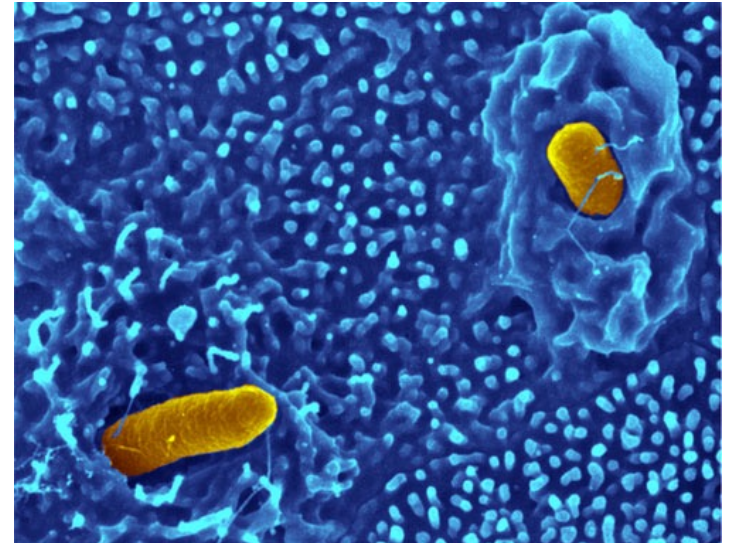


Welcome back to



- Today: Continuing introduction to programming in Python!
- Today (Thursday) 1:15-3:15 PM
Lab in Shan B442
 - Learn to use the submission system
 - Friendly intro to Python and the command line
- Homework 0 is due on Monday at 11:59 PM



Getting help

CS 5 Green: *Welcome!*

Lectures, Homework Assignments, and Readings

Week	Tuesday	Thursday	Homework	Re
0	08/31/21 - Lec 0: Introduction + picobot (M)	09/02/21 - Lec 1: Intro to Python (M)	Homework 0	

Course Resources

Course Syllabus
Work/Pairs Policy
Getting Help
Textbook
Piazza Q&A System
Submission site

Office hours and
Grutoring hours
Posted here!

Piazza

Slicing

```
>>> myDNA = "AATGCCGTGCTT"
```

0	1	2	3	4	5	6	7	8	9	10	11
A	A	T	G	C	C	G	T	G	C	T	T

```
>>> myDNA[0:4]  
'AATG'
```

```
>>> myDNA[3:7]  
'GCCG'
```

```
>>> myDNA[1:]  
'ATGCCGTGCTT'
```

```
>>> myDNA[:4]  
'AATG'
```

```
>>> myDNA[10:12]  
'TT'
```



Indexing and slicing: you try

0 1 2 3 4 5 6 7 8 9 1 1
0 1

```
>>> S = "I love Spam!"
```

```
>>> len(S)
```

```
>>> S[7]
```

```
>>> S[13]
```

```
>>> S[2:6]
```

```
>>> S[7:42]
```

I found "love"!



Indexing and slicing: negative indices

```
      0 1 2 3 4 5 6 7 8 9 1 1
      0 1
>>> S = "I love Spam!"
```

```
>>> S[len(S)-1]
'!'
```

```
>>> S[-1]
'!'
```

```
>>> S[1:-1]
' love Spam'
```

Fancy slicing

```
      0 1 2 3 4 5 6 7 8 9 1 1  
      0 1  
>>> A = "abcdefghijkl"
```

```
>>> A[1:9:3]  
'beh'
```

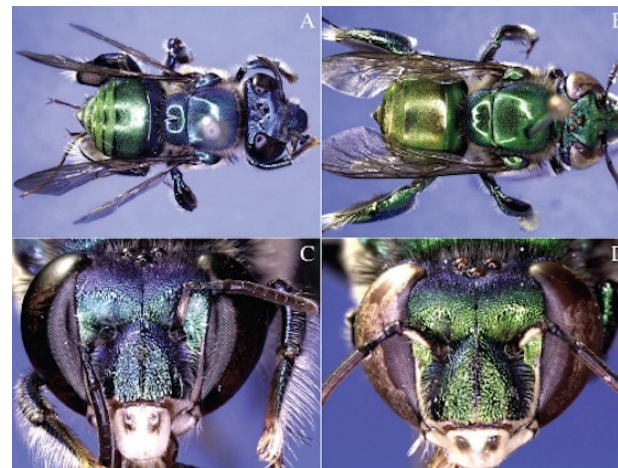
```
>>> A[5:0:-1]  
'fedcb'
```

Adding strings

```
>>> healthFood = "spam"
>>> healthFood
'spam'
>>> healthFood + "!!!"
'spam!!!'
>>> healthFood
'spam'
>>> healthFood = healthFood + "ityspam"
>>> healthFood
'spamityspam'
```



Remixing species names



```

                                0123456789111
                                012
jellyFish = "Bazinga rieki"
                                01234567891111111
                                012345
orchidBee = "Euglossa bazinga"

```

How could we name a new jellyfish *Bazinga bazinga* by slicing and adding strings?

```
>>> newJellyFish =
```


Lists

```
>>> primes = [2,3,5,7,11]
```

```
>>> biologists = ["McClintock", "Blackburn", "Franklin"]
```

```
>>> L = [2, "turtle", 11]
```

```
>>> M = [2, "turtle", 11, ["spam", "spamity", "spam"] ]
```



Lists index/slice the same as strings

```
>>> M = [2, "turtle", 11, ["spam", "spamity", "spam"] ]
```

```
>>> len(M)
```

```
4
```

```
>>> M[2]
```

```
11
```

```
>>> M[3]
```

```
['spam', 'spamity', 'spam']
```

```
>>> M[3][0]
```

```
???
```

```
>>> M[2:]
```

```
???
```

Getting a list with a range of numbers: the range function

```
>>> list(range(0,25))  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,  
17, 18, 19, 20, 21, 22, 23, 24]
```

```
>>> list(range(25))  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,  
17, 18, 19, 20, 21, 22, 23, 24]
```

```
>>> list(range(3,9))  
[3, 4, 5, 6, 7, 8]
```

```
>>> list(range(3,15,3))  
[3, 6, 9, 12]
```

Addition in lists

```
>>> my_list = [42, 47, 23]
>>> new_list = my_list + 100
BARF!
```

```
>>> new_list = my_list + [100]
>>> new_list
[42, 47, 23, 100]
```

```
>>> my_list
[42, 47, 23]
```

```
>>> new_list = new_list + new_list
>>> new_list
[42, 47, 23, 100, 42, 47, 23, 100]
```

Types of data in Python

```
>>> good_num = 42
```

← Integer

```
>>> pi = 3.1415926
```

← "Floating point" number

```
>>> special = [2.718, 3.141, 42]
```

← List

```
>>> best_food = "spam"
```

← String (single or double quotes work)

```
>>> ok_food = 'chocolate'
```

←



Defining your own functions!

```
def dbl(x):  
    return 2 * x
```



Syntax

```
def function_name(parameters):  
    function body here...
```



Notice the indentation.
This is done using “tab”
and it’s absolutely
necessary!

Functions can have more than one line

```
def dbl(x):  
    return 2 * x
```



```
def dbl(my_input):  
    my_output = 2 * my_input  
    return my_output
```

A variant which would not work:

```
def dbl(my_input):  
    2 * my_input = my_output  
    return my_output
```



What's wrong here?

Docstrings

```
def dbl(x):  
    '''This function takes a number x as input  
    and returns 2 * x'''  
  
    return 2 * x
```



This is sort of like
teaching your program
to talk to you!



Single quotes and
double quotes are both
fine!



Hey look! Three
talking turtles!
Amazing!!!

Comments

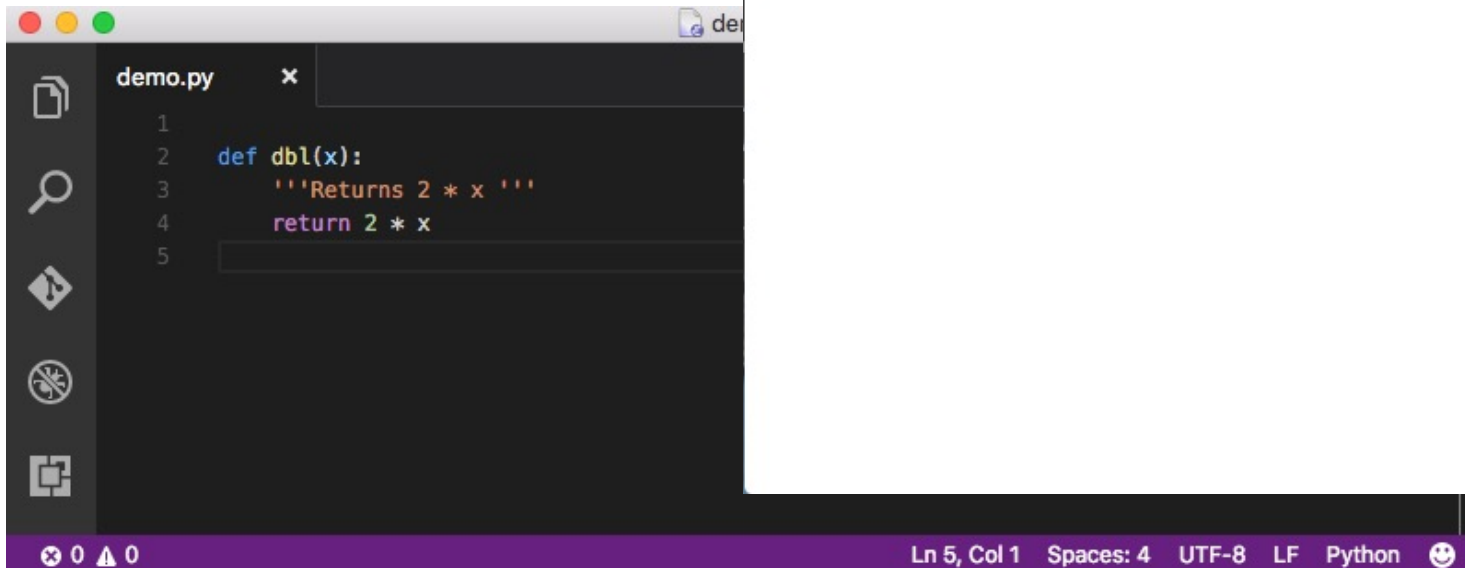
```
# Doubling program  
# Author:  Jessica  
# Date:   September 42
```

```
def dbl(x):  
    """This function takes a number x as input  
    and returns 2 * x"""  
    # Comments begin with a hash mark...  
    return 2 * x
```

Working cooperatively: editor and shell

Python shell

Visual studio code editor



The image shows two overlapping windows. The background window is the Visual Studio Code editor, displaying a file named `demo.py`. The code in the editor is:

```
1
2 def dbl(x):
3     '''Returns 2 * x'''
4     return 2 * x
5
```

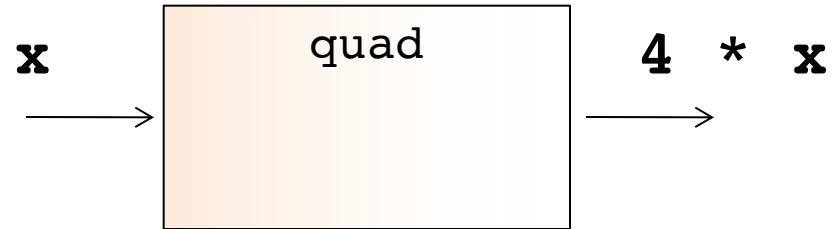
The foreground window is a terminal titled "Desktop — python2.7 — 80x24". It shows the command `python -i demo.py` being executed. The terminal output is:

```
bush@DHCP-36-125:Desktop$ python -i demo.py
>>> dbl(5)
10
>>>
```

The status bar at the bottom of the Visual Studio Code window indicates "Ln 5, Col 1", "Spaces: 4", "UTF-8", "LF", and "Python".

Functions calling functions

```
def quad(x):  
    return 4 * x
```



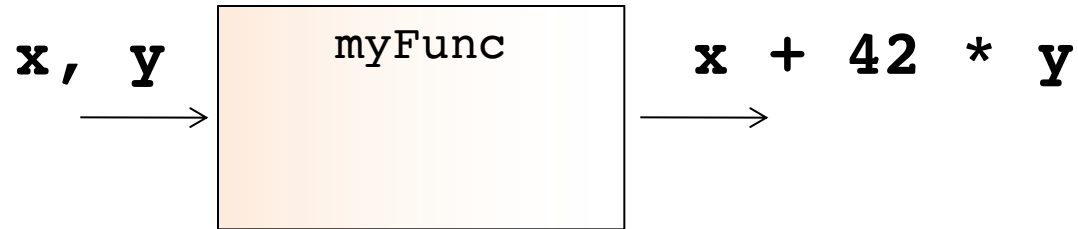
```
def quad(x):  
    return dbl(dbl(x))
```



Doubly cool!



Multiple inputs...



```
# my_func
```

```
# Author:  Jessica Wu
```

```
# Date:   September 42
```

```
def myFunc(x, y):  
    """returns x + 42 * y"""  
    return x + 42 * y
```

Mapping with Python...

```
def dbl(x):  
    '''returns 2 * x'''  
    return 2 * x
```

```
>>> list( map(dbl, [0, 1, 2, 3, 4, 5]) )  
[0, 2, 4, 6, 8, 10]
```

```
def evens(n):  
    my_range = range(n)  
    doubled = list( map(dbl, my_range) )  
    return doubled
```

We should really have docstrings here!



...or alternatively

```
def evens(n):  
    return list( map(dbl, range(n)) )
```

reduce-ing with Python...

```
def add(x, y):  
    '''returns x + y'''  
    return x + y
```

```
>>> reduce(add, [1, 2, 3, 4])  
10
```

add
= 3

add
= 6

add
= 10

To get reduce in python3, type
`from functools import reduce`



Try this...



1. Using map and/or reduce, write a python function called `gauss` that takes as input a positive integer `N` and returns the sum $1 + 2 + \dots + N$. You can assume you have `add`.
2. Using map and/or reduce, write a python function called `sum_of_squares` that takes as input a positive integer `N` and returns the sum $1^2 + 2^2 + 3^2 + \dots + N^2$



You can write extra
“helper” functions too!



Now try this...

Using map and/or reduce, write a function called `span` that returns the difference between the maximum and minimum numbers in a list...

```
>>> span([3, 1, 42, 7])
```

```
41
```

```
>>> span([42, 42, 42, 42])
```

```
0
```

```
min(x, y)
```

```
max(x, y)
```

These are built-in to Python!