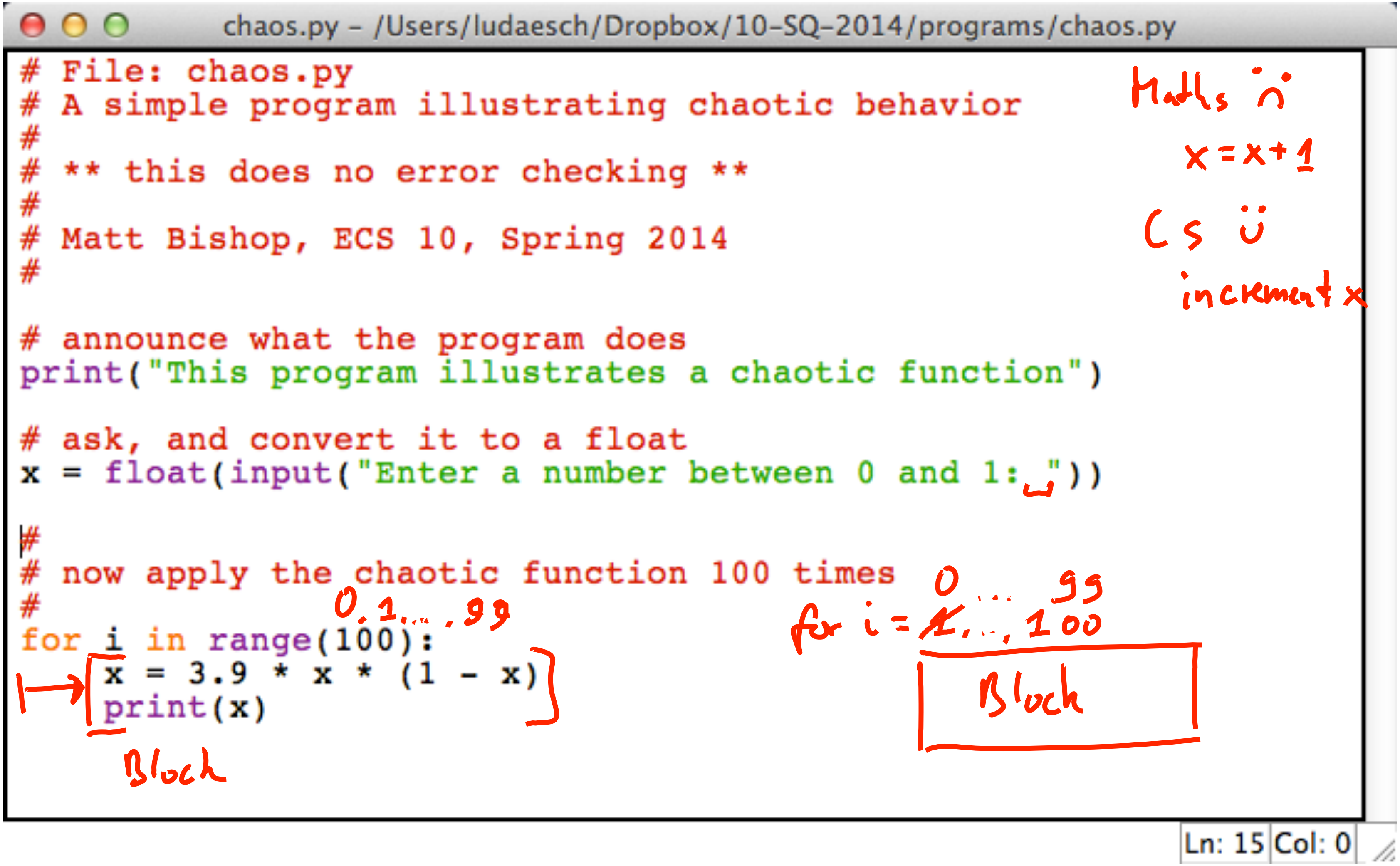
ECS-10

Bertram Ludaescher

Prediction is difficult, especially about the future …



chaos.py - /Users/ludaesch/Dropbox/10-SQ-2014/programs/chaos.py

```
# File: chaos.py
# A simple program illustrating chaotic behavior
#
# ** this does no error checking **
#
# Matt Bishop, ECS 10, Spring 2014
#

# announce what the program does
print("This program illustrates a chaotic function")

# ask, and convert it to a float
x = float(input("Enter a number between 0 and 1: "))

#
# now apply the chaotic function 100 times
#
for i in range(100):
    x = 3.9 * x * (1 - x)
    print(x)
```

0, 1, ..., 99

Block

Maths ☹

x = x + 1

CS ☺

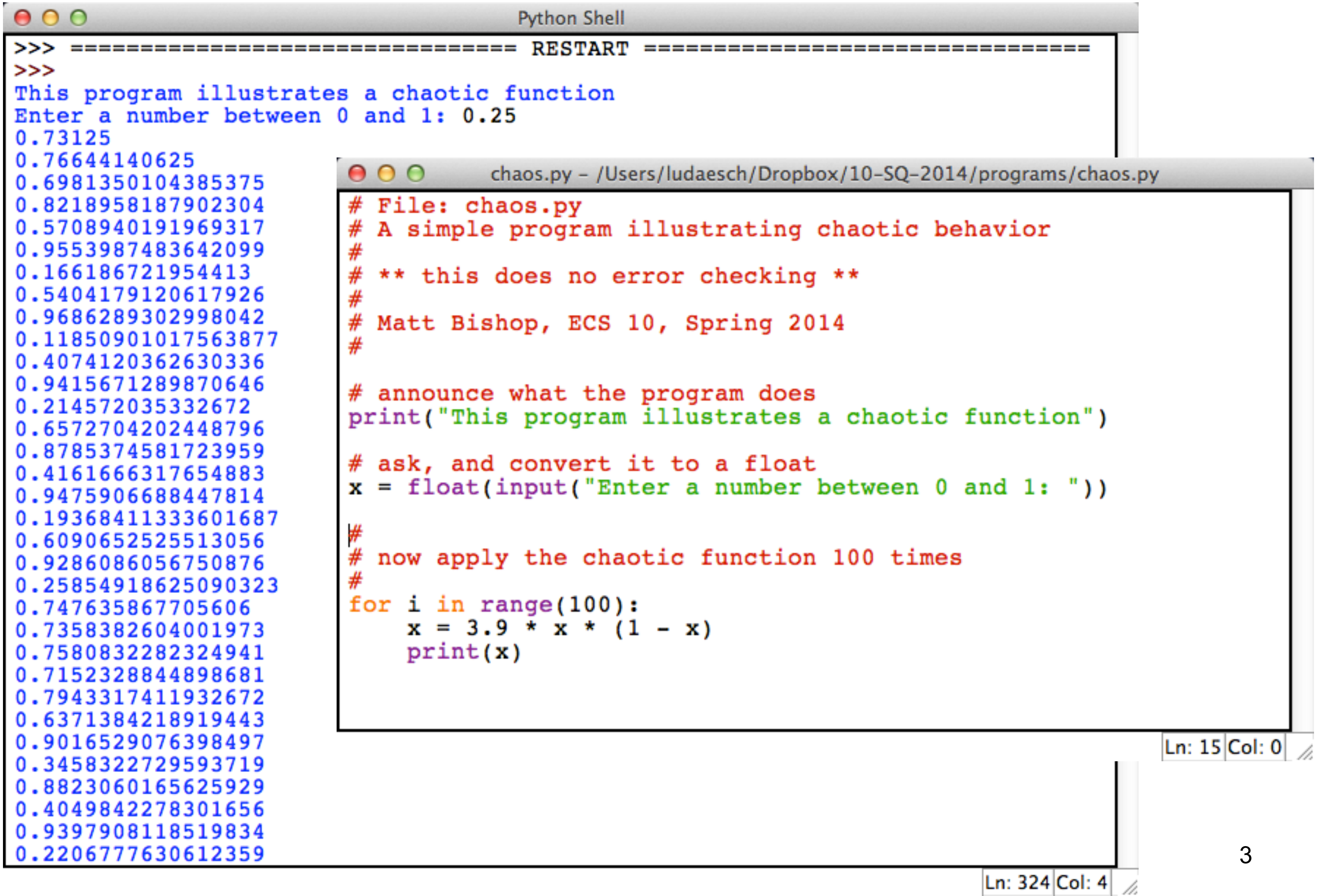
increment x

0 ... 99

for i = ~~1~~, ..., 100

Block

Sample Run …

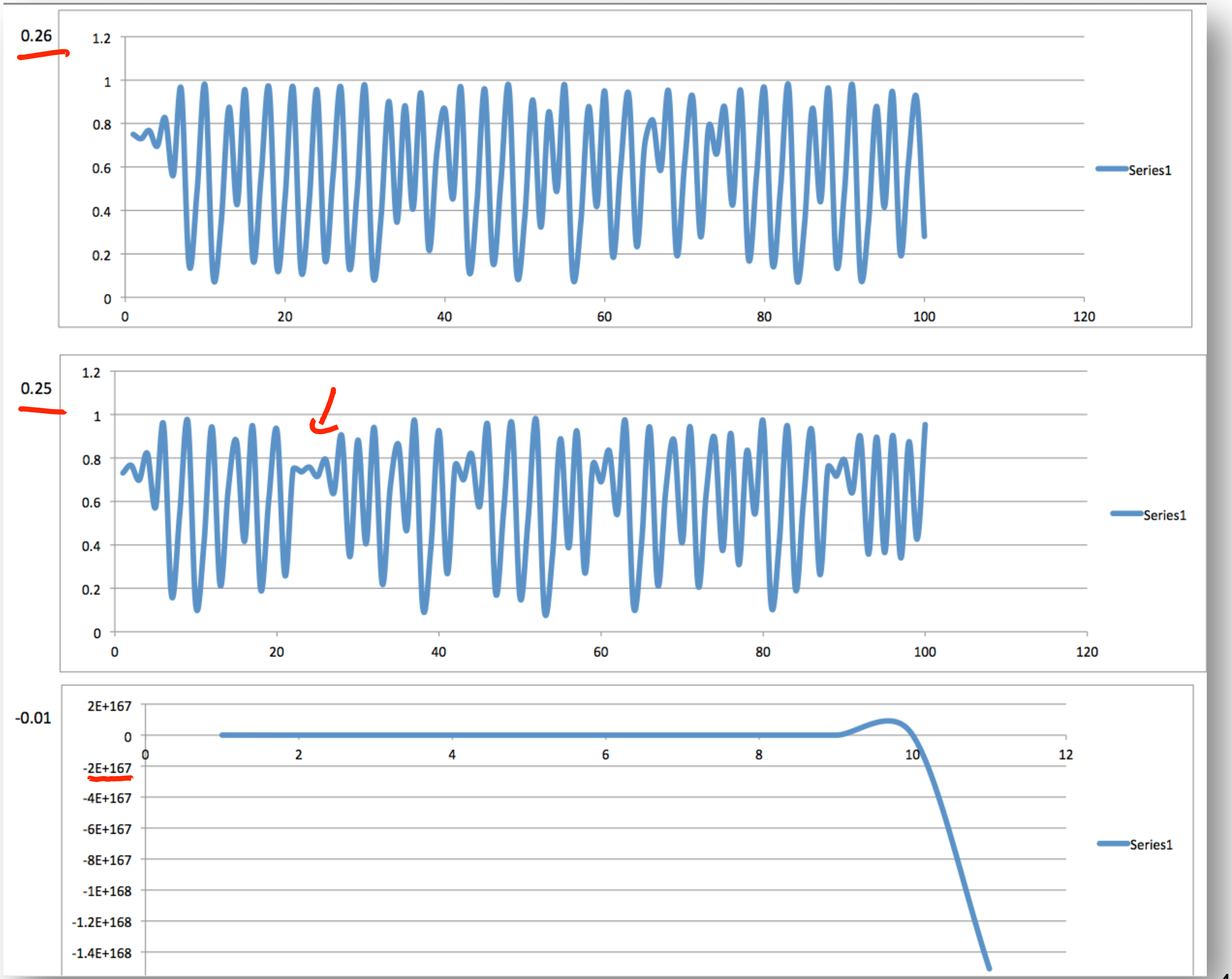

Python Shell
chaos.py - /Users/ludaesch/Dropbox/10-SQ-2014/programs/chaos.py
Ln: 15 Col: 0
Ln: 324 Col: 4

```
>>> ================================ RESTART ================================
>>>
This program illustrates a chaotic function
Enter a number between 0 and 1: 0.25
0.73125
0.76644140625
0.6981350104385375
0.8218958187902304
0.5708940191969317
0.9553987483642099
0.166186721954413
0.5404179120617926
0.9686289302998042
0.11850901017563877
0.4074120362630336
0.9415671289870646
0.214572035332672
0.6572704202448796
0.8785374581723959
0.4161666317654883
0.9475906688447814
0.19368411333601687
0.6090652525513056
0.9286086056750876
0.25854918625090323
0.747635867705606
0.7358382604001973
0.7580832282324941
0.7152328844898681
0.7943317411932672
0.6371384218919443
0.9016529076398497
0.3458322729593719
0.8823060165625929
0.4049842278301656
0.9397908118519834
0.2206777630612359
```

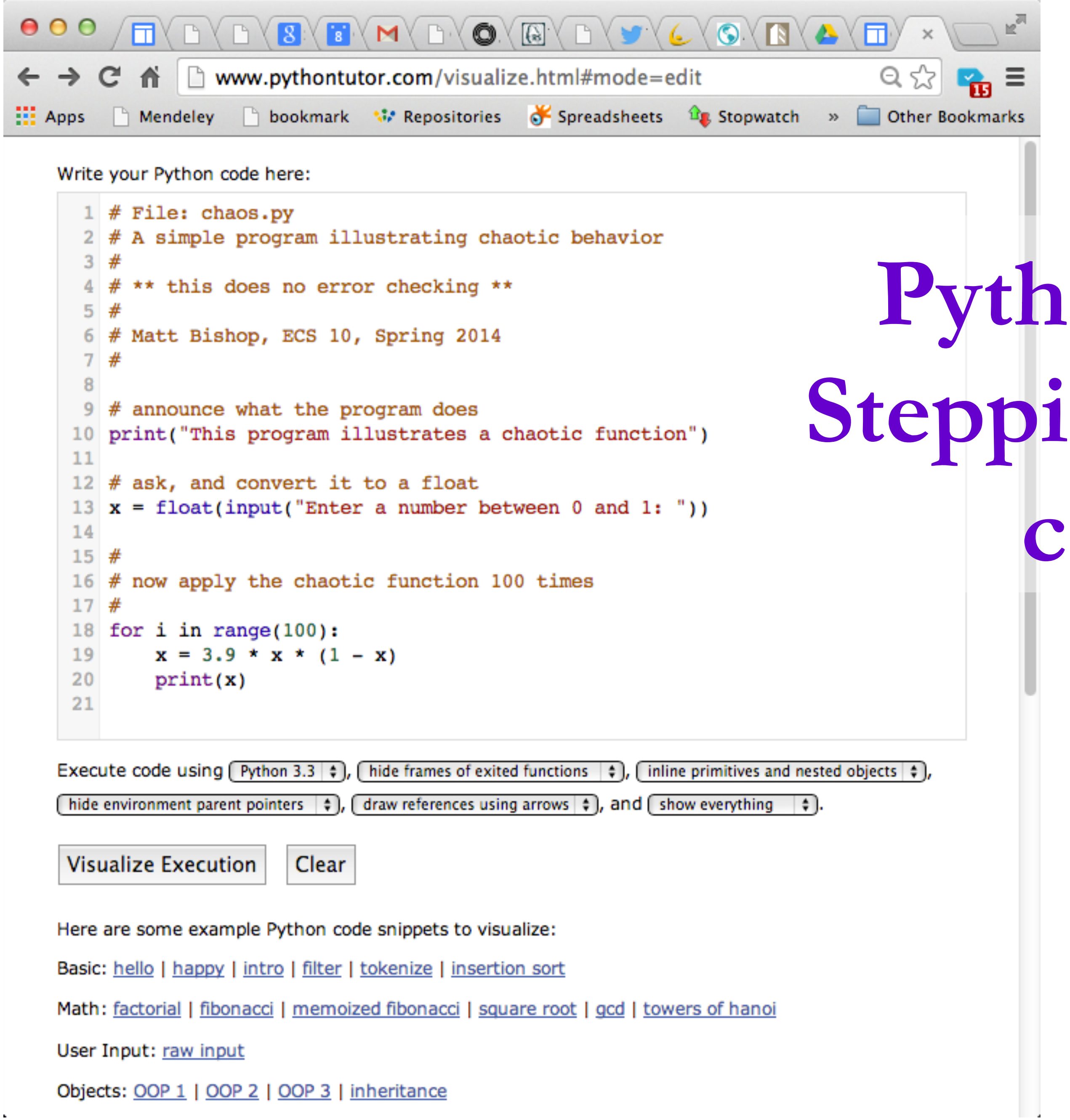
```
# File: chaos.py
# A simple program illustrating chaotic behavior
#
# ** this does no error checking **
#
# Matt Bishop, ECS 10, Spring 2014
#

# announce what the program does
print("This program illustrates a chaotic function")

# ask, and convert it to a float
x = float(input("Enter a number between 0 and 1: "))

#
# now apply the chaotic function 100 times
#
for i in range(100):
    x = 3.9 * x * (1 - x)
    print(x)
```

Where chaos reigns …

0.75036	0.73125	-0.03939
0.73054749	0.76644141	-0.1596721
0.76770663	0.69813501	-0.7221526
0.69549933	0.82189582	-4.8502617
0.82594204	0.57089402	-110.66967
0.56067097	0.95539875	-48192.737
0.96064423	0.16618672	-9.058E+09
0.14744688	0.54041791	-3.20E+20
0.49025455	0.96862893	-3.99E+41
0.9746296	0.11850901	-6.22E+83
0.09643429	0.40741204	-1.51E+168
0.3398254	0.94156713	
0.87494198	0.21457204	
0.42673221	0.65727042	
0.95406414	0.87853746	
0.17092045	0.41616663	
0.55265593	0.94759067	
0.96418668	0.19368411	
0.13466985	0.60906525	
0.45448213	0.92860861	
0.96691968	0.25854919	
0.12474544	0.74763587	
0.42581765	0.73583826	
0.95353822	0.75808323	
0.17278202	0.71523288	
0.55742074	0.79433174	
0.96214115	0.63713842	
0.14205968	0.90165291	
0.47532704	0.34583227	
0.97262586	0.88230602	
0.10383672	0.40498423	
0.36291316	0.93979081	
0.90170807	0.22067776	
0.34565943	0.67071844	
0.88209805	0.86133533	
0.4056042	0.46580343	
0.94024879	0.97043932	
0.21910591	0.1118787	
0.6672842	0.38751124	
0.86586239	0.92565049	
0.45296437	0.26840449	
0.96637183	0.76581773	
0.12673952	0.69942965	
0.4316388	0.81988847	
0.95677431	0.57591833	
0.16129319	0.95252199	
0.52758301	0.17637301	
0.97203279	0.56653574	
0.10602167	0.95773468	
0.36964619	0.15786794	
0.90873075	0.51848805	
0.32346277	0.97366695	
0.85345496	0.09999452	
0.48777139	0.35098292	
0.9744168	0.88839624	
0.09722193	0.38667859	
0.34230234	0.92491721	
0.87801264	0.27083692	
0.41771512	0.77018871	
0.94859388	0.69029244	
0.19017779	0.83377627	
0.60063976	0.54051427	
0.93549939	0.96859852	
0.23532711	0.11862018	
0.70179821	0.4077428	
0.81618218	0.94180557	
0.58511242	0.21375056	
0.94674791	0.6554389	
0.19662358	0.88077112	
0.61605471	0.40955207	
0.92247208	0.94309477	
0.27891762	0.20930139	
0.78437806	0.64542783	
0.65960956	0.89251791	
0.87565415	0.3741258	
0.42464745	0.91320718	
0.95285577	0.30911334	
0.17519443	0.8328929	
0.56355523	0.54281104	
0.95924686	0.96785214	
0.15246006	0.12134607	
0.50394236	0.41582269	
0.97453939	0.94736531	
0.09528706	0.19447071	
0.33620901	0.61094222	
0.8703728	0.92699812	
0.44001357	0.26392318	
0.96096635	0.75764416	
0.1462891	0.71611599	
0.48706553	0.79284612	
0.97434753	0.64054048	
0.09747825	0.89796866	
0.34310733	0.3573217	
0.87900029	0.89560732	
0.41479924	0.36462991	
0.94668924	0.90353226	
0.19682802	0.33993068	
0.61653834	0.87507347	
0.92203338	0.42634757	
0.28036251	0.95384375	



Python Tutor: Stepping through code …



Write your Python code here:

```
# File: chaos.py
# A simple program illustrating chaotic behavior
#
# ** this does no error checking **
#
# Matt Bishop, ECS 10, Spring 2014
#

# announce what the program does
print("This program illustrates a chaotic function")

# ask, and convert it to a float
x = float(input("Enter a number between 0 and 1: "))

#
# now apply the chaotic function 100 times
#
for i in range(100):
    x = 3.9 * x * (1 - x)
    print(x)

```

Execute code using Python 3.3, hide frames of exited functions, inline primitives and nested objects, hide environment parent pointers, draw references using arrows, and show everything.

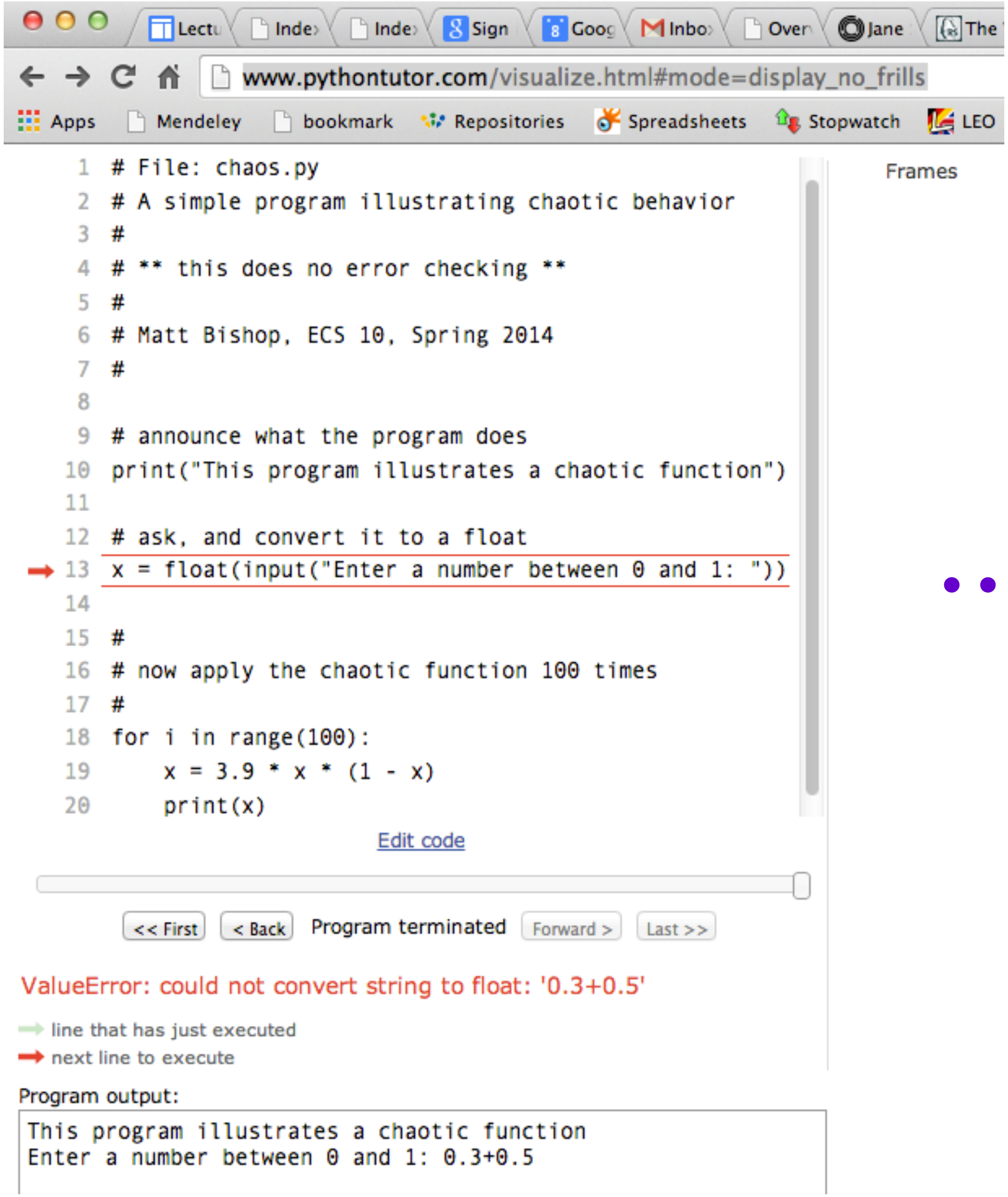
Here are some example Python code snippets to visualize:

Basic: hello | happy | intro | filter | tokenize | insertion sort

Math: factorial | fibonacci | memoized fibonacci | square root | gcd | towers of hanoi

User Input: raw input

Objects: OOP 1 | OOP 2 | OOP 3 | inheritance



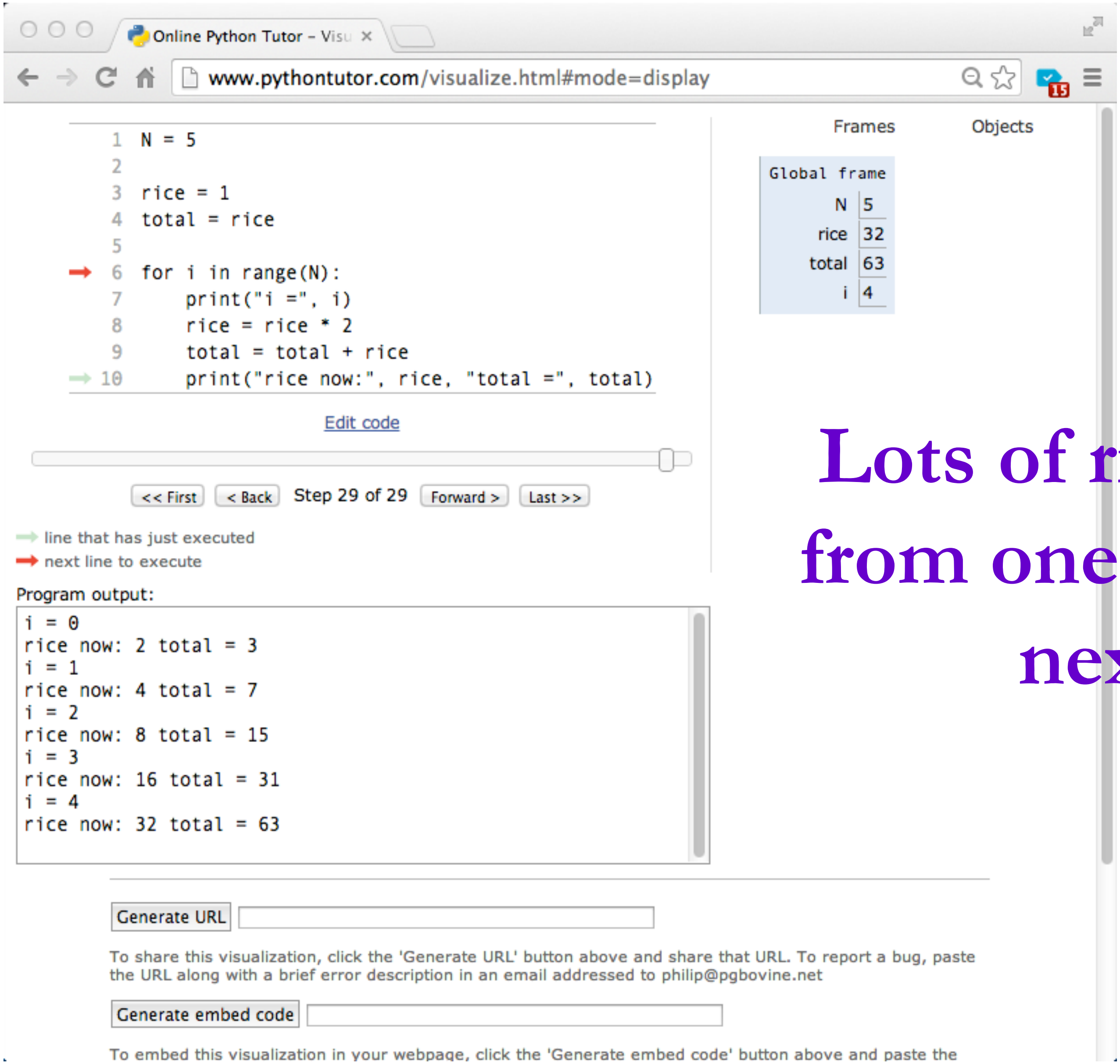
```
# File: chaos.py
# A simple program illustrating chaotic behavior
#
# ** this does no error checking **
#
# Matt Bishop, ECS 10, Spring 2014
#

# announce what the program does
print("This program illustrates a chaotic function")

# ask, and convert it to a float
x = float(input("Enter a number between 0 and 1: "))

#
# now apply the chaotic function 100 times
#
for i in range(100):
    x = 3.9 * x * (1 - x)
    print(x)
```

ValueError: could not convert string to float: '0.3+0.5'

Program output:

```
This program illustrates a chaotic function
Enter a number between 0 and 1: 0.3+0.5
```

… line by line



```
N = 5

rice = 1
total = rice

for i in range(N):
    print("i =", i)
    rice = rice * 2
    total = total + rice
    print("rice now:", rice, "total =", total)
```

Global frame

N	5
rice	32
total	63
i	4

→ line that has just executed

→ next line to execute

Program output:

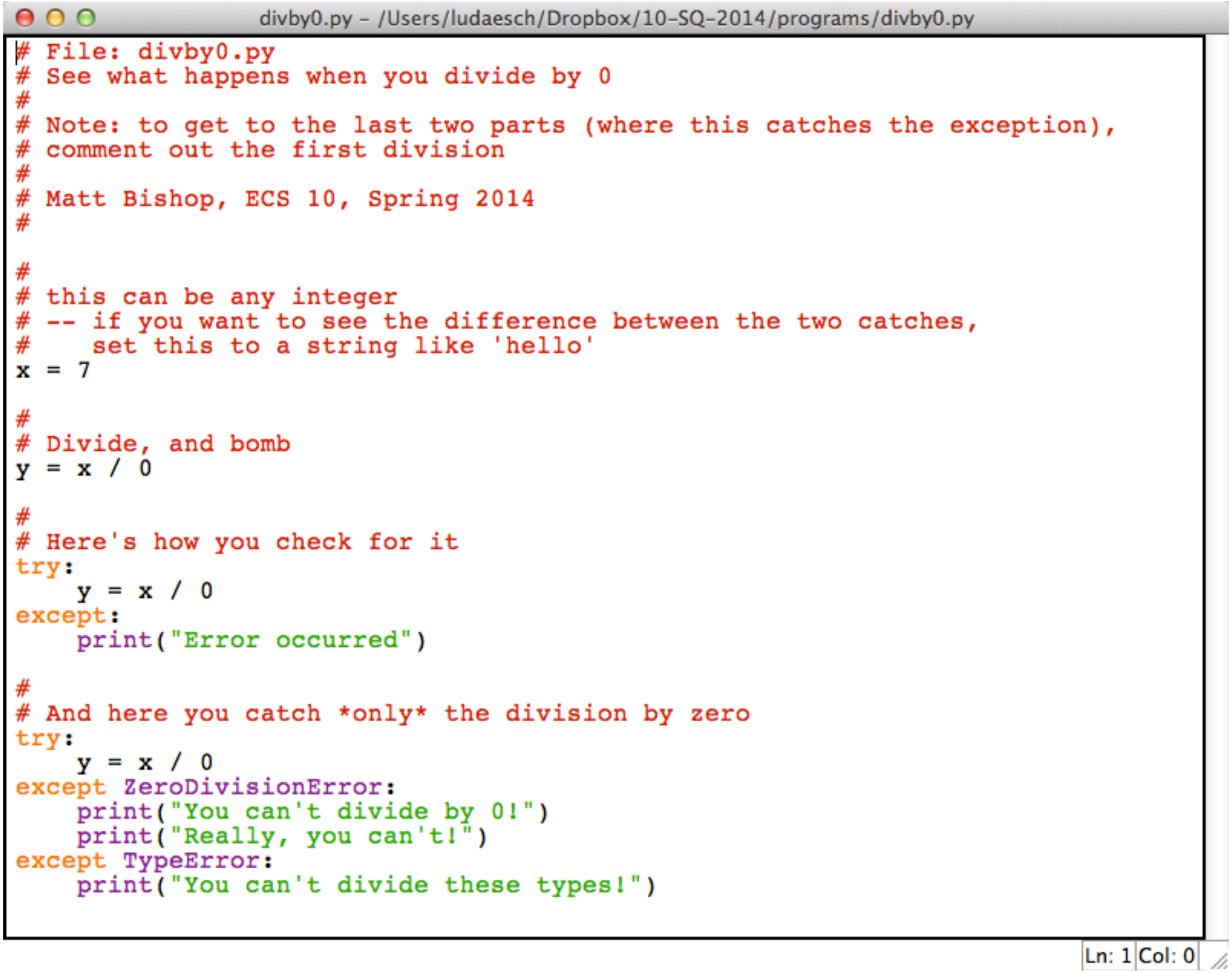
```
i = 0
rice now: 2 total = 3
i = 1
rice now: 4 total = 7
i = 2
rice now: 8 total = 15
i = 3
rice now: 16 total = 31
i = 4
rice now: 32 total = 63
```

To share this visualization, click the 'Generate URL' button above and share that URL. To report a bug, paste the URL along with a brief error description in an email addressed to philip@pgbovine.net

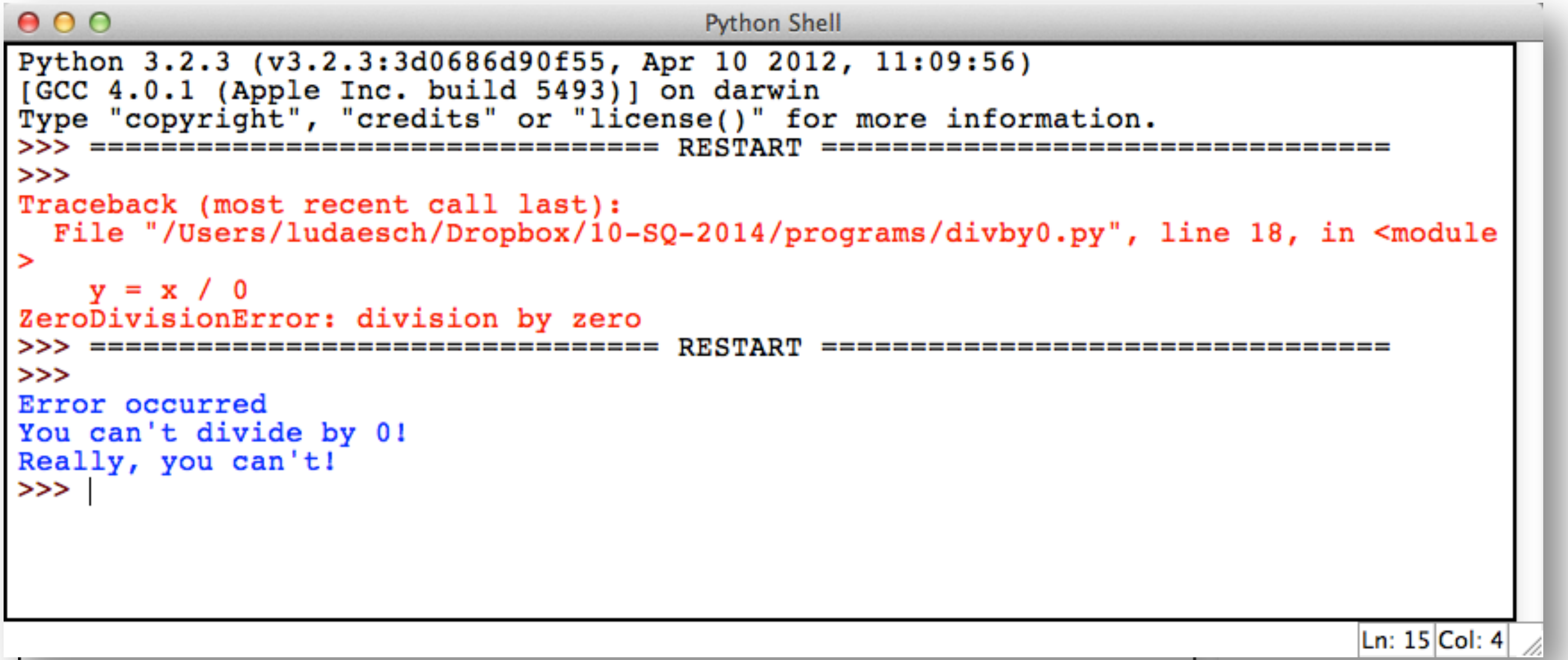
To embed this visualization in your webpage, click the 'Generate embed code' button above and paste the

Lots of rice: doubling from one square to the next … …

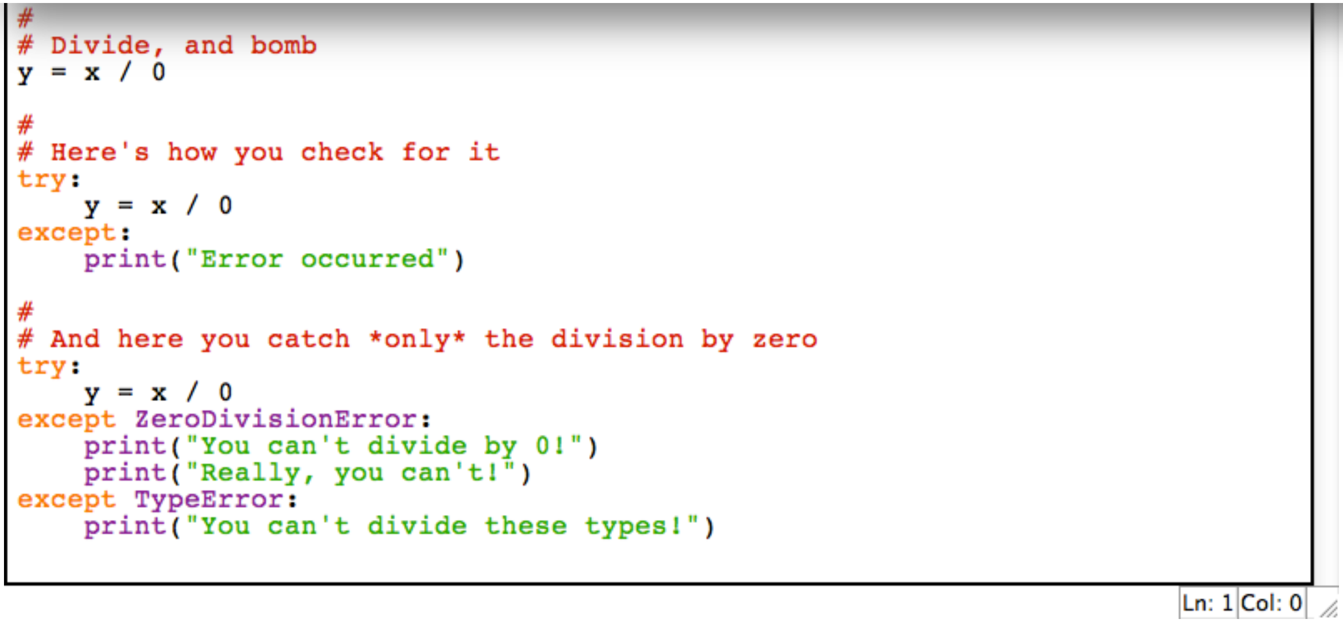
Exception Handling


divby0.py – /Users/ludaesch/Dropbox/10-SQ-2014/programs/divby0.py
Ln: 1
Col: 0

```
# File: divby0.py
# See what happens when you divide by 0
#
# Note: to get to the last two parts (where this catches the exception),
# comment out the first division
#
# Matt Bishop, ECS 10, Spring 2014
#


#
# this can be any integer
# -- if you want to see the difference between the two catches,
#    set this to a string like 'hello'
x = 7

#
# Divide, and bomb
y = x / 0

#
# Here's how you check for it
try:
    y = x / 0
except:
    print("Error occurred")

#
# And here you catch *only* the division by zero
try:
    y = x / 0
except ZeroDivisionError:
    print("You can't divide by 0!")
    print("Really, you can't!")
except TypeError:
    print("You can't divide these types!")
```



```
Python 3.2.3 (v3.2.3:3d0686d90f55, Apr 10 2012, 11:09:56)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>> 
Traceback (most recent call last):
  File "/Users/ludaesch/Dropbox/10-SQ-2014/programs/divby0.py", line 18, in <module
>
    y = x / 0
ZeroDivisionError: division by zero
>>> ================================ RESTART ================================
>>> 
Error occurred
You can't divide by 0!
Really, you can't!
>>> |
```



```
#
# Divide, and bomb
y = x / 0

#
# Here's how you check for it
try:
    y = x / 0
except:
    print("Error occurred")

#
# And here you catch *only* the division by zero
try:
    y = x / 0
except ZeroDivisionError:
    print("You can't divide by 0!")
    print("Really, you can't!")
except TypeError:
    print("You can't divide these types!")
```

Points to remember

- Python variables and expressions have **types**
  - We can ask, e.g., `type(x)` to learn about x’s type.
  - There are functions to convert between types
- Loops are used to execute the same steps over and over:
  - `for` loops:
    - do something a **number of times**, e.g., by stepping through a sequence values
  - `while` loops (later):
    - do something until a **condition** becomes true

Points to remember

- Sometimes Python "crashes" during program execution
- We get an **error message** which helps to fix the error (read the message carefully!)
- We can use **exception handling** to deal with situations that might raise a runtime warning or error.