get set up for today's workshop

Take a brief poll before we get started

Poll: **bit.ly/python923**

Introduction to Python

NaLette Brodnax

Indiana University Bloomington

School of Public & Environmental Affairs

Department of Political Science

Center of Excellence for Women in Technology

CEWiT

CENTER OF EXCELLENCE
FOR WOMEN IN TECHNOLOGY

We started a conversation about women in technology, ensuring all women have a seat at the table in every technology venture.

CEWiT addresses the global need to increase participation of women at all stages of their involvement in technology related fields.

Faculty, staff, alumnae and student alliances hold events, host professional seminars, and give IU women opportunities to build a community.

CONNECT WITH US

iucewit @IU_CEWiT iucewit

cewit.indiana.edu

my goals

- Demystify programming
- Help you write some code you can use

your goals

Please take the quick poll at: bit.ly/python923

what we'll cover today

1

getting set up

2

programming basics

3

working with files

what is python?

Python is a general purpose programming language.

It is easy to learn, highly readable, powerful and flexible.

when should we use python?

Programming should help you be more **efficient**

part 1: getting started

Go to www.python.org

getting the tools

Interpreter → Output

Text Editor + Interpreter → Output

Command Line + Text Editor + Interpreter → Output

Integrated Development Environment (IDE) → Output

installing the tools on your machine

Download Python 3.5
https://www.python.org/downloads/
→Includes IDLE, an IDE with a text editor and interpreter
→Includes pip, Python's standard package manager

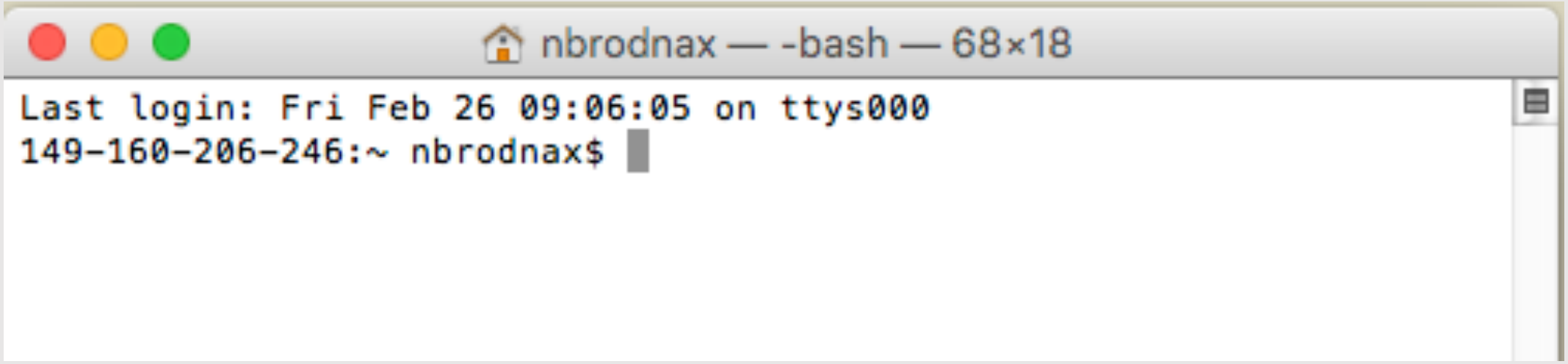
Install the necessary libraries **from the command line**:

```
$ pip3 install --upgrade pip
$ pip3 install requests
$ pip3 install beautifulsoup4
```

python development environment

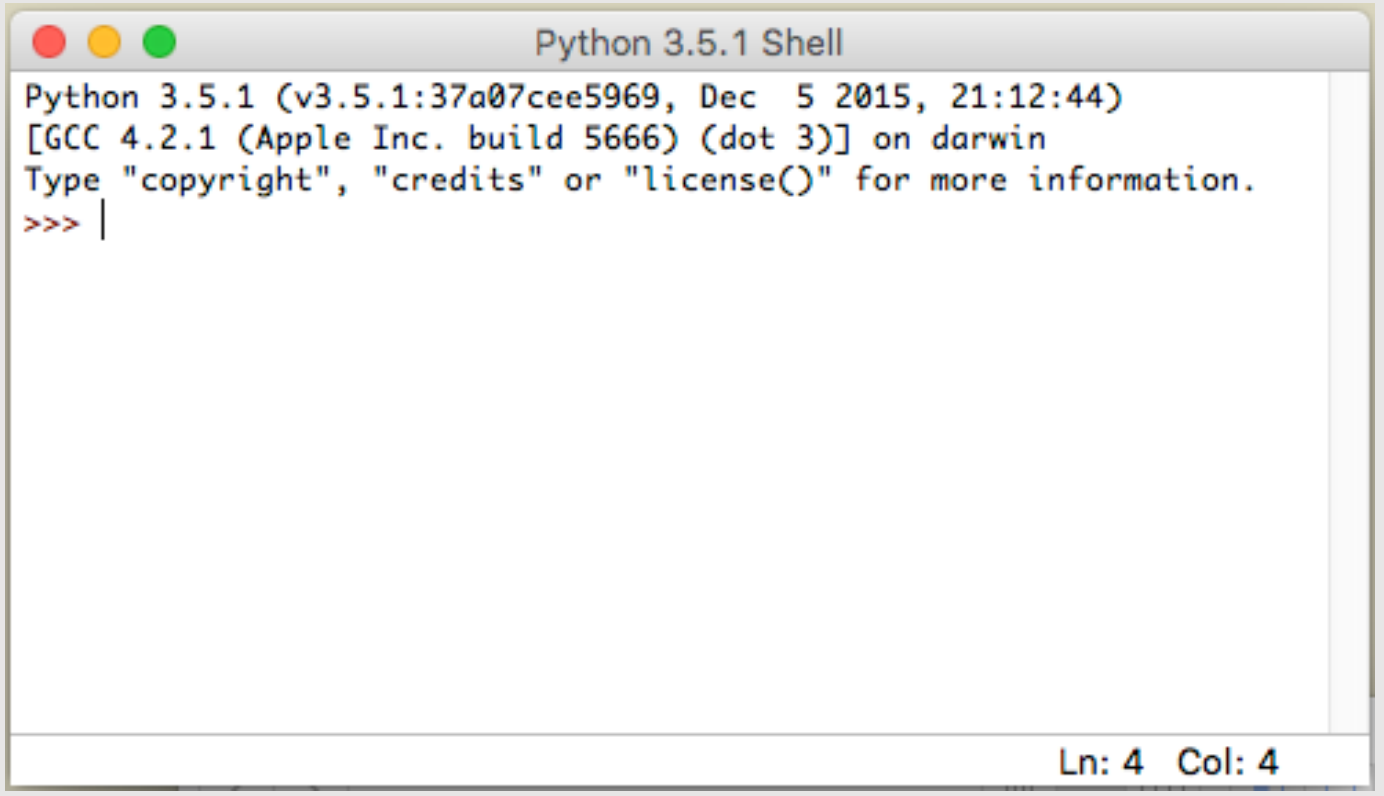
Command Line → **Terminal (Bash)/Powershell**

- Interact with computer's operating system
- Manage Python installation
- Access the Python Interpreter
- Execute a program without the Interpreter



python development environment

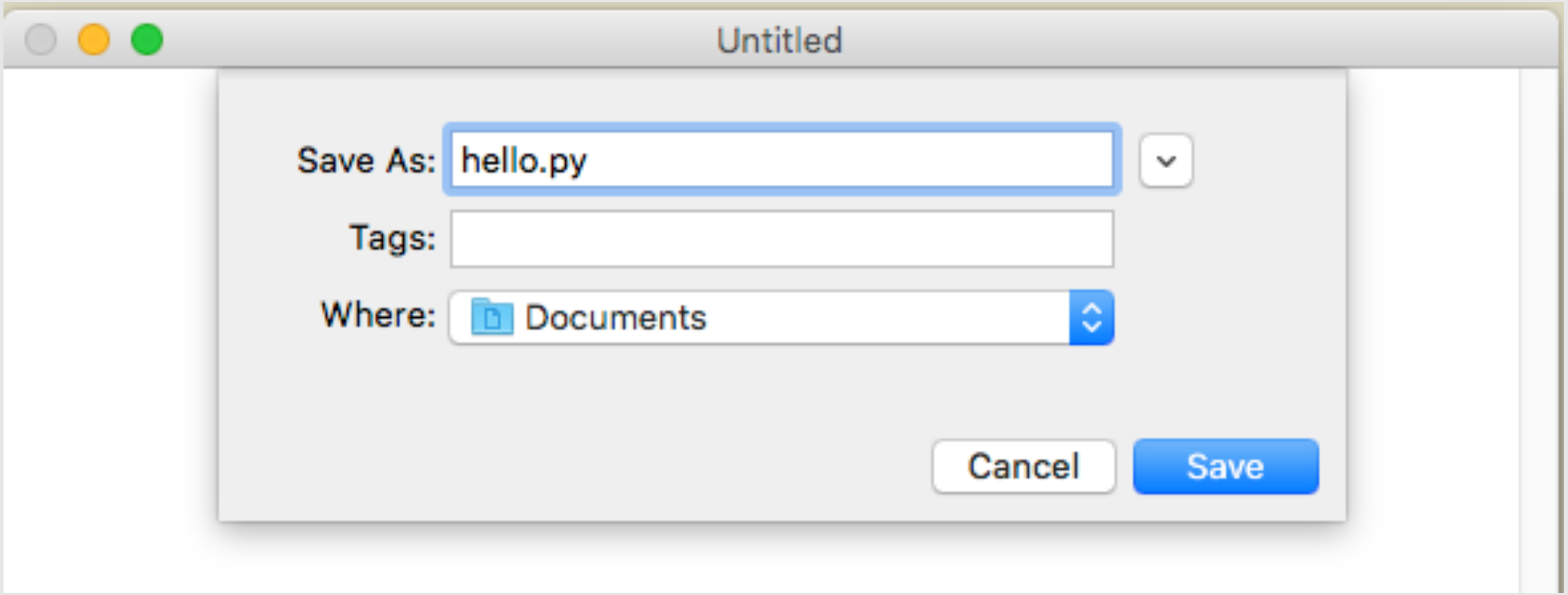
IDE → **IDLE**



Interact with Python

Write programs: **File → New File**

create your first script



create your first script

```
print(“Hello, world.”)
```

Save the program/script.
Run the program in the interpreter: **F5** or **Run → Run Module**

Optional: Download all the code for today’s workshop
From the Command Line:

```
git clone https://github.com/nmbrodnax/wim_python.git
```

a few programming language features

1. **Data types** – categories for storing different kinds of information in memory

2. **Conditionals** – control structures that allow decision making within a program

3. **Loops** – control structures that allow repeated behavior within a program

4. **Functions** – blocks of commands that can be reused

data types: sequences

String—ordered sequence of characters

```
‘happy’
```

List—ordered sequence of items

```
[‘Leia’, ‘Rey’, ‘Maz’]
```

Dictionary—unordered sequence of key-value pairs

```
{‘name’:‘Kylo’, ‘side’:‘dark’}
```

referencing sequences

Reference by index number, starting with zero

```
mystring = ‘happy’
print(mystring[0])
print(mystring[2:4])

mylist = [‘Leia’, ‘Rey’, ‘Maz’]
print(mylist[-1])
```

Reference by key

```
mydict = {‘name’:‘Kylo’, ‘side’:‘dark’}
print(mydict[‘name’])
```

conditionals

4-space **indentation** is very important in Python. In this case, it tells Python what to execute if the condition is true.

```
name = ‘Grace Hopper’

if len(name) < 20:
    print(‘Yes’)
else:
    print(‘No’)
```

equal	==	not equal	!=
greater than	>	greater than/equal	>=
less than	<	less than/equal	<=

loops

colon

```
name = ‘Grace Hopper’

i = 0
for letter in name:
    if letter in [‘a’,‘e’,‘i’,‘o’,‘u’]:
        i = i + 1
print(name + ‘ has ’ + str(i) + ‘ vowels.’)
```

indentation

convert the integer i to a string in order to concatenate it with other strings

loops

```
name = ‘Grace Hopper’

i = 0
vowel_count = 0
while i < len(name):
    if name[i] in [‘a’,‘e’,‘i’,‘o’,‘u’]:
        vowel_count = vowel_count + 1
    i = i + 1
print(name + ‘ has’ + str(vowel_count) + ‘ vowels.’)
```

functions v. methods

Function-–named block of code that can accept any number of arguments

```
my_string = ‘aBcDe’
print(my_string)
```

Method—a function with a built-in parameter for the object being acted on

```
print(my_string.lower())
```

writing functions

```
def function_name(argument1, argument2, ...):
    first command
    second command
    return output
```

```
def say_hello(name_string):
    print('Hello, ' + str(name_string) + '!')
    return None

say_hello('NaLette')
```

part 2: working with files

But first, let's take a 5-min break

what do we want to do?

1. Ask the user for a text filename: `input()`

   Note: `input()` accepts and returns a string

2. Ask the user for the number of lines of text to display

3. Convert the number of lines to an integer

4. Display that number of lines: `print()`

Practice text file: **kipling_jungle_book.txt**
https://github.com/nmbrodnax/wim_python

user input

displayed to the user

```
print("What file should I read from?")
filename = input("> ")

print("How many lines should I read?")
lines_to_read = input("> ")
```

prompt

store input in variable

reading files using **open()**

You must close any file that has been opened

```
line_counter = 0

file = open(filename, 'r')
while line_counter < int(lines_to_read):
    print(file.readline())
    line_counter = line_counter + 1
file.close()
```

reading files using **with open()**

```
with open(filename, 'r') as file:
     while line_counter < int(lines_to_read):
          print(file.readline())
          line_counter += 1
print(str(line_counter) + " lines read\n")
```

The **with** clause automatically closes the file

working with CSV files

1. Count the number of words in each line
   User-defined function: `count_words()`

2. Find length of the longest word
   User-defined function: `longest_word_length()`

3. Record the line number, word count, and length of the longest word in a CSV file

writing a function

function name

argument

```
def count_words(mytext):
    """Returns the number of words in a string
    str -> int"""
    words = mytext.split(" ")
    return len(words)
```

return value

another function

```
def longest_word_length(mytext):
"""Returns the average word length in a string
    str -> int"""
    words = mytext.split(" ")
    word_lengths = []
    for word in words:
        word_lengths.append(len(word))
    return max(word_lengths)
```

modules

Import statements allow you to add functions

```
import csv

with open("workshop.csv", 'w') as csvfile, \
     open(filename, 'r') as txtfile:
    writer = csv.writer(csvfile)
```

a “writer” object

writing files

```
with open("workshop.csv", 'w') as csvfile, \
     open(filename, 'r') as txtfile:
    writer = csv.writer(csvfile)
    line_counter = 0
    while line_counter < int(lines_to_read):
        line_number = line_counter + 1
        content = txtfile.readline()
        word_count = count_words(content)
        longest_word = longest_word_length(content)
        writer.writerow([line_number, word_count, longest_word])
        line_counter += 1
```

next up:

Introduction to Web Scraping with Python
Friday, September 30th

Introducing to Using APIs with Python
~~Friday, November 4th~~
Friday, November 11th

Thank you!

email: nbrodnax@indiana.edu

linkedin: nalettebrodnax
github: nmbrodnax
twitter: @nbrodnax