Introduction

Administrative trivia

- Instructor: Ben Bolker
 - bolker@mcmaster.ca: please include 1mp3 in Subject:
 - http://www.math.mcmaster.ca/bolker
 - HH 314 (sometimes LSB 336); office hours TBA
- TA: Jake Szamosi
- Grading:
 - midterm 20%
 - final (take-home?) 30%
 - weekly assignments 30%
 - project 20%
- Laptop policy
- Course material on Github and Avenue
- Expectations of professor and students
- Textbook (none); see resources
- Course content: reasonable balance among
 - nitty-gritty practical programming instruction
 - conceptual foundations of computing/computer science
 - context/culture of mathematical/scientific computing
 - interesting applications

More interesting stuff

Using computers in math and science

- math users vs. understanders vs. developers
- develop conjectures; draw pictures; write manuscripts
- mathematical proof (e.g. four-colo(u)r theorem and other examples); computer algebra
- applied math: cryptography, tomography, logistics, finance, ...
- applied statistics: bioinformatics, Big Data/analytics, \dots
- discrete vs. continuous

Fun!

Hello, world

```
print('hello, python world!')
## hello, python world!
  Python as a fancy calculator:
print(62**2*27/5+3)
## 20760
Interlude: about Python
• scripting; high-level; glue; general-purpose; flexible
  - contrast: domain-specific scripting languages (MATLAB, R,
     PHP)
  - contrast: general-purpose compiled languages (Java, C, C++)
• relatively modern (1990s; Python 3, 2008)
• currently the 8th most popular computer language overall; most
  popular for teaching
• well suited to mathematical and scientific programming (NumPy;
  SciPy)
• ex.: Sage; BioPython
  the Mandelbrot set
  Suppose we iterate z_{n+1} = z_n^2 + c, for some complex number c,
starting with z_0 = 0. The Mandelbrot set is the set for which the
iterations do not go off to infinity. (What happens for c = 0? c = -1?
c = i? \ c = 1?
  We can iterate by hand ...
print(complex(0,0.65)**2+complex(0,0.65))
print((complex(0,0.65)**2+complex(0,0.65))**2+complex(0,0.65))
print(((complex(0,0.65)**2+complex(0,0.65))**2+complex(0,0.65))**2)+complex(0,0.65)
## (-0.4225+0.65j)
## (-0.24399375+0.10075j)
## (0.0493823875391+0.600835259375j)
  Use assignments to simplify ...
```

```
z0=0

c=complex(0,0.65)

z1=z0**2+c

z2=z1**2+c

z3=z2**2+c

print(abs(z3)<2)
```

True

The basic method for generating pretty pictures is:

- for lots of different values of c
 - $\sec z_0 = 0$
 - keep calculating $z_{n+1}=z_n^2+c$ until $\operatorname{mod}(z_{n+1})$ is greater than 2
 - record the final value of n
- translate values of n into some colour scale and plot the results

Complex arithmetic is built into Python (What is $(2 + 3i)^2 = (complex(2,3))**2?$)

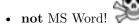
Mandelbrot set program

Note:

- easier to understand/modify than write from scratch
- build on existing components (modules)

Interfaces

- command line
- programming editor
- integrated development environment (IDE)



Features

- syntax highlighting
- bracket-matching
- hot-pasting
- integrated help
- integrated debugging tools

• integrated project management tools

most important: maintain reproducibility; well-defined workflows

Assignment

```
• superficially simple
```

```
- = is the assignment operator
```

- <variable>=<value>
- variable names
 - * what is legal? (letters, numbers, underscores, start with a letter)
 - * what is customary? convention is variables_like_this
 - * what works well? v vs. temporary_variable_for_loop
- variables can be of different types
 - built-in: integer (int), floating-point (float), complex, Boolean (bool: True or False),
 - dynamic typing
 - (relatively) strong typing
 - * try print(type(x)) for different possibilities (x=3; x=3.0; x="a")
 - * what is the result of x=a?
 - * don't be afraid to experiment!

```
x=3
y=3.0
z="a"
q=complex(1,2)
type(x+y) ## mixed arithmetic
type(int(x+y)) ## int(), float() convert explicitly
type(x+z)
type(q)
type(q)
type(x+q)
type(True)
type(True+1) ## WAT
```

Comparisons and logical expressions

• comparison: (==, !=)

```
inequalities: >, <, >=, <=,</li>
basic logic: (and, or, not)
remember your truth tables, e.g. not(a and b) equals (not a) or (not b)
a = True; b = False; c=1; d=0
a and b
not(a and not b)
a and not(b>c)
a==c ## careful!
not(d)
not(c)
```

• operator precedence: same issue as order of operations in arithmetic; not has higher precedence than and, or. When in doubt use parentheses . . .

String operations

- Less generally important, but fun
- + concatenates
- * replicates and concatenates
- in searches for a substring

```
a = "xyz"
b = "abc"
a+1 ## error
a+b
b*3
(a+" ")*5
"yz" in a
```

$Regular\ expressions$

Large topic – somewhat more advanced than 'basic programming', but interesting.

What if we are looking for some number, but we don't know what number?

```
import re
bool(re.search('[0-9]', 'Plan 9'))
```

Pattern	Description
^	Beginning of line
\$	End of line
	Any single character except newline
[]	Any single character in brackets
[^]	Any single character not in brackets
re*	0 or more occurrences of preceding
	expression
re+	1 or more occurrence of preceding
	expression
re?	0 or 1 occurrence of preceding expression
re1 re2	match re1 or re2
()	grouping

- How would you test whether a string contains a numeric value at the end (e.g. "Plan 99")?
- What if the string might contain a comma (e.g. "Plan 99,478")?
- What if you're looking for the abbreviations of rooms in Hamilton Hall (my office is HH314)?
- ... rooms in LSB or HH?

Lists, arrays, and indexing