

CSC148 – Introduction to Computer Science

Lecture 1: Objects, Classes and
Exceptions

Sean Henderson

Options

- CSC108 – Introduction to Programming; for those completely new to programming
- CSC148 – Follow up to CSC108; for those who took CSC108, or previous experience with object oriented programming (in C++, Java, Python, etc.)
- CSC150 – CSC148 + selected topics from CSC108; for those with non-object oriented programming experience

CSC148 – Introduction to Computer Science

- Focus on data structures (like stacks, queues, trees, etc.) and algorithms
- Not really about programming (although there will be programming)
- Small amount of OOP, mostly to work towards other goals
- Python
- Textbook: Problem Solving with Algorithms and Data Structures using Python

Course Work

- 3 Assignments (10% each)
- 8 Lab sessions (1.25% each)
- 1 Midterm test (20%)
- 1 Final Exam (40%)
- Dates on course information sheet

Assignments

- Written in Python
- Done **individually**
- No late assignments
- Submit early, submit often

Lab Sessions

- Done in pairs
- Short exercise(s), supervised by TA
- Full marks for showing up on time, putting in a good effort to complete exercise(s)
- Start the week of Sept. 21

CSC148 Ramp Up

- Saturday Sept. 12 **or** Sept. 19
- 11am – 5pm
- Help those who skipped CSC108 acclimate to CSC148 (Python, etc.)
- Send an e-mail to jonathan.taylor@utoronto.ca with name, CDF login, and day you want to attend

Questions?

Object Oriented Analysis/Programming

- We have a problem
- How can should we model this problem in an OOP way?
- Look at the problem domain to help figure out what classes, methods and attributes you require
- Important nouns => Classes
- Important verbs => Methods
- Minor nouns => Attributes

Cliché Example

- As a young, ambitious programmer, you are asked to create an address book to keep track of contacts, etc.
- Contacts are either personal or business
- What sort of information would a personal contact have? Would it be different for a business contact?
- What sort of methods should our address book have?

Wing-101

Exceptions

- How do we handle errors?
- How do we detect them?
- In the days of yore, we would send an error via the return value of a function

```
int error = printf("Hello world!\n");  
If (error < 0) {  
    // We have failed to properly greet the world.  
}
```

Exceptions

- In Python (and Java, C++, ...) we have Exceptions
- An Exception is a special object which represents an error
- Propagates until it gets handled
- If it never gets handled, the program will crash

Using Exceptions

- When something goes wrong, an exception is raised

```
def division(x,y):  
    if (y == 0):  
        raise ZeroDivisionError("y cannot be zero")  
    else:  
        return x / y
```

Using Exceptions

- If nothing is done to handle the exception, the program will crash

```
def division(x,y):  
    if (y == 0):  
        raise ZeroDivisionError('y cannot be zero')  
    else:  
        return x / y
```

```
print division(3,4)      # What does this output?  
print division(1,0)
```

Handling Exceptions

- Python catches exceptions using a try-except block
- Similar to Java with try-catch

```
try:
    print division(3,4)      # What does this output?
    print division(1,0)
except Exception, e:
    # Place code here to deal with the exception
```

Handling Exceptions

- Can only catch particular exceptions if we choose
- Exceptions not handled will continue to propagate

```
try:
    print division(3,4)      # What does this output?
    print division(1,0)
except ZeroDivisionError, e:
    # Place code here to deal with the exception
```

Handling Exceptions

- Once we catch an exception, we are not forced to deal with it; we can log the error and then re-raise the exception

```
try:
    print division(3,4)      # What does this output?
    print division(1,0)
except ZeroDivisionError, e:
    print 'There was an exception!'
    raise # Let the exception continue; program may
          # still crash
```

Custom Exceptions

- We can create our own exceptions by subclassing the class Exception

```
class MyException(Exception):  
    '''A new type of Exception.'''  
  
    def __init__(self, value):  
        self.value = value  
  
    def __str__(self):  
        return str(self.value)
```

Handling Multiple Exceptions

- Can catch multiple exceptions at the same time

```
try:  
    do_some_work()  
except (MyException, ZeroDivisionError), e:  
    print 'Caught exception', e
```

Finally/Else

- May want to perform some cleanup code depending on whether or not there were any exceptions caught
- `finally`: Always performed no whether an exception is raised or not
- `else`: Code to be executed if and only if there were no exceptions

Finally/Else

```
try:
    do_some_work()
except (MyException, ZeroDivisionError), e:
    print 'Caught exception', e
else:
    print 'Only printed if no exceptions'
finally:
    print 'This will always be printed'
```

Wing-101