Week 9

CSC111 - Fall 2018

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

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

main()
# getInPut: returns an integer larger
# than 0.
def getInPut():
    while True:
        x = eval( input( "Enter a positive int: " ) )
        if x >= 0:
            return x
        print( "You entered a negative integer." )

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

main()
# getInput: returns an integer larger
# than 0.
def getInput():
    while True:
        x = eval( input( "Enter a positive int: " ) )
        if x >= 0:
            return x
        print( "You entered a negative integer." )

def main():
    num = getInput()
    print( "num =", num )

main()
RAM

Processor

Python Program

Keyboard
Disk
Network
RAM

- Operating System
- Python Program

Processor

Keyboard

Disk

Network
Exception
Some exceptions are generated by the Python interpreter.
If Exception generated by Python Interpreter, OS "kills" the Python Program
We want to prevent exceptions from going all the way up to the operating system…
Python Program
• We want to "isolate" dangerous code areas

• We want a **tight** isolation blocks around the potentially dangerous code sections
Coding exceptions is a pain in the neck but exceptions are an integral part of programming.
Try/Except Statement

try:

python code that **might** generate an exception

Except `exceptionXYZ`:

python code to run **in case** there's an exception
# get_input: returns an integer larger
# than 0.

def get_input():
    while True:
        try:
            x = eval(input("Enter a positive int: "))
        except SyntaxError:
            print("Invalid input. Try again!")
            continue
        if x >= 0:
            return x
        print("You entered a negative integer.")

def main():
    num = get_input()
    print("num =", num)

main()
```python
# 'getInput': returns an integer larger than 0.
def getInput():
    while True:
        try:
            x = eval( input( "Enter a positive int: ") )
        except SyntaxError:
            print( "Invalid input. Try again!" )
            continue
        if x >= 0:
            return x
        print( "You entered a negative integer." )

def main():
    num = getInput()
    print( "num =", num )

main()
```

```
Enter a positive int: hello
NameError: name 'hello' is not defined
```
# get_input(): returns an integer larger than 0.
def get_input():
    while True:
        try:
            x = eval(input("Enter a positive int: "))
        except SyntaxError:
            print("Invalid input. Try again!")
            continue
        except NameError:
            print("Invalid input. Not an integer")
            continue
        if x >= 0:
            return x

print("You entered a positive integer.

def main():
    num = get_input()
    print("num = ", num)
main()
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)
Solving 2nd Degree Equations $ax^2 + bx + c = 0$

Use the quadratic formula (QF)

The roots for the equation $ax^2 + bx + c = 0$ are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$b^2 - 4ac$ is called the **discriminant** because its value indicates what type of roots there are. Specifically, if $b^2 - 4ac$ is a perfect square, we have fractional roots, if $b^2 - 4ac < 0$ there is no real roots.
Solving Equation of Degree 2

```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 )
    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..." )
```
Hardening the Function

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)

CSV Files

Defining Classes (Chapter 10)
CSV Format

• **Comma-Separated Values**

• Very popular way of representing information where all records have the same format

• Easy to implement

• Examples: [https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html](https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html)
<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height (in)</th>
<th>Weight (lbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex</td>
<td>M</td>
<td>41</td>
<td>74</td>
<td>170</td>
</tr>
<tr>
<td>Bert</td>
<td>M</td>
<td>42</td>
<td>68</td>
<td>166</td>
</tr>
<tr>
<td>Carl</td>
<td>M</td>
<td>32</td>
<td>70</td>
<td>155</td>
</tr>
<tr>
<td>Dave</td>
<td>M</td>
<td>39</td>
<td>72</td>
<td>167</td>
</tr>
<tr>
<td>Elly</td>
<td>F</td>
<td>30</td>
<td>66</td>
<td>124</td>
</tr>
<tr>
<td>Fran</td>
<td>F</td>
<td>33</td>
<td>66</td>
<td>115</td>
</tr>
<tr>
<td>Gwen</td>
<td>F</td>
<td>26</td>
<td>64</td>
<td>121</td>
</tr>
<tr>
<td>Hank</td>
<td>M</td>
<td>30</td>
<td>71</td>
<td>158</td>
</tr>
<tr>
<td>Ivan</td>
<td>M</td>
<td>53</td>
<td>72</td>
<td>175</td>
</tr>
<tr>
<td>Jake</td>
<td>M</td>
<td>32</td>
<td>69</td>
<td>143</td>
</tr>
<tr>
<td>Kate</td>
<td>F</td>
<td>47</td>
<td>69</td>
<td>139</td>
</tr>
<tr>
<td>Luke</td>
<td>M</td>
<td>34</td>
<td>72</td>
<td>163</td>
</tr>
<tr>
<td>Myra</td>
<td>F</td>
<td>23</td>
<td>62</td>
<td>98</td>
</tr>
<tr>
<td>Neil</td>
<td>M</td>
<td>36</td>
<td>75</td>
<td>160</td>
</tr>
<tr>
<td>Omar</td>
<td>M</td>
<td>38</td>
<td>70</td>
<td>145</td>
</tr>
<tr>
<td>Page</td>
<td>F</td>
<td>31</td>
<td>67</td>
<td>135</td>
</tr>
<tr>
<td>Quin</td>
<td>M</td>
<td>29</td>
<td>71</td>
<td>176</td>
</tr>
<tr>
<td>Ruth</td>
<td>F</td>
<td>28</td>
<td>65</td>
<td>131</td>
</tr>
</tbody>
</table>
Reading CSV Files

```python
# readCSVFile.py
# D. Thiebaut
# Example of a program that reads a CSV file
# and displays some of its contents.

def readCSV( fileName ):
    file = open( fileName, 'r' )
    Records = [ ]
    lines = file.readlines()
    for line in lines:
        words = line.strip().split( ',', )
        Records.append( words )
    file.close()
    return Records

def main():
    fileName = input( "File name? " )
    recs = readCSV( fileName )
    for i in range( len( recs ) ):
        record = recs[i]
        # record is a tuple
        # join each word in the tuple by a tab, and print
        # resulting string
        print( "Record ", i, "=", \t.join( record ) )

main()
```
CSV and MS Excel

The Text Wizard has determined that your data is Fixed Width.

If this is correct, choose Next, or choose the Data Type that best describes your data.

Choose the file type that best describes your data:

- **Delimited** - Characters such as commas or tabs separate each field.
- **Fixed width** - Fields are aligned in columns with spaces between each field.

More about how to import data...
CSV and MS Excel
Dealing with Exceptions (Chapter 7.4)

CSV Files

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

CSV Files

Defining Classes (Chapter 10)
Coding Dice
Using the Objects

# 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)
We need to create the blueprint for the box...

getValue()
object

blueprint
object

object

object

blueprint = class
# libraries
from random import randrange

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

    def roll(self):
        self.value = 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
# A Die Class

```python
# libraries
import random

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())
# 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
# 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
# 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
roll(d1)
roll(d2)

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

# a class for a die
class Die:
```python
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
Playing dice…
We stopped here last time...
# hw6_5.py
# Naomi Jahan
# Homework 6, Problem 5
# 25x25 squares on 600x600 window
# box with my name
from graphics import *
from random import *

def main():
    # open the graphics window
    win = GraphWin( "hw6_5", 600, 600 )

    for i in range( 24 ):
        for j in range( 24 ):
            # define the small rectangles
            r = Rectangle( Point( i*25, j*25 ), Point( (i+1)*25, (j+1)*25 ) )

            # create a random color
            red = randint( 0, 255 )
            green = randint( 0, 255 )
            blue = randint( 0, 255 )
            color = color_rgb( red, green, blue )

            # set the rectangle's color
            r.setFill( color )

            # draw the rectangle
            r.draw( win )

    # draw white rectangle with label
    rLabel = Rectangle( Point( 100, 250 ), Point( 500, 350 ) )
    rLabel.setFill( "white" )
    rLabel.draw( win )

    # draw label
    label = Text( Point( 300, 300 ), "NAOMI JAHAN" )
    label.draw( win )

    # close the graphics window
    win.getMouse()
    win.close()

main()
from graphics import *  
from random import *

win = GraphWin("PBS", 600, 600)

for x in range(0, 600, 25):
    for y in range(0, 600, 25):
        rect = Rectangle(Point(0,0), Point(25, 25))
        rect2 = Rectangle(Point(x, y), (Point(x+25, y+25)))
        red = randint(0, 255)
        green = randint(0, 255)
        blue = randint(0, 255)
        color = color_rgb(red, green, blue)
        rect2.setFill(color)
        rect2.draw(win)

rect3 = Rectangle(Point(100, 250), Point(500, 350))
words = Text(Point(300, 300), "Mickey Mouse")

rect3.setFill("white")
rect3.draw(win)
words.draw(win)

win.getMouse()
win.close()
# This program fills a 600x600 pixel window with 25x25 pixel squares of random colors then adds a large white rectangle with my name inside.

```python
from graphics import *
from random import *

def squares(i,j,win):
    sq=Rectangle(Point(i,j),Point(i+25,j+25))
    color=randColor()
    sq.setFill(color)
    sq.draw(win)

# create a random color from 3 different RGB values
def randColor():
    red = randint( 0 , 255 )
    green = randint( 0 , 255 )
    blue = randint( 0 , 255 )
    color = color_rgb( red, green, blue )
    return color

# draw a white rectangle with name inside
def nameRect(win):
    rect=Rectangle(Point(100,250),Point(500,350))
    rect.setFill("white")
    rect.draw(win)

    name=Text(Point(300,300), "TASADAY GREEN")
    name.draw(win)

# Main function
def main():
    win=GraphWin("Homework 6 Problem 5",600,600)
    for i in range(0,600,25):
        for j in range(0,600,25):
            squares(i,j,win)
    nameRect(win)
main()

```

Homework Programs

Nice Job!
From Now On...
All Programs Submitted...
Must Be Documented...
Or Else!
Up to 1 Letter Grade 
*Down*,
For the whole assignment if Documentation Missing
Min/Max Revisited...

who is associated with the largest number?

Alex, 3
Max, 4
Sophia, 10
Lujun, 2
Maggie, 5
Classes and Objects
from dieClass import Die

def main():
    # 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())

main()
Why Create a Die Class? Randint Could have Sufficed

• Modularity

• Details are hidden (Information hiding)

• The Die class can easily be enhanced/modified without having to change main program
  
  • die with a bias

  • history of rolls

  • keeping track of statistics
Write a program that maintains a list of cat objects. Cats have a name, a breed, may or may not be vaccinated and have an age expressed in years.
Examples

Minou, 3, vac, stray
Max, 1, not-vac, Burmese
Gizmo, 2, vac, Bengal
Garfield, 4, not-vac, Orange Tabby
# Example: using a cat object

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

# Print if cat is vaccinated or not
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
Good Methods
To Start With When Creating a Class

• Constructor
• Inspector Methods
• Mutator Methods
• Default string representation

• __init__()
• getValue()
• roll()
• __str__()
We stopped here last time…
Converting A String to a Boolean

```python
a = "Minou, not vaccinated"
words = a.split( "," )
```
Converting A String to a Boolean

```python
a = "Minou, not vaccinated"
words = a.split( "," )

if len( words ) == 2:
    # 1
    vaccinated = words[1].strip() == "vaccinated"
```
Converting A String to a Boolean

```python
a = "Minou, not vaccinated"
words = a.split( "," )

if len( words ) == 2:
    # 1
    vaccinated = words[1].strip() == "vaccinated"

    # 2
    vaccinated = True
    if words[1].find( "not" ) != -1:
        vaccinated = False
```
Converting A String to a Boolean

```python
a = "Minou, not vaccinated"
words = a.split( "," )

if len( words ) == 2:
    # 1
    vaccinated = words[1].strip() == "vaccinated"

    # 2
    vaccinated = True
    if words[1].find( "not" ) != -1:
        vaccinated = False

    # 3
    vaccinated = words[1].find( "not" ) != -1
```
Converting A String to a Boolean

```python
a = "Minou, not vaccinated"
words = a.split( "," )

if len( words ) == 2:
    # 1
    vaccinated = words[1].strip() == "vaccinated"

    # 2
    vaccinated = True
    if words[1].find( "not" ) != -1:
        vaccinated = False

    # 3
    vaccinated = words[1].find( "not" ) != -1

    # 4
    vaccinated = a.lower().find( "not vac" ) != -1
```
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 `a`...
What can you say about this program? Focus on the variable \texttt{a}...
What can you say about this new program? Focus on the variable a…
```python
a = 3
def func1( x):
    print( x * a)
def func2( y):
    print( y * a)
def main():
    func1( 10)
    func2( 10)
main()
```

Output:
```
30
30
30
```
```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 third program? Focus on the variable `a`...
```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()
```

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: `a`

Local Variable: `a` in `func2` function.
What can you say about this fourth 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

class def

Blueprint
Review

Blueprint
class def
Review

Blueprint
class def

value
Instantiation: Object is instance of a Class

member variable
instance variable

method

value
# libraries
import random

da1 = Die( 6 )
da2 = Die( 8 )

d1.roll()
d2.roll()

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

# 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
# Minou, 3, vaccinated, stray

```python
cat1 = Cat("Minou", True, "stray", 3)

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

Using Cat Objects
Step 1: Implement the Class

class Cat:
    def __init__(self, name, breed, age):
        self.name = name
        self.breed = breed
        self.age = age
    def getName(self):
        return self.name
    def isVaccinated(self):
        return self.vaccinated
    def __str__(self):
        s = self.name + ' (' + self.breed + '), ' + str(self.age) + ' yrs old.
        return s

def main():
    # Minou, 3, not vaccinated
    # Silky, 2, vaccinated
    # Cat1
    cat1 = Cat('Minou', 'Persian', 3)
    if cat1.isVaccinated():
        print(cat1)
    else:
        print('not vaccinated')
    cat2 = Cat('Silky', 'Burmese', 2)
    if cat2.isVaccinated():
        print(cat2)
    else:
        print('not vaccinated')
Step 2: Create a List of Cats

class Cat:
    def __init__(self, name, breed, age):
        self.name = name
        self.breed = breed
        self.age = age
    
def getName(self):
        return self.name
    
def isVaccinated(self):
        return self.vaccinated
    
def __str__(self):
        if self.vaccinated:
            s = f"{self.name} (\{self.breed\}), vaccinated, \{self.age\} yrs old."
        else:
            s = f"{self.name} (\{self.breed\}), not vaccinated, \{self.age\} yrs old."
        return s
    
def main():
        # Minou, 3, vaccinated
        cat1 = Cat("Minou", "Persian", 3)
        if cat1.isVaccinated():
            print(cat1)
        else:
            print("Missed Vaccination")
        
        # Silky, 2, not vaccinated
        cat2 = Cat("Silky", "Burmese", 2)
        if cat2.isVaccinated():
            print(cat2)
        else:
            print("Missed Vaccination")

D. Thiebaut, Computer Science, Smith College
Step 3: Read a CSV File of Cats

class Cat:
    def __init__(self, name, breed, age):
        self.name = name
        self.breed = breed
        self.age = age

    def getName(self):
        return self.name

    def isVaccinated(self):
        return self.isVaccinated

    def __str__(self):
        if self.isVaccinated:
            return f'{self.name} is vaccinated

        return f'{self.name} is not vaccinated

    def main(self):
        for cat in cats:
            if cat.isVaccinated:
                print(f'{cat.name} is vaccinated
            else:
                print(f'{cat.name} is not vaccinated

        print(f'Number of vaccinated cats: {len(cats)}')

        for cat in cats:
            if cat.isVaccinated:
                print(f'{cat.name} is vaccinated
            else:
                print(f'{cat.name} is not vaccinated

D. Thiebaut, Computer Science, Smith College
Step 4: Display Only Vaccinated Cats

class Cat:
    def __init__(self):
        self.name = ""  
        self.vacc = False
        self.breed = "" 
        self.age = 0
    def __str__(self):
        s = ""  
        if self.vacc:
            s = s + "Vaccinated"  
        else:
            s = s + "Not vaccinated"  
        return s

def main():
    Minou = Cat("Minou" , 3)  
    Silky = Cat("Silky" , 2)  
    Ninou = Cat("Ninou" , 5 , vac = False)  
    if Minou.isVaccinated:
        print("Minou is vaccinated")  
    else:
        print("Minou is not vaccinated")  
    if Silky.isVaccinated:
        print("Silky is vaccinated")  
    else:
        print("Silky is not vaccinated")  
    if Ninou.isVaccinated:
        print("Ninou is vaccinated")  
    else:
        print("Ninou is not vaccinated")  

def getName(self):
    return self.name

def isVaccine(self):
    return self.vacc
Step 5: Search for the Youngest Cat

class Cat:
def __init__(self, name, vacc):
    self.name = name
    self.vacc = vacc
    self.breed = 
    self.age = 
return

def getName(self):
    return self.name

def isVaccinated(self):
    return self.vacc

def __str__(self):
    if self.vacc:
        s = 'Yes'
    else:
        s = 'No'
return s

def main():
    # Minou, 3, unvaccinated
    cat1 = Cat('Minou', False)
    if cat1.isVaccinated:
        print('Minou is vaccinated')
    else:
        print('Minou is not vaccinated')
    
    # Silky, 3, vaccinated
    cat2 = Cat('Silky', True)
    if cat2.isVaccinated:
        print('Silky (Sphynx), vaccinated, 3 yrs old.')
    else:
        print('Silky (Sphynx), not vaccinated, 3 yrs old.')
• Looping through a list of objects

• Object-Oriented Graphics
Graphic Cars Moving Around
Car Geometry

Diagram:

- Ref Point
- 140 units
- 40 units
- 180 units
- 60 units
- 20 units
Inspiration...
Crosswalk - Low density
16.5 ms per timestep

https://www.youtube.com/watch?v=pqBSNAOoSMDc
Video 3

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