

# Teaching Programming with Python and PyGame



*Celebrating 25 Years of Excellence*

Vern Ceder  
Nathan R. Yergler  
Canterbury School  
Fort Wayne, IN

# Agenda

- Background
- Python as an introductory language
- Utilizing games in the classroom
- The Python + PyGame Experience

# Canterbury School, Ft Wayne, IN

- Independent college prep school
- 260 students in Grades 9-12
- Intro to Computer required of all students
- Courses in Pascal, C/C++, Java and AP Computer Science
- Students (and parents) are demanding consumers of these courses

# Python At All Levels

- Intro to Computer
- Python Programming
- Programming Games with Python

# Intro to Computers

- Single quarter (8 weeks) course required for all Freshmen
- Basic computer skills
  - Word processing
  - Spreadsheets and databases
  - Powerpoint
- Programming languages – Python and HTML

# Background

- Required course since the late 80's
- Until 1995, a year long course with a large Pascal component
- Move to semester length, C and HTML for “programming” component
- Factors influencing change
  - Student and staff time
  - Increasing student comfort and familiarity
  - Need to balance current practical topics with general background

# Why Programming in Intro?

- Provides practice in critical and logical thinking
- Promotes problem solving (debugging)
- Provides basic understanding of the nature of software

# Why not Pascal, C/C++ or Java?

- Too much “magic” required for even “hello world”  
*(you know....)*

“It's magic, don't ask...”  
Pascal

```
program hello;  
uses CRT;  
begin  
    write('hello world');  
end.
```

# “It's magic, don't ask...” C++

```
#include <iostream.h>

int main()
{
    cout << “hello world”;
    return (0);
}
```

# “It's magic, don't ask...” Java

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println(“hello world”);
    }
}
```

# Why not Pascal, C/C++ or Java?

- Too much “magic” required for even “hello world”
- Confusing syntax
  - Semi-colon placement
  - Need for explicit blocking with curly braces or begin-end
  - Relaxed formatting can muddle logic
  - Confusing use of symbols: =, :=, ==, ','

# Why Python for Introductory Classes?

- Shorter time to working code
- Simple programs are simple
- Command line allows interaction for first programs
- Enforced formatting makes program logic clearer

# First Day of Class

```
# Lauren S  
# 2/13/03  
# Hello World  
  
print "Hello World. This is me!"  
print "Are we just supposed to uhh figure out how this works or are  
there instructions or somethin'??"  
print "definitely!!"
```

# Fifth Day of Class

```
# Lauren S
# February 19, 2003
# Branching in Python

grade=input ("Enter your grade:")
if ((grade>=90) and (grade<=100)):
    print "Your letter grade is an 'A'. I suggest you offer tutoring
services to the person sitting to your left."
elif ((grade>=80) and (grade<=89)):
    print "Your letter grade is a 'B'. Hmm. Acceptable. But do we
settle for less?"
elif ((grade>=70) and (grade<=79)):
    print "Your letter grade is a 'C'. Think about that for a while."
elif ((grade>=60) and (grade<=69)):
    print "Your letter grade is a 'D'. I suggest you ask for tutoring
services from the person sitting to your right."
else:
    print "Huh. Must be an 'F'. Sorry!"
```

# Eighth Day of Class

```
# Lauren S
# Monday, February the twenty-fourth, Year Two Thousand and Three
# Looping

number=input("Enter a number less than 42:")
if (number >=42):
    print "YOU IDIOT! Any number larger than 43 is also larger than
42. Yes, that's right."
else:
    x=1
while (x<=number):
    print x
    x=x+1
print "I hope this sinks into your calculus packed brain. Thanks for
playing."
```

# Python “Gotchas” for beginners

- Console input – `raw_input()` requires additional processing, `input()` won't work for strings
- Weak typing isn't always weak enough
  - `A=10`
  - `B="10"`
  - `A == B --> false (0)`, but no error
  - `A + B --> type mismatch error`
- `==` vs `=`
- Integer division (as always)

# Why Games?

- “Real” programs promote more learning than toy programs
- Particularly for teenagers, games are more “real” than most programs
- Game ideas come easier for students
- Even simple games are more complex than most programming exercises
- Games can be played by other students, giving a reason to fix bugs and interface problems

# Why not Games?

- Some concepts don't easily lend themselves to games, e.g. sorting and searching
- AP and other exams require some familiarity with conventional topics and exercises

# Games vs. School Environment

- “un-academic” in minds of faculty, administration, and parents
- Time coding/playing games is time not spent on other subjects
- Large groups of game-players can disrupt all studying around them

# Games at Canterbury

- Frequent “education” of faculty and administration
- Only games written by students may be “tested” (played) in school
- Research and paper writing takes precedence over games
- Student self-control is demanded – especially when game “crazes” occur

# Programming Games with Python

- MayTerm course, Spring 2002
- Programming experience required
- Most students had little Python experience
- Three weeks to teach Python, PyGame and make a game

# Course Information

- Wide range of programming experience
- Students were given a self-paced tutorial for Python/PyGame basics
- Required to develop a simple game for successful completion of the course

# Zero to Python in 60 seconds: Tic-Tac-Toe

- Created a Tic-Tac-Toe tutorial
- Tutorial deconstructed a simple game written in Python
- Assumed no prior exposure to Python
- Experienced programmers came up to speed in about two days
- Novice programmers completed the tutorial in about a week

# Tutorial Deconstruction

- Do *something* as early as possible
- Program Initialization
- Main Loop
- Draw/Display the Board
- Respond to User Input
- Draw graphic primitives
- Check for Winner

# Lessons Learned: MayTerm

- Experienced programmers had an anti-interpreted language bias
  - “Too slow”, “too cumbersome” common assumptions
  - PyGame experience overcame most of these
- Complexity of the games proportional to level of programming experience
- Python allowed creation of “real” games in shorter amount of time

# Student Observations

- Lack of Boolean types frustrating
  - Addressed by demonstrating “if x:” syntax
- C++ programmers wanted ternary, switch capabilities
  - No good reason for ternary, just showing off
- Extreme novices overwhelmed by event model

# Programming Python

- Semester elective
- Previous programming experience required
- Focused on more advanced topics:
  - Network programming
  - GUI-driven programming
  - Game programming

# Programming Python: Curriculum Overview

- Programming primitives
- Built-in classes
- User classes
- Network sockets (low-level)
- TkInter
- PyGame (final “project”)

# PyGame in the Classroom

- Began with directed exercises
- Reused PyGame Tic-Tac-Toe tutorial
- Assigned checkers as the class project:
  - Deceptively simple logic
  - Simple graphics requirements
  - Focus was on the logic behind the game
- Follow up was network checkers: how will students architect a protocol from scratch?

# How PyGame Fit

- Python allowed coverage of more topics
- Students more interested in “real” programs
  - network, GUI, *games*
- Large projects help reinforce cumulative topics
- PyGame provides an excellent framework
- Decided on projects that required a large amount of behind-the-graphics logic

# Lessons Learned: Programming Python

- Python allowed the class to cover more than previously offered electives
- Experienced programmers were initially distrustful of indentation blocking
- Programmers with a C++/Java background were used to compiler type checking:
  - Enjoyed auto-magic nature of Python
  - Often burned in comparisons (“1” != 1)

# Lessons Learned: Programming Python

- Some students found powerful features “too easy”
  - `try...except:pass` around every block != good code
- While indented-blocks make readable code, this is not always logical, functional code
- Ease of use requires/allows focus on skills

# General Observations

- PyGame documentation could be improved
- Python textbooks: not so much

# The Future of Python at Canterbury

- Intro course seems to work better with Python
- Elective will be offered again in 2003-2004
- AP Exam switches to Java next year
- College prep schools will focus on what colleges use