Python 5: Exceptions & Automation Testing

IN608: Intermediate Application Development Concepts

Kaiako: Tom Clark & Grayson Orr

Last Session's Content

- In-Built Functions
  - Enumerate
  - Reversed
  - Slice
  - Sorted
  - Zip
- SOLID
  - Single-responsibility principle
  - Open-closed principle
  - Liskov substitution principle
  - Interface segregation principle
  - Dependency inversion principle

Today’s Content

- Exceptions
  - Try
  - Catch
  - Finally
- Automation testing
  - Unit testing
  - Integration testing
  - Selenium Python

Exceptions

Try

- Specifies exception handlers & cleanup code for a group of statements
- Resources:
  - https://docs.python.org/3/reference/compound_stmts.html#the-try-statement
  - https://docs.python.org/3/library/exceptions.html#concrete-exceptions

```
try:
    with open('foo.txt') as f:
        data = f.read()
        print(data)
except:
    Pass

# Hello World!
```

Except

- Specifies one or more exception handlers
- Resource: https://docs.python.org/3/reference/compound_stmts.html#except

```
try:
    with open('bar.txt') as f:
        data = f.read()
        print(data)
except FileNotFoundError as e:
    print(e)

# [Errno 2] No such file or directory: 'bar.txt'
```

Finally

- Specifies a ‘cleanup’ handler
- Resource: https://docs.python.org/3/reference/compound_stmts.html#finally

```
try:
    with open('foo.txt') as f:
        data = f.read()
        print(data)
except FileNotFoundError as e:
    print(e)
finally:
    print('Cleanup')

# Hello World!
# Cleanup
```

Automation Testing

Unittest

- Unit testing framework
- Inspired by JUnit (Java)
- Support for test fixtures, cases, suites & runners
- Resource: https://docs.python.org/3/library/unittest.html
- Alternative - Pytest
    - Resource: https://docs.pytest.org/en/stable/

Unit Testing

- Individual units of software are tested
- Validate that each unit of software performs as designed

```
from unittest import TestCase, main

class Person:
    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age

    def is_legal(self):
        return True if self.age >= 18 else False

class TestPerson(TestCase):
    def setUp(self):
        self.person_one = Person('John', 'Doe', 25)
        self.person_two = Person('Jane', 'Doe', 5)

    def test_first_name(self):
        self.assertEqual('John', self.person_one.first_name)

    def test_is_legal(self):
        self.assertTrue(self.person_one.is_legal())

    def test_is_not_legal(self):
        self.assertFalse(self.person_two.is_legal())

    def tearDown(self):
        self.person_one = None
        self.person_two = None

if __name__ == '__main__':
     main(argv=[''], verbosity=2, exit=False)
```

Programming Activity (30 Minutes)

Programming Activity

- Please open 05-practical.ipynb
- Please **ONLY** answer questions 1-4
- We will go through the solutions after 30 minutes

Solutions

Integration Testing

- Individual units of software are tested as a group
- Expose bugs in the interaction between integrated units

Selenium Python

- An API for writing functional/acceptance tests using Selenium WebDriver
- Resources:
  - https://selenium-python.readthedocs.io
  - https://pypi.org/project/selenium
  - https://selenium-python.readthedocs.io/installation.html#drivers
- Alternative - Robot Framework
  - Resource: https://robotframework.org

Selenium Python

```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from unittest import TestCase, main

class TestPythonOrgSearch(TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome('chromedriver')
        self.driver.get('https://www.python.org')

    def test_python_org_search(self):
        self.assertTrue('Python' in self.driver.title)
        elem = self.driver.find_element_by_name('q')
        elem.clear()
        elem.send_keys('pycon')
        elem.send_keys(Keys.RETURN)
        self.assertTrue('No results found.' not in self.driver.page_source)

    def tearDown(self):
        self.driver.close()

if __name__ == '__main__':
    main(argv=[''], verbosity=2, exit=False)
```