Try-Except and Testing

David H Smith IV

University of Illinois Urbana-Champaign

Tues, Dec 07 2021

Reminders

Reminders

- Quiz Thursday
- Attempt Practice Quiz before Thursday 10am
- Topic 15 Participation due Wednesday

Try-Except

Making a Class: Poll Question

What is the result of the following code?

```
strings = ["This", "Is", "A", "String", "For", "Testing"]
for string in strings:
    print(string[3], end=" ")
```

A. IndexError
B. s i t
C. This String Testing
D. This Stri Test

Making a Class: Poll Question

What is the result of the following code?

```
strings = ["This", "Is", "A", "String", "For", "Testing"]
for string in strings:
    try:
        print(string[3], end=" ")
    except IndexError:
        continue
```

A. IndexError
B. s i t
C. This String Testing
D. This Stri Test

Types of Exceptions

A. **EOFError:** input() hits an end-of-file condition (EOF) without reading any input.
B. **KeyError:** A dictionary key is not found in the set of keys.
C. **ZeroDivisionError:** Divide by zero error.
D. **ValueError:** Invalid value.
E. **IndexError:** Index out of bounds.

Try-Catch Template

```
try:
    print(string[3], end=" ")
except <ExceptionType>:
    # Some code ...
```

or

```
try:
    print(string[3], end=" ")
except <ExceptionType> as e:
    # Other code....
    print(e)
```

A Warning Against Bare Excepts

Will this ever exit if the user trys to hit press Ctrl+C?

```
while True:
    try:
        print("Hi")
    except:
        continue
```

A Warning Against Bare Excepts

Will this ever exit if the user trys to hit press Ctrl+C?

```
while True:
    try:
        print("Hi")
    except:
        continue
```

No, it wont. So be sure to always have at least:

```
while True:
    try:
        print("Hi")
    except Exception:
        continue
```

Modules

Modules and the Standard Library

Let's import some modules we've used in the class...

```
import sys
import numpy
import requests
import math
import pygame

for module in sys.modules:
    print(module)
```

Modules and the Standard Library

Let's import some modules we've used in the class...

```
import sys
import numpy
import requests
import math
import pygame

for module in sys.modules:
    print(module)
```

1. There's lots of other modules:
   A. `flask` →http server
   B. `bottle` →a simple http server
   C. `BeautifulSoup` →an html processing library.
   D. `pandas` →a data processing library.
2. **Standard Library:** Modules that have been imported for you.

unittest

Meet how PrairieLearn grades your code

```
import unittest

class TestFoo(unittest.TestCase):
    def test0(self):
        # assert something
    def test1(self):
        # assert something
    # ...
    def testn(self):
        # assert something
```

Meet how PrairieLearn grades your code

```
import unittest

class TestFoo(unittest.TestCase):
    def test0(self):
        # assert something
    def test1(self):
        # assert something
    # ...
    def testn(self):
        # assert something
```

1. `unittest` → This is a very large module with lots of classes and functionality.

Meet how PrairieLearn grades your code

```
import unittest

class TestFoo(unittest.TestCase):
    def test0(self):
        # assert something
    def test1(self):
        # assert something
    # ...
    def testn(self):
        # assert something
```

1. `unittest` → This is a very large module with lots of classes and functionality.
2. `class TestFoo(unittest.TestCase)` →Our new class *extends* unittest.TestCase and thus gets all of it's functionality plus whatever test cases we add..

unittest Assertions

assertEqual(a, b)	assert a == b
assertNotEqual(a,b)	assert a != b
assertTrue(x)	assert bool(x) is True
assertFalse(x)	assert bool(x) is False
assertIs(a, b)	assert a is b
assertIsNot(a,b)	assert a is not b
assertIsNone(x)	assert x is None
assertIsNotNone(x)	assert x is not None
assertIn(a, b)	assert a in b
assertNotIn(a, b)	assert a not in b
assertAlmostEqual(a, b)	assert round(a - b, 7) == 0
assertGreater(a, b)	assert a >b
assertGreaterEqual(a, b)	assert a >= b
assertLess(a, b)	assert a <b
assertLessEqual(a, b)	assert a <= b

Meet how PrairieLearn grades your code

Lets see how this works in practice. Look at class info for a Colab link.