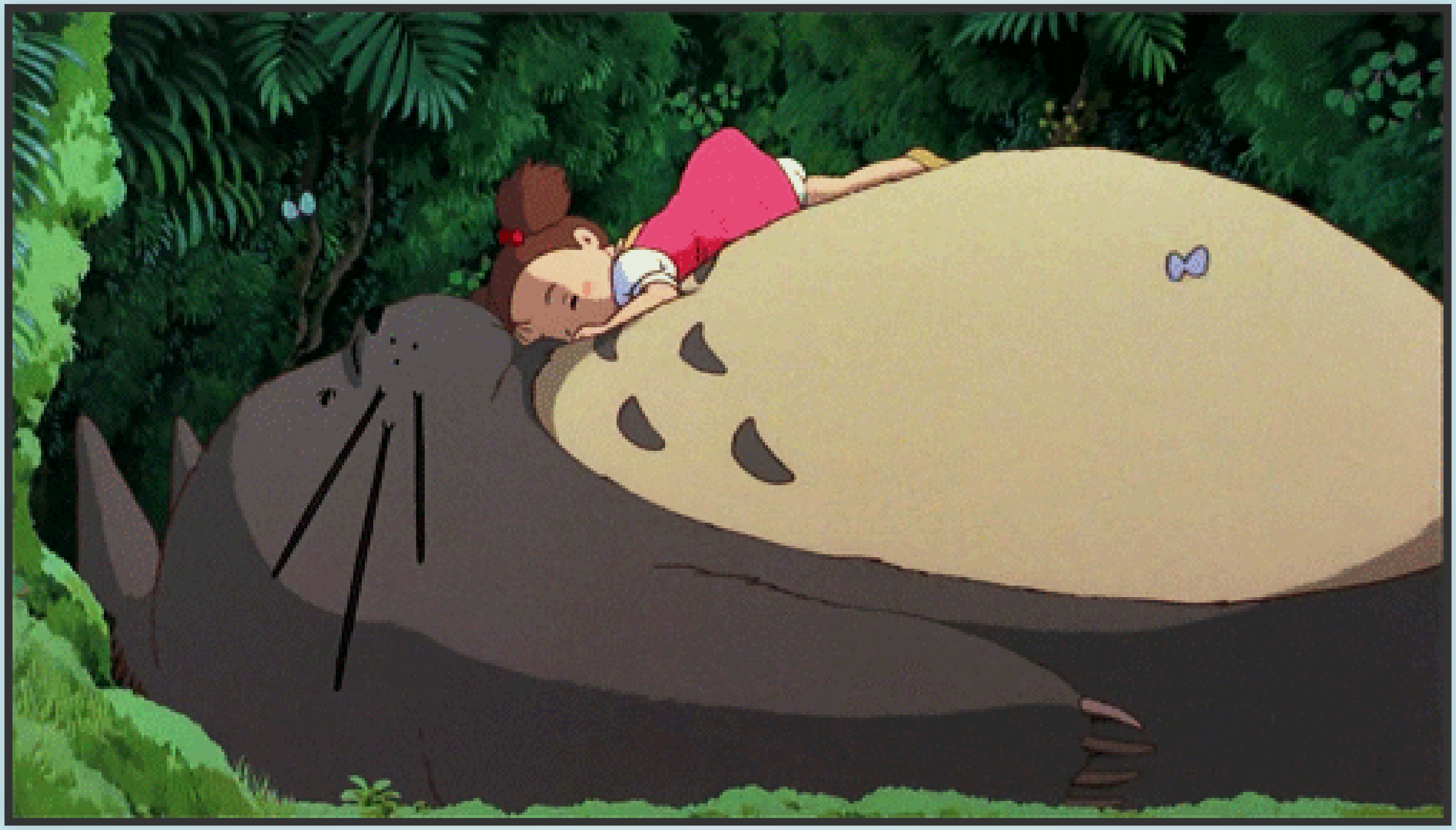
CSC110 Lecture 11: Data Classes

David Liu, Department of Computer Science

*Navigation tip for web slides: press **?** to see keyboard navigation controls.*

Announcements and today's plan

Term Test 1 done!



On the horizon

- Assignment 2 has been posted—please start early!
    - Check out the A2 FAQ
    - Additional TA office hours (starting today)
    - Review advice on academic integrity

Story so far

Data: data types, literals, basic operators, comprehensions

Functions: using built-in functions, methods; defining our own (top-level) functions

Logic: translating boolean expressions, filtering comprehensions, if statements

Function correctness: unit tests, property-based tests, proofs

Complex data: tabular data (last class)

Today you'll learn to...

1. Define and use new data types using Python data classes.
2. Create representation invariants for Python data classes.
3. Use PythonTA to check representation invariants.
4. Use the Data Class Design Recipe to design Python data classes.

Python data classes: quick recap

A **data class** is a Python data type whose purpose is to bundle individual pieces of data into a single Python object.

```
from dataclasses import dataclass


@dataclass
class Person:
    """A custom data type that represents data for a perso
    """
    given_name: str
    family_name: str
    age: int
    address: str
```

The parts of a data class definition

```
@dataclass                # decorator to create a data class
```

```
class Person:             # class header (specify class name)
```

```
    """A custom data type that represents data for a perso
```

```
    given_name: str       # Instance attribute names and type
    family_name: str
    age: int
    address: str
```

Creating a data class value

```
>>> mario = Person('Mario', 'Badr', 100, '123 Fake Street'
```

or,

```
>>> mario = Person(
...     given_name='Mario',
...     family_name='Badr',
...     age=100,
...     address='123 Fake Street'
... )
```

Accessing data class attributes

```
>>> mario.given_name
'Mario'
>>> mario.age
100
```

Exercise 1: Reviewing data classes

Representation invariants

A second look at `Person`

```
@dataclass
class Person:
    """A custom data type that represents data for a perso
    """
    given_name: str
    family_name: str
    age: int
    address: str
```

Instance attribute values are constrained by data types, but...

a person can’t have a negative age!

Representation invariant: a property of a data class' instance attributes that must always be true for every instance of the data class.

"For every `Person` instance `p`, `p.age >= 0`."

```python
@dataclass
class Person:
    """A custom data type that represents data for a perso

    Representation Invariants:
        - self.age >= 0
    """
    given_name: str
    family_name: str
    age: int
    address: str
```

By convention, `self` is the name we use for an “arbitrary” instance of the data class.

“For every `Person` instance `self`, `self.age >= 0`.”

Representation invariants vs. preconditions

Preconditions

- properties of function arguments that must be true
- can assume they are true in function body
- must ensure they are true before calling the function
- all parameter type annotations are precondtions (but not vice versa)

Representation invariants

- properties of instance attributes that must be true
- can assume they are true when using an instance
- must ensure they are true when creating an instance
- all instance attribute type annotations are representation invariants (but not vice versa)

Exercise 2: Representation Invariants

Demo: checking representation invariants with `python_ta`

Note: “`Representation Invariants:`” must be spelled exactly like that for `python_ta` to check.

Designing data classes

Given a description of some (complex) data, how do we design a data class to represent the data in Python?

Marriage license data revisited

ID	Civic Centre	Marriage Licenses Issued	Time Period
1657	ET	80	January 2011
1658	NY	136	January 2011
1659	SC	159	January 2011
1660	TO	367	January 2011
1661	ET	109	February 2011
1662	NY	150	February 2011
1663	SC	154	February 2011
1664	TO	383	February 2011

```
marriage_data = [
    [1657, 'ET', 80, datetime.date(2011, 1, 1)],
    [1658, 'NY', 136, datetime.date(2011, 1, 1)],
    [1659, 'SC', 159, datetime.date(2011, 1, 1)],
    [1660, 'TO', 367, datetime.date(2011, 1, 1)],
    [1661, 'ET', 109, datetime.date(2011, 2, 1)],
    [1662, 'NY', 150, datetime.date(2011, 2, 1)],
    [1663, 'SC', 154, datetime.date(2011, 2, 1)],
    [1664, 'TO', 383, datetime.date(2011, 2, 1)]
]
```

Goal: turn each row into an instance of a data class.

```
[1657, 'ET', 80, datetime.date(2011, 1, 1)]
```

Data Class Design Recipe

The Data Class Design Recipe is a structured process for taking a data description and turning it into a data class.

Five steps—demo time!

Step 1: Write the class header

```
[1657, 'ET', 80, datetime.date(2011, 1, 1)]
```

Step 2: Write the instance attributes for the data class

```
[1657, 'ET', 80, datetime.date(2011, 1, 1)]
```

Step 3: Write the data class docstring

```
[1657, 'ET', 80, datetime.date(2011, 1, 1)]
```

Step 4: Write an example instance

```
[1657, 'ET', 80, datetime.date(2011, 1, 1)]
```

Step 5: Document any additional representation invariants

```
[1657, 'ET', 80, datetime.date(2011, 1, 1)]
```

Exercise 3: Marriage licenses, revisited

Summary

Today you learned to...

1. Define and use new data types using Python data classes.
2. Create representation invariants for Python data classes.
3. Use PythonTA to check representation invariants.
4. Use the Data Class Design Recipe to design Python data classes.

Homework

- Start working on **Assignment 2**!
- Readings from today: 5.2 (prep), 5.3
- Readings for next class: 5.4, 5.5, 5.8

Precondition
Representation Invariant
STUDENT WHO DIDN'T PAY ATTENTION TODAY
JAKE-CLARK.TUMBLR
imgflip.com