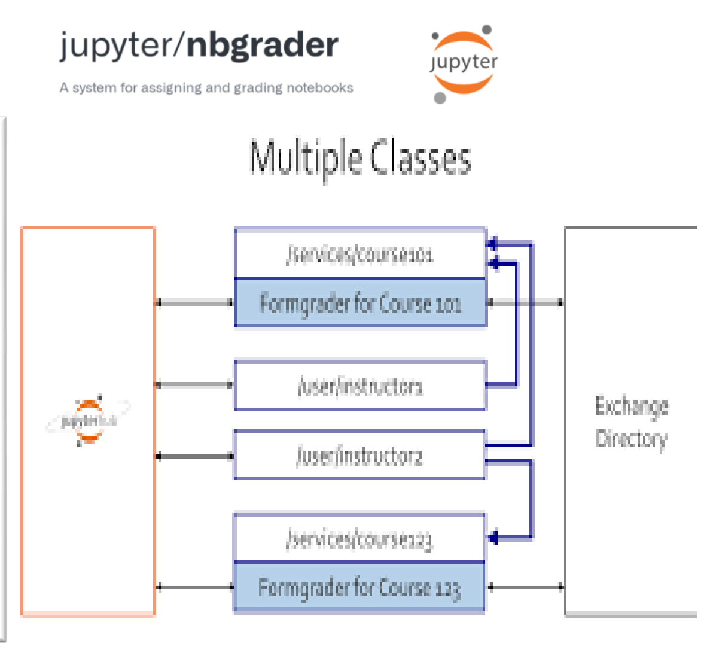
```
[ ]:
"""Check that squares returns the correct output for several inputs"""
assert squares(1) == [1]
assert squares(2) == [1, 4]
assert squares(10) == [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
assert squares(11) == [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
```

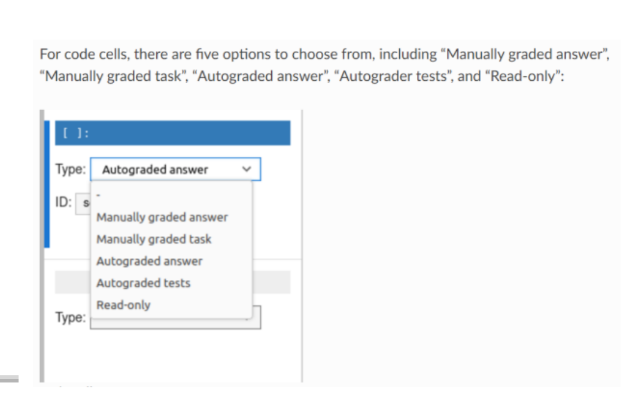
```
[ ]:
"""Check that squares raises an error for invalid inputs"""
try:
    squares(0)
except ValueError:
    pass
else:
    raise AssertionError("did not raise")

try:
    squares(-4)
except ValueError:
    pass
else:
    raise AssertionError("did not raise")
```

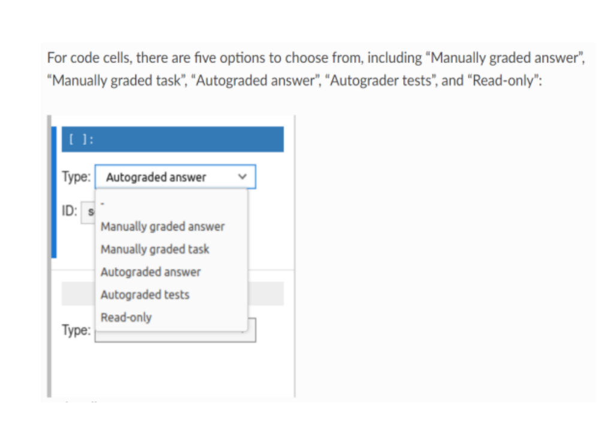
Type: Autograded tests
ID: correct_squares
Points: 1

Type: Autograded tests
ID: squares_invalid_input
Points: 1





For code cells, there are five options to choose from, including "Manually graded answer", "Manually graded task", "Autograded answer", "Autograder tests", and "Read-only":



For code cells, there are five options to choose from, including "Manually graded answer", "Manually graded task", "Autograded answer", "Autograder tests", and "Read-only":