Week 9

CSC111 - Fall 2015

Dominique Thiébaut
dthiebaut@smith.edu
Dealing with Exceptions (Chapter 7.4)

Defining Classes (Chapter 10)
# getInput: returns an integer larger than 0. Expected to be robust...

def getInput():
    while True:
        x = int(input("Enter a positive int: " ))
        if x >= 0:
            return x
        print("Invalid number: Please try again: " )

def main():
    num = getInput()
    print("you entered", num )
# getInputs: returns an integer larger
# than 0. Expected to be robust...

def getInput():
    while True:
        x = int(input("Enter a positive int: ") )
        if x >= 0:
            return x
        print("invalid input")

def main():
    num = getInput()
    print("you entered", num)
main()
# getInPut: returns an integer larger
# than 0. Expected to be robust...

def getInPut():
    while True:
        x = int( input( "Enter a positive int: " ) )
        if x >= 0:
            return x
        print( "invalid input" )

def main():
    num = getInPut()
    print( "you entered", num)

main()
RAM

Processor

Keyboard

Disk

Network
RAM

Operating System

Python Program

Processor

Keyboard

Disk

Network
The OS manages the resources of a computer.

Whenever an operation is illegal, the OS generates an Exception to flag the fault.
Some exceptions are generated by the Python interpreter.
While Exceptions are a pain in the neck, they are an integral part of programming.
Try/Except Statement

```python
try:
    python code that might generate an exception

Except exceptionXYZ:
    python code to run in case there's an exception
```
# getInput: returns an integer larger than 0. Expected to be robust...

def get_input():
    while True:
        try:
            x = int(input("Enter a positive int: ")
        except ValueError:
            print("Invalid number. Please reenter")
            continue

        if x >= 0:
            return x

        print("invalid input. Please reenter")

def main():
    num = get_input()
    print("you entered", num)
Approach to Handling Exceptions

1. Run code **without try/except statements**

2. Test thoroughly

3. Fix whatever can be fixed with "regular" python code

4. Record all exceptions that cannot be fixed otherwise, and add try/except to catch them.

5. Make the try section as small as possible.
Multiple Exceptions
(taken from Zelle)
Solutions to: \( a*x^2 + b*x + c = 0 \)
Hardening the Function

```python
def ZelleExample():
    import math
    print( "solution for quadratic equation" )
    try:
        a, b, c = eval( input( "enter 3 coefficients ( a, b, c ) " ) )
        disc = math.sqrt( b*b - 4*a*c )
        root1 = (-b + disc) / (2*a)
        root2 = (-b - disc) / (2*a)
        print( "solutions: ", root1, root2 )
        return True
    except NameError:
        print( "You didn't enter 3 numbers" )
    except TypeError:
        print( "your input were not all numbers" )
    except SyntaxError:
        print( "Forgot a comma between the numbers?" )
    except ValueError:
        print( "No real roots, negative discriminant" )
    except:
        print( "Something went wrong..." )
        return False
```
Dealing with Exceptions (Chapter 7.4)

Defining Classes (Chapter 10)
Dealing with Exceptions (Chapter 7.4)

Defining Classes (Chapter 10)
Coding Dice
Using the Objects

```python
# Create 2 dice, one with 6 sides, one with 8
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll()
d2.roll()

# display their value
print( "Die 1: ", d1.getValue() )
print( "Die 2: ", d2.getValue() )
```
We need to create the blueprint for a box…
(object)

roll()
We need to create the blueprint for the box...

```
getValue()
```
Blueprint
class def
object

blueprint
object → blueprint → object
blueprint = class
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value = 1

    def roll( self ):
        self.value = random.randrange( 1, self.noSides+1 )

    def getValue( self ):
        return self.value
# libraries
import random

# a class for a die
class Die:
    def __init__(self, n):
        self.noSides = n
        self.value = 1

    def roll(self):
        self.value = random.randrange(1, self.noSides+1)

    def getValue(self):
        return self.value
# libraries
import random

da Die Class

class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value = 1

    def roll( self ):
        self.value = random.randrange( 1, self.noSides+1 )

    def getValue( self ):
        return self.value

# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll()
d2.roll()

# display their value
print( "Die 1: ", d1.getValue() )
print( "Die 2: ", d2.getValue() )
# libraries
import random

die 1

# a class for a die
class Die:
    def __init__(self, n):
        self.noSides = n
        self.value = 1

    def roll(self):
        self.value = random.randrange(1, self.noSides + 1)

    def getValue(self):
        return self.value

# Create 2 dice, one with 6 sides
d1 = Die(6)
d2 = Die(8)

# Roll both dice
d1.roll()
d2.roll()

# display their value
print("Die 1: ", d1.getValue())
print("Die 2: ", d2.getValue())
A Die Class

```python
# libraries
import random

# a class for a die
class Die:
    def __init__(self, n):
        self.noSides = n
        self.value = 1

    def roll(self):
        self.value = random.randrange(1, self.noSides+1)

    def getValue(self):
        return self.value
```

reference to the object
A Die Class

# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value = 1

    def roll( self ):
        self.value = random.randrange( 1, self.noSides+1 )

    def getValue( self ):
        return self.value

# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll( )
d2.roll( )

# display their value
print( "Die 1: ", d1.getValue() )
print( "Die 2: ", d2.getValue() )
A Die Class

# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value = 1

def roll( self ):
    self.value = random.randrange( 1, self.noSides+1 )

def getValue( self ):
    return self.value

# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll()
d2.roll()

# display their value
print( "Die 1: ", d1.getValue() )
print( "Die 2: ", d2.getValue() )
A Die Class

```python
# libraries
import random

# a class for a die
class Die:
    def __init__(self, n):
        self.noSides = n
        self.value = 1

    def roll(self):
        self.value = random.randrange(1, self.noSides+1)

    def getValue(self):
        return self.value
```

makes the variable a "member" of the object
Exercise

Write a program that maintains a list of objects that are *cats*. Cats have a *name*, have a *breed*, may or may not be *vaccinated*, and have an *age* expressed in years.

Image credits: nicepixy.net
Examples

Minou, 3, vac, stray
Max, 1, not-vac, Burmese
Gizmo, 2, vac, Bengal
Garfield, 4, not-vac, Orange Tabby
Using Cat Objects

# Minou, 3, vac, stray
cat1 = Cat("Minou", True, "stray", 3)

if cat1.isVaccinated():
    print(cat1.getName(),
          "is vaccinated")
else:
    print(cat1.getName(),
          "is not vaccinated")
Wanted:

A program that
- outputs all the cats
- outputs only the vaccinated cats
- outputs the cats 2 or older
We stopped here last time...
Review of Classes and Objects

Cats, Cats, Cats...
Default string representation
List of Cats
Reading CSV Files of Cats
Searching for a Cat in a List
Important Concepts:

LOCAL vs. GLOBAL
What can you say about this program? Focus on the variable `a`...
What can you say about this program? Focus on the variable a...
What can you say about this program?
Focus on the variable \texttt{a}…
What can you say about this program? Focus on the variable a…
What can you say about this program?
Focus on the variable a...
```python
def main():
    func1(10)
    func2(10)
    def main():
        print(x * a)
        def func2(a):
            print(x * a)
        def func1():
            a = 3
```
```python
a = 3
def func1(x):
    print(x * a)
def func2(y):
    print(y * a)
def main():
    func1(10)
    func2(10)
main()
```

Global Variable

30

30
What can you say about this program? Focus on the variable a...
a = 3

def func1( x ):    
    print( x * a )

def func2( y ):      
    a = 8
    print( y * a )

def main():
    func1( 10 )
    func2( 10 )
    print( a )

main()

30
80
3
```python
a = 3

def func1(x):
    print(x * a)

def func2(y):
    a = 8
    print(y * a)

def main():
    func1(10)
    func2(10)
    print(a)

main()
```

Global Variable

Local Variable

30
80
3
What can you say about this program? Focus on the variable a…
```python
a = 3
def func1(x):
    print(x * a)

def func2(y):
    global a
    a = 8
    print(y * a)

def main():
    func1(10)
    func2(10)
    print(a)
```

Output:

30
80
8
Review
Review
Review
Review
Review

Blueprint

class def

value
Instantiation:
Object is instance of a Class

member variable

value

method

Class

Blueprint
class def
# libraries
import random

# a class for a die
class Die:
    def __init__( self, n ):
        self.noSides = n
        self.value = 1

    def roll( self ):
        self.value = random.randrange( 1, self.noSides+1 )

    def getValue( self ):
        return self.value

# Create 2 dice, one with 6 sides
d1 = Die( 6 )
d2 = Die( 8 )

# Roll both dice
d1.roll()
d2.roll()

# display their value
print( "Die 1: ", d1.getValue() )
print( "Die 2: ", d2.getValue() )
# libraries
import random

# a class for a die
class Die:
    def __init__(self, n):
        self.noSides = n
        self.value = 1

    def roll(self):
        self.value = random.randrange(1, self.noSides + 1)

    def getValue(self):
        return self.value

# Create 2 dice, one with 6 sides
d1 = Die(6)
d2 = Die(8)

# Roll both dice
d1.roll()
d2.roll()

# display their value
print("Die 1: ", d1.getValue())
print("Die 2: ", d2.getValue())
Pair Programming in Lab 9

Review of Classes and Objects

Cats, Cats, Cats...
- Default string representation
- List of Cats
- Reading CSV Files of Cats
- Searching for a Cat in a List
Back to Cats
Using Cat Objects

# Minou, 3, vaccinated, stray
cat1 = Cat( "Minou", True, "stray", 3 )

if cat1.isVaccinated():
    print( cat1.getName(),
           "is vaccinated" )
else:
    print( cat1.getName(),
           "is not vaccinated" )
Step 1: Implement the Class

class Cat:
    def __init__(self, name):
        self.name = name
        self.vacc = False
        self.breed = "Persian"
        self.age = 2
    def getName(self):
        return self.name
    def isVaccinated(self):
        return self.vacc
    def __str__(self):
        if self.vacc:
            s = "Vaccinated"
        else:
            s = "Not Vaccinated"
        return s

def main():
    # Minou, 3, stray
    cat1 = Cat("Minou")
    if cat1.isVaccinated():
        print(cat1)
    else:
        print("Not Vaccinated")

    cat2 = Cat("Silky")
    if cat2.isVaccinated():
        print(cat2)
    else:
        print("Not Vaccinated")

D. Thiebaut, Computer Science, Smith College
Step 2: Create a List of Cats
Step 3: Read a CSV File of Cats
Step 4: Display Only Vaccinated Cats
Step 5: Search for the Youngest Cat
Looping through a list of objects

Object-Oriented Graphics
Graphic Cars Moving Around
Car Geometry
Demo Time!
Inspiration...
Video 1

https://www.youtube.com/watch?v=4GxPrESfdnM
Video 2

https://www.youtube.com/watch?v=3wjCwtc_-hk
We stopped here last time...