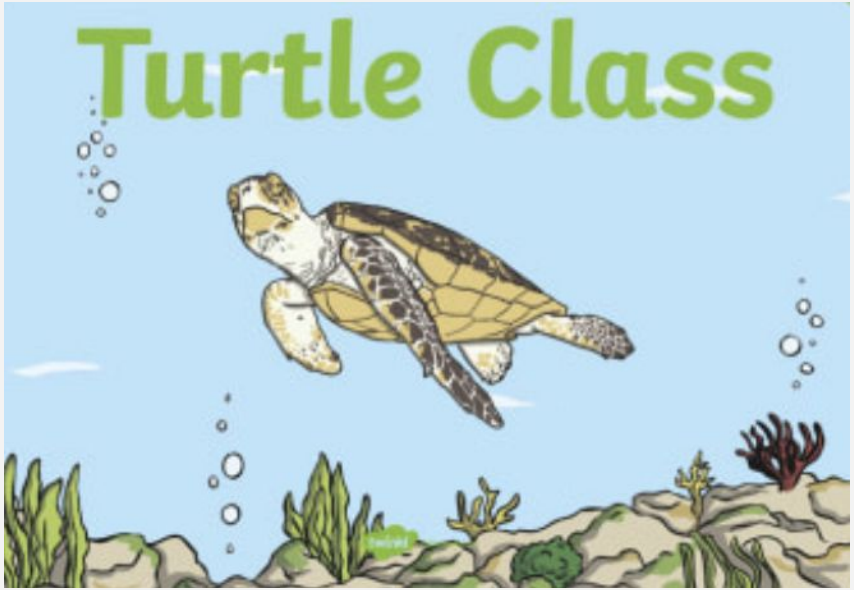
MAKE SCHOOL

Properties & Methods & Testing

(and some turtle graphics)



What we're going to learn

MAKE SCHOOL

- More complex uses of properties and methods
- PEP 8 Python Style Guide
- Basic accessibility principles
- How to create classes and objects from a problem description
- How to use turtle graphics to draw
- How to use turtle graphics and OOP together

Check In

MAKE SCHOOL

Drag the icon to the correct square

Savory, not sweets!

😡😑

Students, drag the icon!

MAKE SCHOOL

Warm Up: Complete the TODO items

Students browse: repl.it/@MakeSchool/creatingclasses?lite=true

MAKE SCHOOL

Let's take a look at a slightly more complex class definition

Students browse: repl.it/@MakeSchool/morecomplexclass?lite=true

PEP8 Style Guide

MAKE SCHOOL

- When writing Python code best practice is to follow the PEP8 standard
- These standards exist to improve **readability**
- What are some reasons why readability might be important when you are writing code in industry?
- Here is a nice summary article

Students, write your response!

Linter

MAKE SCHOOL

- A Linter is a tool that is used by Integrated Development Environments (IDE's) like VSCode or XCode to analyze code
- Linter's can help you with checking for errors, risky constructs, or stylistic concerns
- We can use a linter to help us adhere to stylistic standards like PEP8
- Let's look at how to download an enable different linters in VSCode

Docstrings

MAKE SCHOOL

- A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition
- It will become the __doc__ special attribute of that object

```
def process_cats():
    """Read the cats file, then format and print the cat names."""
    cats_list = read_file("cats.txt")
    print_cats(cats_list)
```

Things we can use docstrings for

MAKE SCHOOL

- Docstrings show up when accessing .__doc__ property of any object
- Docstrings will also show up when we use help() with an object
- You can also use it with Pydoc to automatically generate documentation for your code!
- Let's see it in action! 😆

MAKE SCHOOL

Let’s learn turtle graphics!

Students browse: repl.it/@MakeSchool/turtlegraphicsexample?lite=true

MAKE SCHOOL

How can we combine turtle graphics and OOP?

Students, write your response!

MAKE SCHOOL

Testing

MAKE SCHOOL

Automated vs. Manual Testing

Manual Testing

- Engineers manually execute test cases
- When might manual testing be used?
- What are some drawbacks?

Students, write your response!

Automated Testing

MAKE SCHOOL

- Engineers use code to test their code
- When might automated testing be useful?
- What are some drawbacks?

Students, write your response!

Assert

```python
#Assert usage example
def average_scores(scores):
    #Assert that the list is not empty
    assert len(scores) != 0, "The list of scores is empty"
    return sum(scores)/len(scores)

scores = []
average_scores(scores)
```

```
  File "/Users/jess/Documents/Spaceman/errors.py", line 32, in <module>
    average_scores(scores)
  File "/Users/jess/Documents/Spaceman/errors.py", line 25, in average_scores
    assert len(scores) != 0, "The list of scores is empty"
AssertionError: The list of scores is empty
```

Students browse: repl.it/@MakeSchool/testingexample?lite=true

PyTest

MAKE SCHOOL

- A concise Python testing framework
- Automatically run tests you write with helpful output
- Helps you quickly identify bugs after you make changes
- Looks for files and functions that start with "`test_...`"

Demo: PyTest

5 mins

MAKE SCHOOL

This demo will show:

- How to install PyTest

```
$  pip3 install pytest
$  pytest --version
```

  - ✓ Check for `python3.7`

- How to run PyTest

```
$  pytest filename.py
```

- Plain Python output
- Read PyTest output

Different Types of Tests

MAKE SCHOOL

- **Unit Test**
- Feature Test
- Integration Test
- Performance

Differences are based on granularity

MAKE SCHOOL

Shout Outs

Students, write your response!