WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Principles of Software Programming: Exceptions

Svitlana Vakulenko, MSc.

WS 2017

Acknowledgement: slides adopted from Dr. Jürgen Umbrich GZP SS 2014 lecture slides

This Episode

- 13:00-15:45
- Debugging:
  - Exception
  - Assertion
  - Unit test
  - Pair programming

Exception

- Error that happens during execution of a program
- When that error occurs, Python generate an exception that can be handled, which avoids your program to crash
- Class

http://www.pythonforbeginners.com/error-handling/exception-handling-in-python

Common exceptions

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

IOError

file cannot be opened

ImportError

cannot find the module (library)

ValueError

built-in operation (function) receives an argument that has an inappropriate value

KeyboardInterrupt

user hits the interrupt key (Control-C or Delete)

http://www.pythonforbeginners.com/error-handling/exception-handling-in-python

ValueError

```
number = int(input("Enter a number between 1 - 10"))

print(”you entered number", number)
```

http://www.pythonforbeginners.com/error-handling/exception-handling-in-python

try ... except ... else

```
try:
    number = int(input("Enter a number between 1 - 10”))
except ValueError:
    print(“Numbers only, please!”)
else:
    print(“you entered number", number)
```

http://www.pythonforbeginners.com/error-handling/exception-handling-in-python

try … except

```
x = 2
y = 2

try:
    print x/y
except ZeroDivisionError:
    print "You can't divide by zero, you're silly."
```

http://www.pythonforbeginners.com/error-handling/exception-handling-in-python

try ... finally

- optional `finally` clause, which is executed no matter what
- generally used to release external resources

```
try:
    f = open("test.txt",encoding = 'utf-8')
    # perform file operations
finally:
    f.close()
```

User-Defined Exception

```
class Error(Exception):
   """Base class for other exceptions"""
   pass

class ValueTooSmallError(Error):
   """Raised when the input value is too small"""
   pass

class ValueTooLargeError(Error):
   """Raised when the input value is too large"""
   pass

number = 10
while True:
   try:
       i_num = int(input("Enter a number: "))
       if i_num < number:
           raise ValueTooSmallError
       elif i_num > number:
           raise ValueTooLargeError
       break
   except ValueTooSmallError:
       print("This value is too small, try again!")
   except ValueTooLargeError:
       print("This value is too large, try again!")

print("Congratulations! You guessed it correctly.")
```

https://www.programiz.com/python-programming/user-defined-exception

Code poetry

- Reflections by Horia Botezan
- https://codepoetry.at
- 1. Festival for Digital Poetry
- Vienna Code Poetry Slam 2017
- **20. November 2017** 19:30 Uhr in **Kontaktraum TU Wien**

https://codepoetry.at/download/Reflections.py

Assertion

- assert statement is a debugging aid that tests a condition
- if condition is true, it does nothing and just continues
- if condition is false, it raises an AssertionError exception

```
def apply_discount(product, discount):
    price = int(product['price'] * (1.0 - discount))
    assert 0 <= price <= product['price']
    return price
```

https://dbader.org/blog/python-assert-tutorial

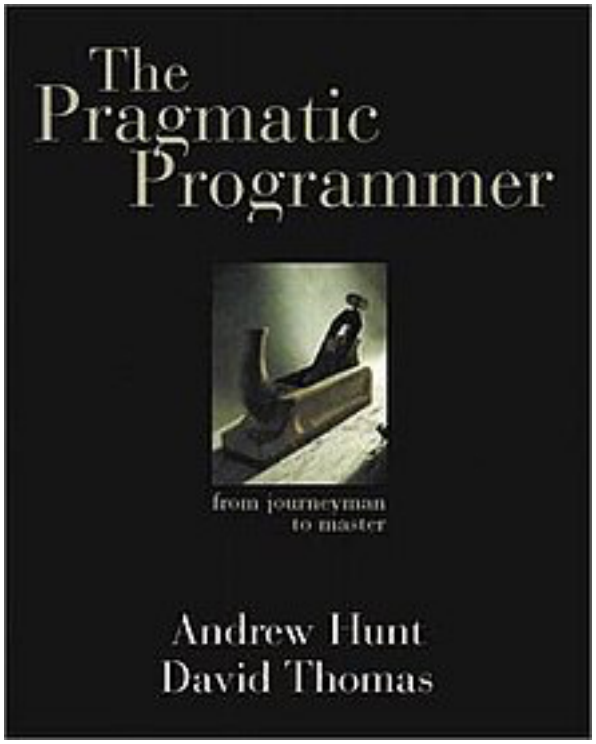
Software testing

- **White-box** (structural) testing: examining source code
- **Black-box** testing: without knowledge of implementation

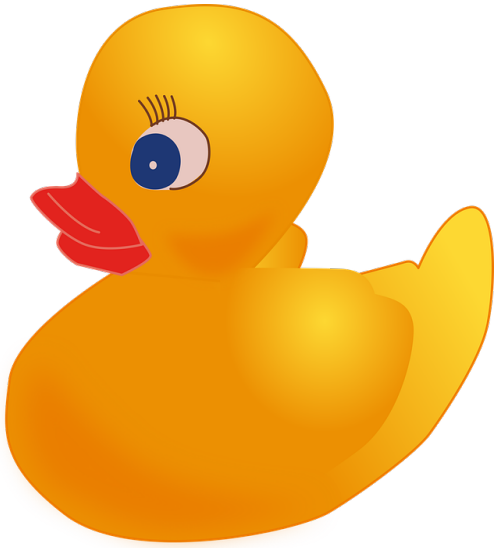
https://en.wikipedia.org/wiki/Software_testing

Rubber duck debugging

- explain the code to a rubber duck line-by-line



https://en.wikipedia.org/wiki/The_Pragmatic_Programmer



https://pixabay.com/en/bath-duck-rubber-toy-verbs-2022661/

https://en.wikipedia.org/wiki/Rubber_duck_debugging

Unit testing

```
import unittest

def fun(x):
    return x + 1

class MyTest(unittest.TestCase):
    def test(self):
        self.assertEqual(fun(3), 4)
```

http://docs.python-guide.org/en/latest/writing/tests/

https://en.wikipedia.org/wiki/Unit_testing

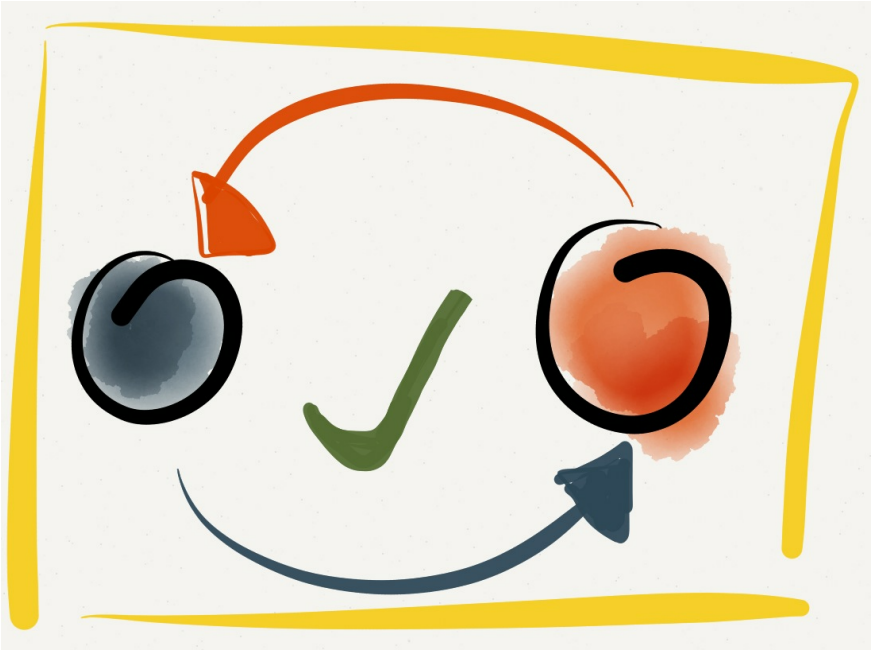
Unit testing

```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])


if __name__ == '__main__':
    unittest.main()
```

https://docs.python.org/2/library/unittest.html

Test-driven development (TDD)

- software development process
- relies on the repetition of a very short development cycle
- requirements are turned into very specific test cases



http://www.tddfellow.com/blog/2017/02/03/learning-test-driven-development-with-javascript-laws-of-tdd/

https://en.wikipedia.org/wiki/Test-driven_development

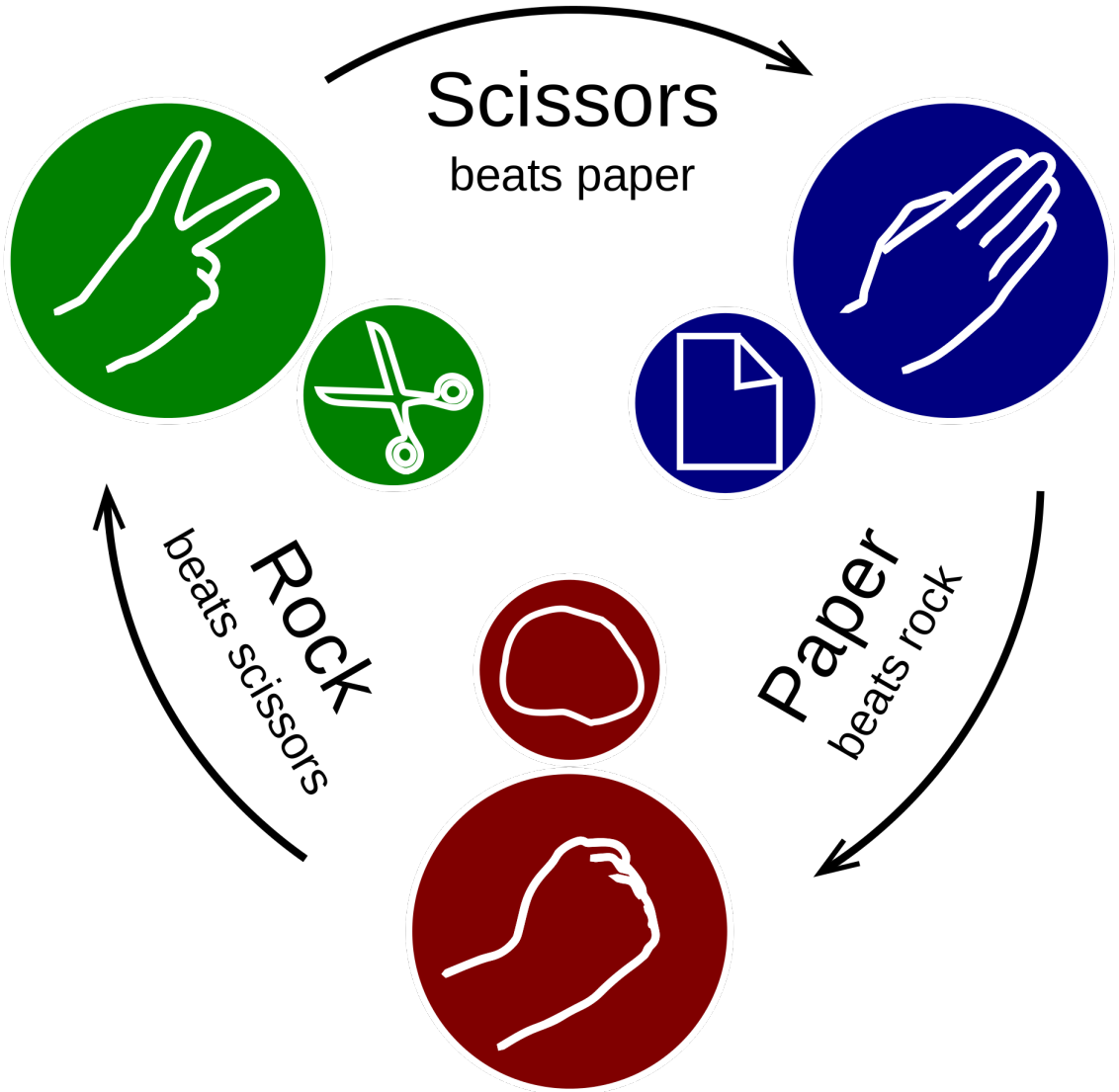
Pair programming

- agile software development technique
- **two** programmers work together at **one** workstation
- switching roles: driver and navigator

https://pixabay.com/en/driving-navigation-car-road-drive-562613/

https://en.wikipedia.org/wiki/Pair_programming

Ex.1: Rock-Paper-Scissors

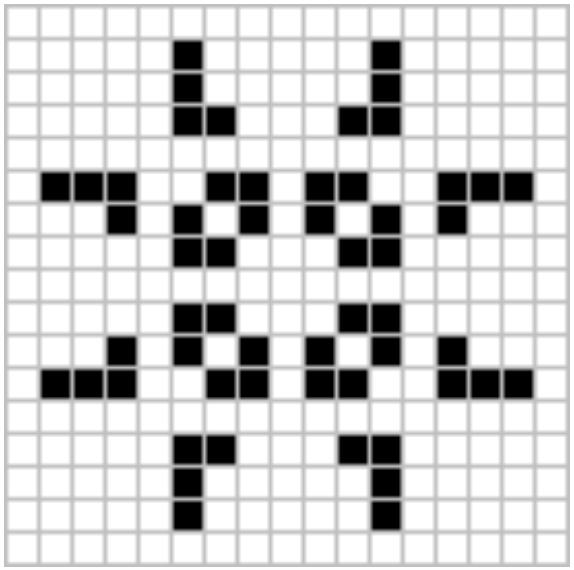


https://commons.wikimedia.org/wiki/File:Rock-paper-scissors.svg
http://www.practicepython.org/exercise/2014/03/26/08-rock-paper-scissors.html

Ex. 2: Game of Life

John Horton Conway 1970

- **underpopulation**: < 2 neighbours
- **overpopulation**: > 3 neighbours
- **reproduction**: == 3 neighbours



https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

https://www.youtube.com/watch?v=E9vclpPkSeo

PyUGAT

- https://pyug.at
- Coding Dojo
- every 4th Thursday of the month, Metalab

Kata

- (Japanese: form, pattern) exercise
- karate: sequence of basic moves
- The goal is the practice, not the solution
- http://kata.coderdojo.com/wiki/Python_Path

https://en.wikipedia.org/wiki/Kata_(programming)

http://codekata.com

Koans

- story, question, or statement to provoke the "great doubt"
- test a student's progress in Zen practice
- https://github.com/gregmalcolm/python_koans