

Programming in the Real World

Dr. Baldassano

chrisb@princeton.edu

Yu's Elite Education

Our programs are getting bigger!

- ▶ Our game was already over 100 lines long - most programs are worked on by teams of people for months or years
- ▶ What kinds of problems start appearing in big programs?

#1: Communicating with code

- ▶ Programs are instructions for a computer, but also need to be understandable to humans
 - ▶ Other people working on your program
 - ▶ Yourself in the future, after some time has passed
- ▶ Coding in a way that is easy for humans to work with is called “good programming style”

Importance of style

- ▶ What does this code do?

```
def DoTheThing(input)
    looper = sys.getsizeof(input) - \
            sys.getsizeof('')
    i = 0
    while i < looper:
        print(chr(ord(word[0]) - \
                32*(ord(word[0]) > 96)))
```

Importance of style

- ▶ What does this code do?

```
def PrintUppercase(word)
    for i in range(len(word)):
        print(word[i].upper())
```

Importance of Style

- ▶ Sun Microsystems (creators of Java) pushed for coding style because:
 - ▶ 80% of the lifetime cost of a piece of software goes to maintenance.
 - ▶ Hardly any software is maintained for its whole life by the original author.
 - ▶ Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.

Basics of good style

- ▶ Use comments to explain complicated code
- ▶ Give variables useful names
- ▶ Break up the program into classes and functions
- ▶ Simple code is better than clever and fast code in many cases
- ▶ No “magic numbers” - don’t use hard-coded numbers without giving them a name
- ▶ Be consistent with tabs/spacing and variable naming (capitalization, using underscores `_`, etc.)

Programming Style

- ▶ All of these sum up 3 random numbers. Which of these has the best style?

a)

```
firstOne = random.randint(0,10)
```

```
For i in range(4/2):
```

```
    firstOne = firstOne + random.randint(0,10)
```

b)

```
variable = random.randint(0,10) + \
```

```
    random.randint(0,10) + \
```

```
    random.randint(0,10)
```

c)

```
randSum = 0
```

```
for i in range(3):
```

```
    randSum += random.randint(0,10)
```


Be a polite coder!

- ▶ Whenever you're writing code that you expect will be used for more than a few days, pay attention to programming style!
- ▶ Lots of online resources, like <https://google.github.io/styleguide/pyguide.html>

“Always code as if the person who ends up maintaining your code is a violent psychopath who knows where you live.”


#2: Debugging programs

- ▶ “Bugs” are errors in programs
- ▶ Very first “bug” was found by Grace Hopper’s research group in 1947

9/9

0800 Antam started
1000 " stopped - antam ✓
1300 (032) MP-MC { 1.2700 9.037 847 025
 (033) PRO 2 2.130476415 9.037 846 895 connect
 connect 2.130676415 4.615925059(-2)
Relays 6-2 in 033 failed special speed test
in relay 10,000 test.

1100 Started Cosine Tape (Sine check)
1525 Started Multi-Adder Test.

1545  Relay #70 Panel F
(moth) in relay.

1700 Antam started.
1700 closed down.

Relay 3145
Relay 3376



Two types of bugs

- ▶ Syntax bugs
 - ▶ These crash the program because of an invalid command
 - ▶ Usually the error tells us what we did wrong
- ▶ Runtime bugs
 - ▶ The program runs, but gives the wrong answer!
 - ▶ How can we fix these?
 - ▶ What if the program is really big?

Debugging

- ▶ A “debugger” lets us run a program step-by-step to figure out where the error is
- ▶ It shows you the values of all variables, and sometimes will even let you try out changes without restarting the program

IDLE Debugger

- ▶ IDLE comes with a very basic debugger, under Debug->Debugger
- ▶ “Step” steps to next line (and goes into functions)
- ▶ “Over” steps to next line (but skips function calls)
- ▶ Can set breakpoints to quickly run to the part of the program causing trouble

#3: Testing Programs

- ▶ How can we make sure our program is working correctly?
- ▶ Can we ever be positive?
- ▶ How sure should we be, if:
 - ▶ We're writing a game for fun?
 - ▶ We're writing a game to sell?
 - ▶ We're writing business software that makes paychecks?
 - ▶ We're writing software to control pacemakers?
 - ▶ We're designing a firewall for nuclear codes?

Famous Bugs: Therac-25

- ▶ Therac-25 was a radiation therapy machine in 1985

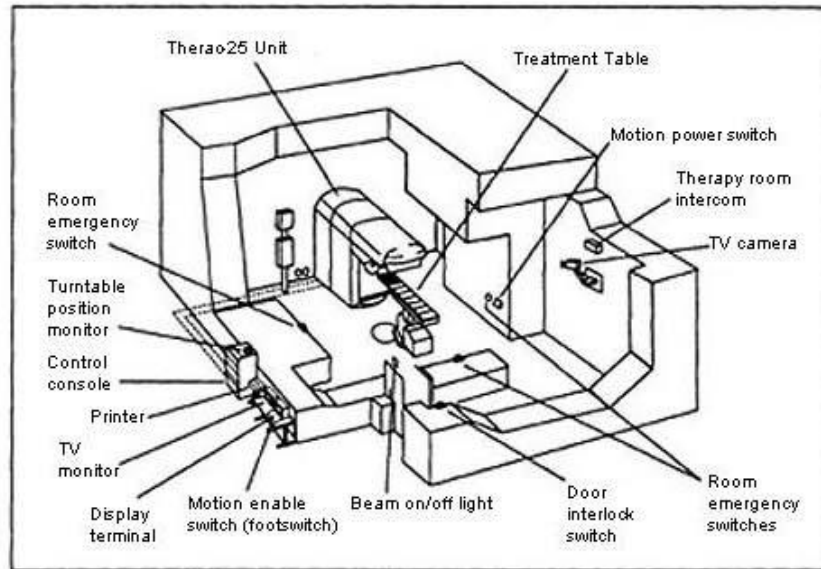


Figure 1. Typical Therac-25 facility

- ▶ At least six people were given massive radiation overdoses due to a software bug

Famous Bugs: Ariane 5

- ▶ In 1996, the European Space Agency's prototype Ariane 5 rocket (costing \$1 billion) blew up after launching, due to a type error bug



Famous Bugs: Mars Climate Orbiter

- ▶ In 1999, a NASA probe to Mars burned up in Martian atmosphere, due to a bug in unit conversion between two of its subsystems



Famous Bugs: 2003 Blackout

- ▶ In 2003, the Northeast US and Canada had an electrical grid blackout for 7 hours, due to a bug in the alarm system of an Ohio power plant



Bugs are everywhere

- ▶ More famous bugs:
https://en.wikipedia.org/wiki/List_of_software_bugs
- ▶ US NIST estimates that software bugs cost the US about \$59 billion every year
- ▶ Our best defense against bugs: testing!

Automated testing

- ▶ If we care about our program being correct, we need to write *another* program, to test our program
- ▶ This test program will run our code with lots of inputs for which we know the right answer, and make sure that everything matches up
- ▶ Sometimes you work with a team to write the tests first, so that everyone agrees on exactly what the program should do before you write it

Running tests

- ▶ Let's say my testing program runs 10 tests
- ▶ The first 9 pass, but the last one fails
- ▶ I figure out the mistake and think I've fixed it
- ▶ Which tests should I re-run?

- ▶ *Regression testing*: Always re-run all tests, to make sure our fix didn't break something else!

Testing example

- ▶ What would be good test cases for:
 - ▶ A program that calculates the day of the week for a given date?

Testing example

- ▶ What would be good test cases for:
 - ▶ A website where people can buy products?

Testing example

- ▶ What would be good test cases for:
 - ▶ A program that shuffles a deck of cards?

Tests in big programs

- ▶ How can we write tests for big programs, like a big video game (League of Legends)?
- ▶ Testing the whole thing at once is good, but we also need to test each piece of the program individually
- ▶ *Unit tests* are small tests that only test one particular part of a program (like one function)
- ▶ Why are these useful?

#4: Keeping track of code

- ▶ Two related problems when working on a programming project:
 - ▶ How can we coordinate between multiple people working on the same program?
 - ▶ How can we keep track of the changes we're making, in case we break something and need to undo changes?

Version Control

- ▶ *Version Control* systems are a way to keep a master copy of our program
 - ▶ Keep track of edits over time, so we can always roll back to a previous version
 - ▶ Allows multiple people to merge together their work into a single master copy
- ▶ Originally designed for code, but these can be used for other things too, like websites or even just text documents!

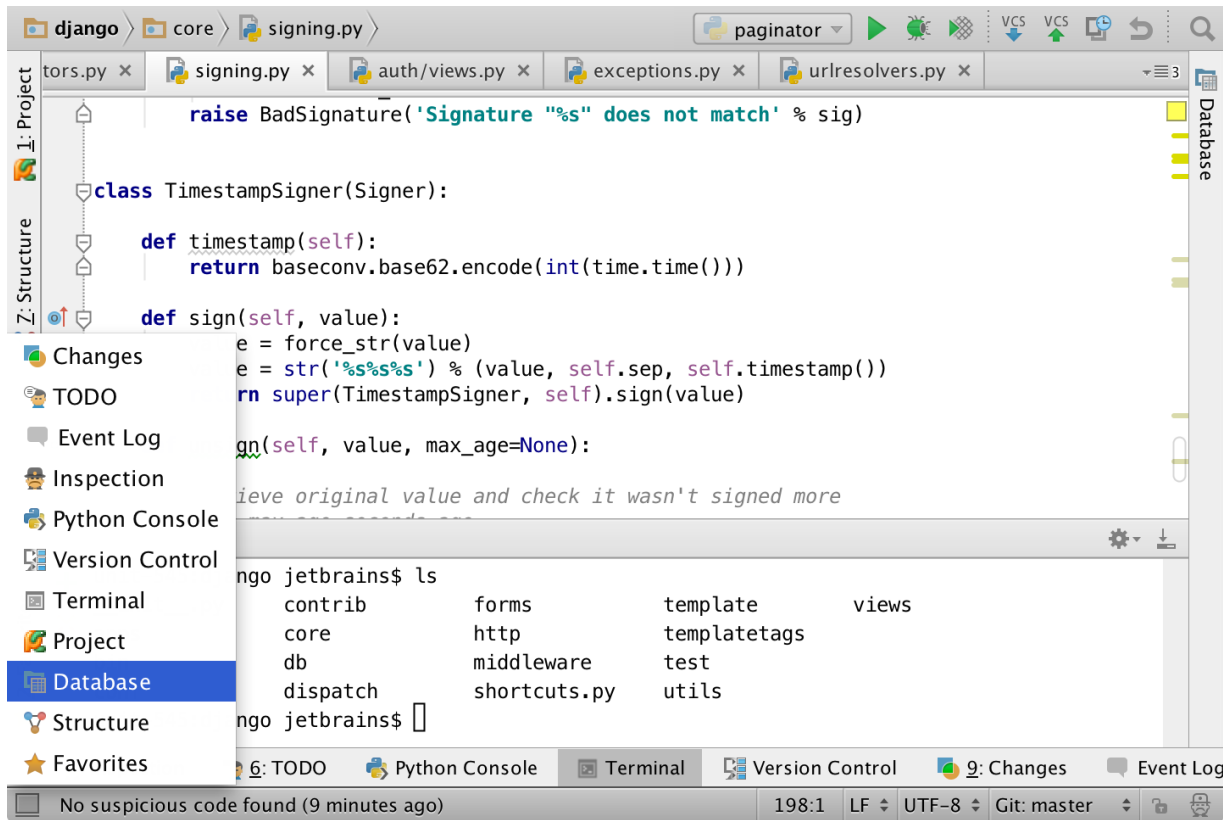
Version Control Systems

- ▶ Most popular version control system today is called “git”
- ▶ Free services like GitHub and BitBucket can back up your master copy, and let others view your code or even propose changes!
- ▶ For any project you’ll be working on for more than a few weeks and/or with other people, definitely think about using version control
- ▶ Git has a great tutorial online at <https://try.github.io/levels/1/challenges/1>

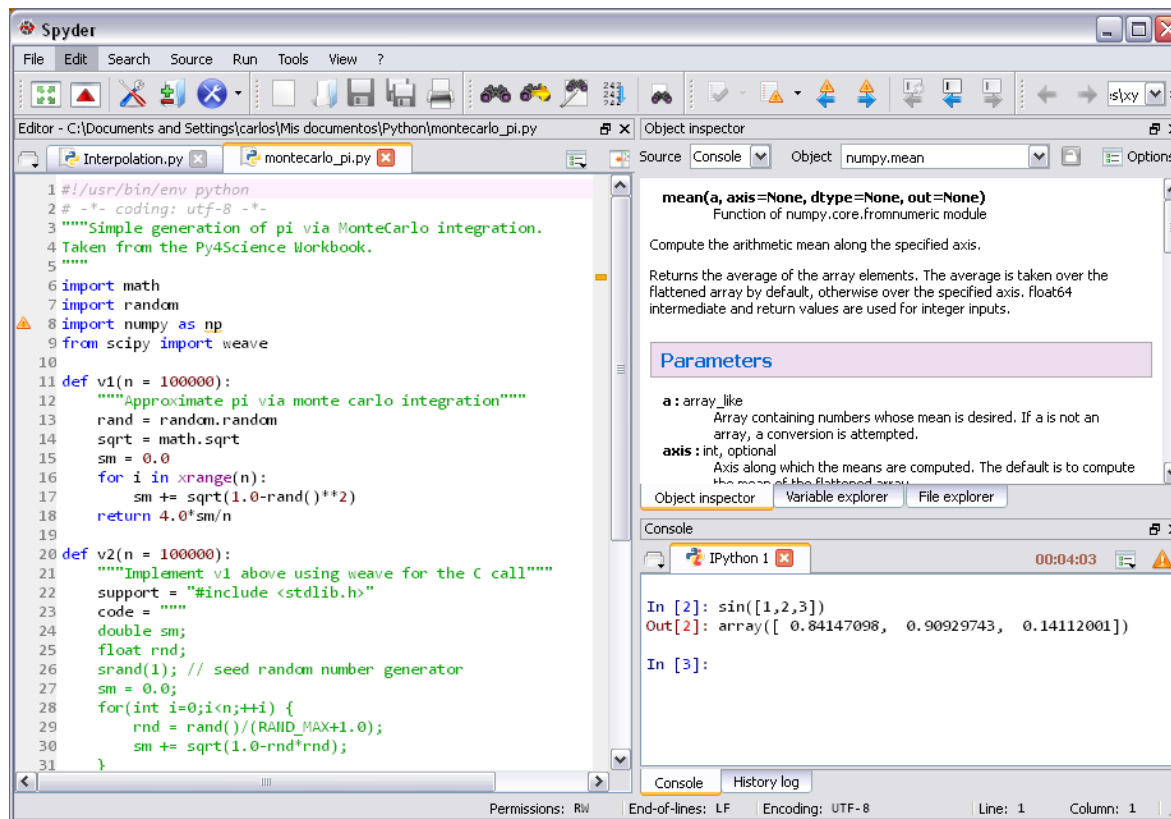
#5: Software for writing programs

- ▶ This seems like a lot to deal with - keeping track of programming style, debuggers, testing, version control...
- ▶ Most programmers use an “Integrated Development Environment” (IDE) that lets them write code and do all of these other important things within a single piece of software
- ▶ We’ve been using IDLE, which is a *very* basic IDE
 - ▶ Some IDE features we’ve been using: syntax highlighting, function hints, automatic indenting...

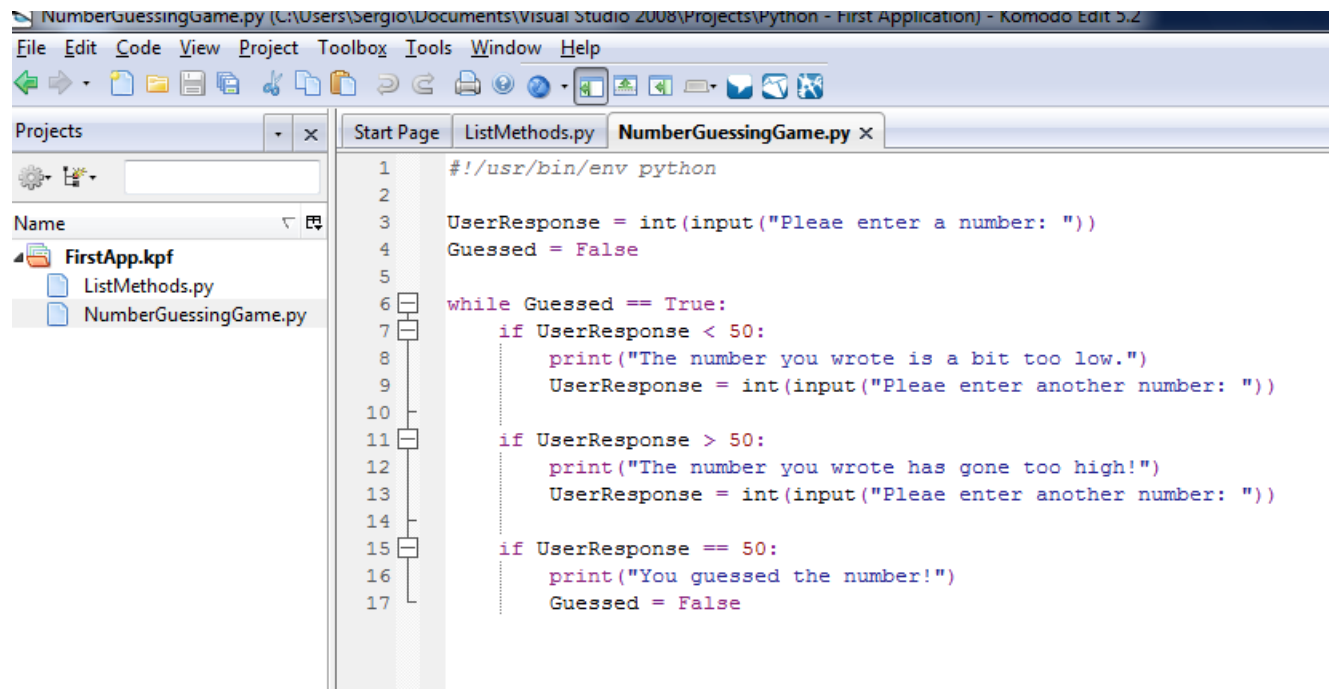
#5: Python IDEs: PyCharm



#5: Python IDEs: Spyder



#5: Python IDEs: Komodo



The screenshot displays the Komodo Edit IDE interface. The title bar shows the file path: `C:\Users\Sergio\Documents\Visual Studio 2008\Projects\Python - First Application - Komodo Edit 5.2`. The menu bar includes `File`, `Edit`, `Code`, `View`, `Project`, `Toolbox`, `Tools`, `Window`, and `Help`. The toolbar contains various icons for file operations and development tools. The `Projects` pane on the left shows a project named `FirstApp.kpf` with two files: `ListMethods.py` and `NumberGuessingGame.py`. The main editor window displays the following Python code:

```
1  #!/usr/bin/env python
2
3  UserResponse = int(input("Pleae enter a number: "))
4  Gessed = False
5
6  while Gessed == True:
7      if UserResponse < 50:
8          print("The number you wrote is a bit too low.")
9          UserResponse = int(input("Pleae enter another number: "))
10
11     if UserResponse > 50:
12         print("The number you wrote has gone too high!")
13         UserResponse = int(input("Pleae enter another number: "))
14
15     if UserResponse == 50:
16         print("You guessed the number!")
17         Gessed = False
```


IDE Features

- ▶ **Code completion** - will try to autocomplete variable and function names
- ▶ **Syntax highlighting** - will colorize text to make syntax more clear, and flag lines with syntax errors
- ▶ **Code style suggestions** - will flag lines that may not be formatted clearly (though a lot is still up to you)
- ▶ **Integrated debugging** - unlike IDLE, you can step through your code right in the same window
- ▶ **Unit testing** - can define tests for individual classes, and have them run automatically
- ▶ **Version control** - can show you whether each file is up-to-date or how it differs from the master version

Summary

- ▶ For building real programs, we need to think about some new issues:
 - ▶ Coding style
 - ▶ Debugging
 - ▶ Testing
 - ▶ Version control
 - ▶ IDEs
- ▶ All of these things have to be learned through practice, and every programmer develops their own set of tools that they like to use

Homework: Debugging a program

- ▶ Download the `palindrome_buggy.py` file from the website
- ▶ The program is supposed to check if a string is a palindrome (same forwards and backwards) but has bugs and is failing its tests
- ▶ Use the IDLE debugger to help find the bugs so that it passes all tests