### An introduction to scientific data analysis, principles of programming, and Python

### watch by Tuesday, October 6, 2020 | Lesson #1

## Why code?

OCEAN 215 | Autumn 2020

Ethan Campbell and Katy Christensen

# What we'll cover in this lesson

### 1. The power of scientific data analysis

- 2. Fundamental principles of programming languages
- 3. Why do we use Python in this course?
- 4. Different ways of using Python

# What can data tell us?



- **Example:** Arctic Ocean sea ice thickness in August from 1979 to 2020
- What can we learn from this data?
  - Ice has become thinner
  - Ice is usually thickest near Greenland
  - lce thickness changes a lot year-to-year, but the patterns are usually similar
- You can't do this in Excel!



### What can data tell us?

### More examples from oceanography and marine science



GPS tracks of marine mammals and seabirds show habitat importance around Antarctica (Hindell et al. 2020)







### What can data tell us?

### More examples from oceanography and marine science



Oxygen isotopes measured in forams show how slower ocean mixing prolonged ice ages

(Hasenfrantz et al. 2019)

Echosounder profiles underneath Arctic sea ice reveal fluctuations in temperature and salinity layers (<u>Shibley et al. 2020</u>)





## Who uses data?

- It's not just oceanography! Every area of earth science research uses data...
  - Satellite data to estimate increases in **harvested forest area** and its impact on soil erosion
  - Seismograms to predict large earthquakes in real-time by identifying foreshocks
  - Hydrometric gauge measurements to show how **river flooding** is affected by climate change
  - *Emissions, air quality, and population data* to relate **air pollution** to human mortality
  - Aircraft remote sensing to identify super-emitting methane sources from landfills, agriculture, and the oil/gas sector
  - Spacecraft measurements to understand how solar plasma ejections impact Earth's atmosphere
- And it's not just earth science research! Data science touches every aspect of everyday life, such as:

  - Sports (predicting successful **basketball shots** based on *a player's performance metrics*)

\* If you're curious about how data can be used in service of racial justice, check out this list of resources and this article in PNAS. \*\* This is a project that I'm currently working on!

• Public health (tracking new COVID-19 outbreaks by looking for keywords in millions of *internet searches*) **Transportation** (updating **driving directions** on-the-fly based on traffic jams inferred from *phone data*)

Social justice\* (assessing racial disparities in police stops of cyclists in Seattle from court infraction records\*\*)



## The modern scientific method



# The power of programming is its versatility

A common – but unnecessarily complicated – workflow using specialized software:



### Programming enables you to make a computer do anything you want, in a unified workflow:







# What we'll cover in this lesson

- 1. The power of scientific data analysis
- 2. Fundamental principles of programming languages
- 3. Why do we use Python in this course?
- 4. Different ways of using Python

# Programming requires a human and a computer





Ada Lovelace (1815-1852)

Modern computers have:

- Processors that can perform a billion calculations per second
- Memory that can store hundreds to thousands of gigabytes (GB) of results
- Input/output devices to take human instructions and display results



### First computer

("Turing complete," i.e. computationally universal)

**The Analytical Machine** 

calculations per second ousands of gigabytes (GB) of results nstructions and display results



# Aspects of languages

**Syntax** describes valid combinations of symbols, words, and phrases:

- English: Cat dog boy → not syntactically valid Cat hugs boy -> syntactically valid
- programming language: "hi″!"hello" → not syntactically valid 3\*5  $\rightarrow$  syntactically valid

**Semantics** gives meaning to a syntactically valid phrase:

- English: Cat hugs boy → syntactically valid but semantic error
- programming language:  $3 + "hi" \rightarrow$  syntactically valid

but semantic error

**Python says:** 

"SyntaxError: invalid syntax"

**Python says:** 

"TypeError: unsupported operand type for +"



# Where things go wrong

### **Syntax errors:**

• Common and easily caught; program won't run

### **Semantic errors:**

- Some languages check for these before running a program, but some (including Python) don't
- Can cause unpredictable behavior

### No semantic errors but **different meaning than what the** programmer intended:

- Program crashes and stops running, or...
- Program runs forever, or...
- Program gives an answer, but different than expected З.



"hi"!"hello"

**Python code:** 



print(a)

a + "hi"

a = 3



#### **Python code:**

print("jello")

` program will run despite programmer intending for it to print "hello"

# Types of programming languages

### **Machine language:**

File that

you run

Your

code

Your code =

file that you run

Compiler

Interpreter

- This is what the computer understands
- Very difficult and error-prone to write

### **Compiled languages** (e.g. C, Fortran, Java):

- Full control over computer, runs very fast
- Programmer sometimes must manage memory manually
- Not easy to read or write

### Interpreted languages (e.g. Python, R, Matlab):

- Less control over computer, runs slightly slower
- Memory management is usually handled automatically
- "Expressive" (syntax closer to human language)  $\rightarrow$  easy to read and write

#### **Assembly language:**

MESSAGE, A1 LEA MOVE.B #14,DO #15 #9,DO TRAP #15 MESSAGE DC.B 'HELLO!' START END

#### **C:**

```
#include <stdio.h>
int main()
    printf("Hello!")
    return 0;
```

### **Python:**

print("Hello!")

,0	
);	

# What we'll cover in this lesson

- 1. The power of scientific data analysis
- 2. Fundamental principles of programming languages

### 3. Why do we use Python in this course?

- 4. Different ways of using Python

### "There are only two kinds of programming languages: the ones people complain about and the ones nobody uses."

**Three different** programming languages:



- Bjarne Stroustrup (the inventor of C++)

![](_page_14_Picture_6.jpeg)

![](_page_14_Picture_7.jpeg)

# But why Python? It is...

- **Free!** (unlike Matlab)
- Open source (unlike Matlab)
  - That means you're not dependent on a company to fix bugs Large user community constantly working to improve language
- Old, so it's very stable (Python was created in 1991)
- General-purpose
- Incredibly popular in all areas of science
- Incredibly popular outside of science, too
- Easy to teach and learn

# Python's rising popularity

![](_page_16_Figure_1.jpeg)

# What we'll cover in this lesson

- 1. The power of scientific data analysis
- 2. Fundamental principles of programming languages
- 3. Why do we use Python in this course?
- 4. Different ways of using Python

# Python versions

#### Python 2

- Released in 2000
- Very few good reasons to use it anymore
- Print statements look like this: print "Hello"
- Some people still use it, but we won't

#### Python 3

- Released in 2009; latest version is 3.7
- Incredibly similar syntax to Python 2, but different enough to not be backwards compatible
- Print statements look like this: print ("Hello")
- This is what we will use

# Different ways to code in Python

Type of Python code:

#### Interactive Python (**IPython**) shell

```
>>> print("Hello")
Hello
>>> print(3)
3
```

# Mac/Windows application:

### **Command line** (MacOS Terminal or Windows Command Prompt)

![](_page_19_Picture_6.jpeg)

![](_page_19_Picture_7.jpeg)

![](_page_19_Figure_8.jpeg)

#### Integrated development environment (IDE)

3.6)								
Projects Tools View Help								
🕨 🗭 州 端 🚝 🚝 🂓 📕 陋 💥 🎤 🍨 🔶 🔶 C:\Users\TestUser\Downloads	16740156	56420fb678bb5ba67c3ee3aae4-551407feca17b0f67bb9	f85687f4db8d1b953	678\16740156420fb678bb5ba67c	3ee3aae4-551407	feca 17b0f67bb9f85687f4db8d 1b953678 🛛 🗸 🏲 🛧		
stUser\Downloads\16740156420fb678bb5ba67c3ee3aae4-551407feca17b0f67bb9f85687f4db8d1b953678\16740156420fb6.	. Ø×	Outline B × v	ariable explorer			8 ×		
interpolation.py 🛛initpy 🖾umd_helper.py 🖾umd_main.py 🖾README.md 🔝	Q,	i = t × # ↔	± B % Ø 0					
	^	Y 👼 interpolation.py	Name	Туре	Size	Value		
pylab		Generate data for analysis	array_int8	int8	(2, 3)	Min: -7		
umpy import cos, linspace, pi, sin, random		Plot results	needer udet 22		(2 2 2)	Min: 1		
scipy.interpolate import spiprep, spiev		0	array_utitoz	411(52	(2, 2, 3)	Max: 7		
Generate data for analysis		0	bars	container.BarContainer	20	BarContainer object of matplotlib.conta.		
		e e e e e e e e e e e e e e e e e e e	df	DataFrame	(3, 2)	Column names: bools, ints		
ascending spiral in 3-space		۰.	filename	str	1	C:\ProgramData\Anaconda3\lib\site-packa		
nspace(0, 1.75 * 2 * p1, 100)		> 🤪 imputerNan	list test	list	2	[Dataframe, Numpy array]		
.n(t)	_ 1	Y 🖲 Queue	-	int	1	344		
us(t)		init		20				
		pop	r	T108164	1	7.611082589334796 Min: 0.4983036638535687		
noise			radii	float64	(20,)	Max: 9.856848974942551		
andom.normal(scale=0.1, size=x.shape)	_	with open(data_path + output_file_n	region	tuple	2	(slice, slice)		
random.normal(scale=0.1, size=y.shape)	- 1	with open(data_path + output_file_n	rgb	float64	(45, 45, 4)	Min: 0.0 Nax: 1.0		
andom.normal(scale=0.1, size=z.snape)		with open(data_path + output_file_n	series	Series	(1,)	Series object of pandas.core.series mod		
		print_file	test none	NoneType	1	NoneType object of builtins module		
'erform calculations		<ul> <li>Example Bitzer class</li> <li>O</li> </ul>	Variable explorer	Help File explorer Find in	n files Breakpo	onts Static code analysis Profiler Online help		
ne narameters	- 1	✓	Python console			8 ×		
iness = 3.0 # Smoothness parameter	- 1	einit	Console 1/A	Console 2/A 🖸 Custon	n Name/A 🖸	00:34:13 🔳 🍠 🔅		
m = 2 # Spline order		prepare_dataset				^		
= -1 # Estimate of number of knots needed (-1 = maximal)		✓ ⊛ Serie	: 1s •	<ul> <li>LightSource(270, 45</li> </ul>	)			
		something		o use a custom milish the pack colors of t	aarng mode, ba shadad s	override the built-in shading		
the knot points		Generation	: rgb	= ls.shade(z, cmap=c	m.gist_eart	h, vert_exag=0.1, blend_mode='soft')		
ornes, u = sprprep([x, y, z], s=smoornness, k=k_param, nests=-1)		✓ ● foo	: surf	<pre>F = ax.plot_surface(x</pre>	, y, z, rst	ride=1, cstride=1, facecolors=rgb,		
uate spline, including interpolated points		_init		1	inewidth-0,	antialiased-False, shade-False)		
<pre>ynew, znew = splev(linspace(0, 1, 400), knot_points)</pre>	- 1	@ spam	plt.	.show()				
		with open(file) as f:						
	- 1	0				90°		
lot results	- 1	0			13	450		
. Rewrite to quaid code small	- 1				1.			
subplot(2, 2, 1)		ě		700	· /			
= pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')	- 1	Base		650		6 8		
pylab.plot(xnew, ynew, 'r-', label='Fit with X-Y Cross Section')	- 1	Derived		550	180°	2 0		
legend()	- 1			450				
xlabel('x')		for r, bar in zip(radii, bars):	1943	400	/			
ylabel('y')	- E   I	with np.load(filename) as dem:		36.73				
			-84.41	36.71	22	25* 315*		
<pre>subplot(2, 2, 2) = m(lab plot(x, z, 'bo-' label='Data with X-7 (nors Section')</pre>	-		-84	-84.38 36.70				
= pyido.piot(x, 2, 00-, iduei= bala with x-2 cross Section)						270°		
legend()			[n [12]-					
xlabel('x')	_ ~		un [44]:			· · · · · · · · · · · · · · · · · · ·		
	>		IPython console	History log Internal console	-			
		Permissions:	RW End-of-lin	nes: LF Encoding: UTF-8	Line: 26	Column: 4 Memory: 49 % CPU: 15 %		

### Jupyter notebook

#### **Internet browser** (Chrome, Safari, Firefox, etc.)

File	Edit	View Ir	nsert C	Cell I	Kernel Hel	p					
In [6]:	# 3	upyter Noteb	ook Chro	me Ext	ension						
In [7]:	impo impo	ort numpy as ort pandas a	np s pd								
Im [8]:	pd.:	read_csv("./	titanic.	csv")							
		Passengerid	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
	0	1	0	3	Braund, Hr. Owen Harris	nale	22.0	1	0	A/5 21171	7.2500
	1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th	fenale	38.0	1	0	PC 17599	71.2833
	2	3	1	3	Heikkinen, Miss. Laina	fenale	26.0	0	0	STON/02. 3101282	7.9250
	3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	fenale	35.0	1	0	113803	53.1000
					Allen, Mr.						0.0500

![](_page_19_Picture_16.jpeg)

# Jupyter vs. Google Colab notebooks

### Where is the code run?

Your computer ("the local machine")

![](_page_20_Picture_3.jpeg)

### How to access them?

- 1. Install Jupyter
- 2. Open command line app (Terminal on Macs,
- 3. Type "jupyter notebook," which will start a local server
- 4. Open internet browser
- 5. Navigate to server address
- 1. Open internet browser
- 2. Navigate to: <u>colab.research.google.com</u>

### Jupyter notebooks

![](_page_20_Picture_13.jpeg)

![](_page_20_Picture_14.jpeg)

Google's servers ("the cloud")

![](_page_20_Picture_16.jpeg)

Command Prompt on PCs)

### Advantages (+) and disadvantages (-)

#### • (-) Some setup required

- (+) No internet connection required
- (+) Code runs fast if your computer is fast
- (-) Code runs slow if your computer is slow
- (+) Bonus features, customizability, ability to install any package, etc.
- (+) Free

#### • (+) No setup required

- (-) Requires internet connection
- (+/-) Code runs decently fast but not blazingly fast
- (-) Less customizability, more difficult package management
- (+/-) Free, as long as Google says it's free
- (+) Google Drive integration; easy to share

![](_page_20_Picture_31.jpeg)

## Texts used to create this lesson + useful resources

- Princeton University Computer Science: An Interdisciplinary Approach, <u>Lecture 1</u>
- MIT OpenCourseware Introduction to Computer Science and Programming in Python, Lecture 1
- Rutgers University Introduction to Computer Science, Lectures <u>1</u> and <u>2</u>
- CU Boulder Introduction to Earth Data Science, Lecture 1
- Johnny Wei-Bing Lin <u>A Hands-On Introduction to Using Python in the Atmospheric and Oceanic Sciences</u>
- Monash University <u>Introduction to Data Science</u>
- Ryan Abernathey and Kerry Key <u>An Introduction to Earth and Environmental Data Science</u>
- Jake VanderPlas <u>Python Data Science Handbook</u>
- Johnny Wei-Bing Lin Bulletin of the American Meteorological Society "Why Python Is the Next Wave in Earth Sciences Computing" Tom Waterman – Medium – "Why Python is better than R for Data Science careers"
- Ada Lovelace image: <u>https://www.computerhistory.org/babbage/adalovelace/</u>
- Analytical Machine image: <u>https://en.wikipedia.org/wiki/File:AnalyticalMachine\_Babbage\_London.jpg</u>

![](_page_21_Picture_13.jpeg)

![](_page_21_Picture_14.jpeg)