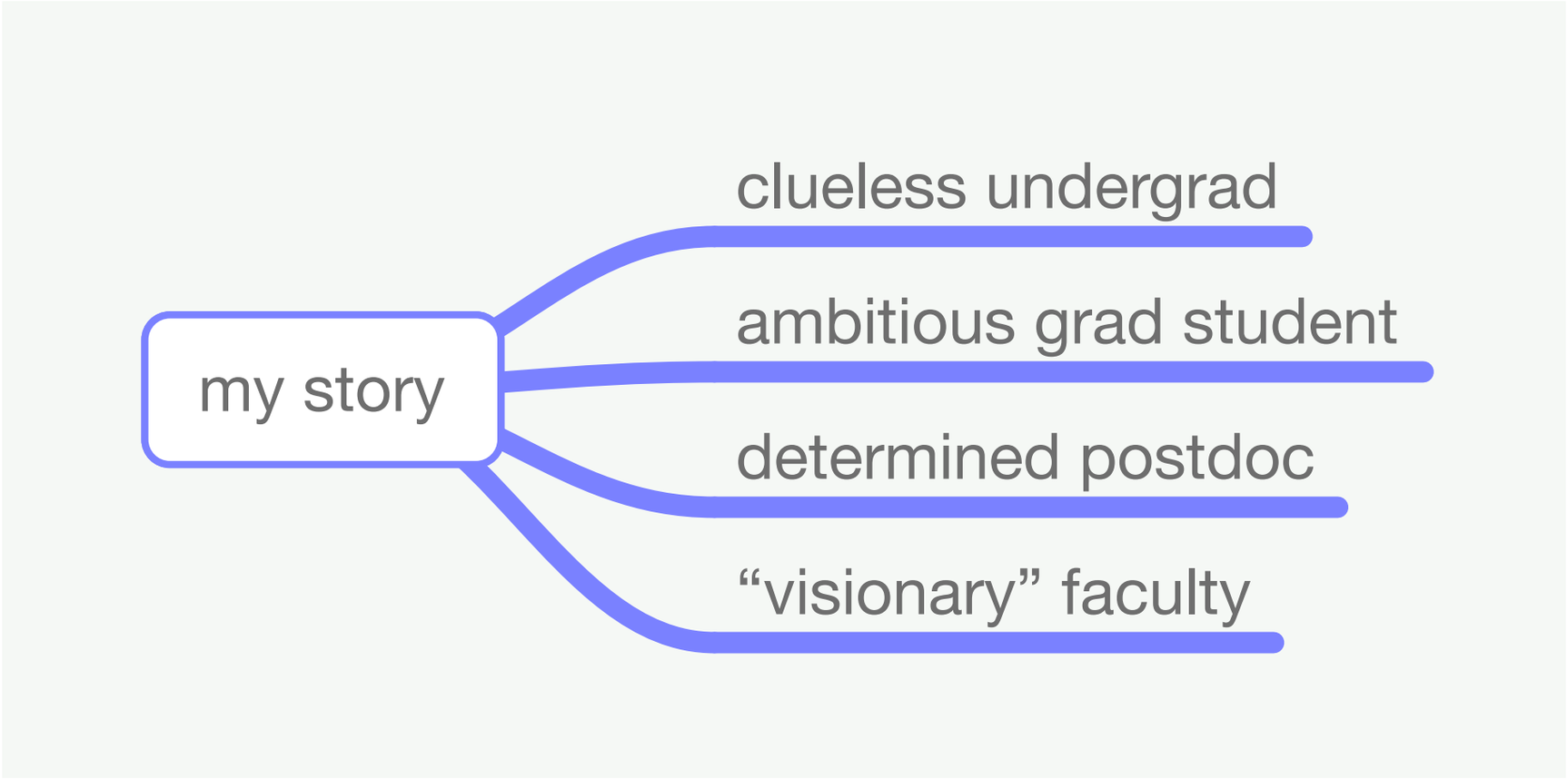
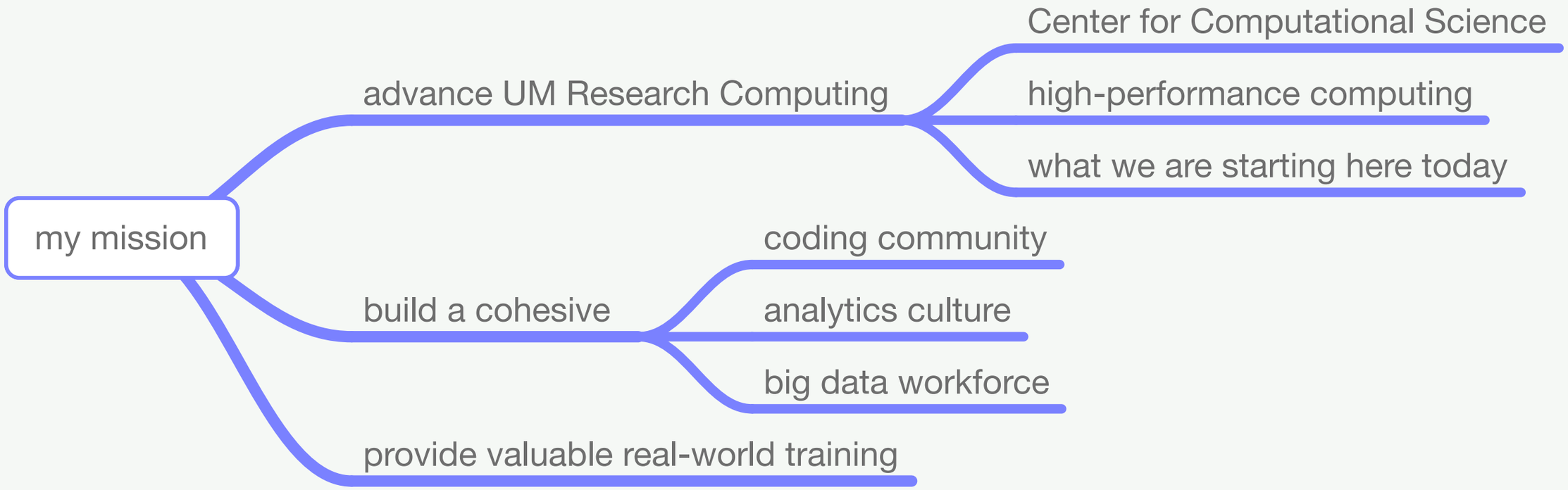
MCP 743
Introductory Python Programming for Bioscientists

Instructor: Dan Isom, Ph.D.

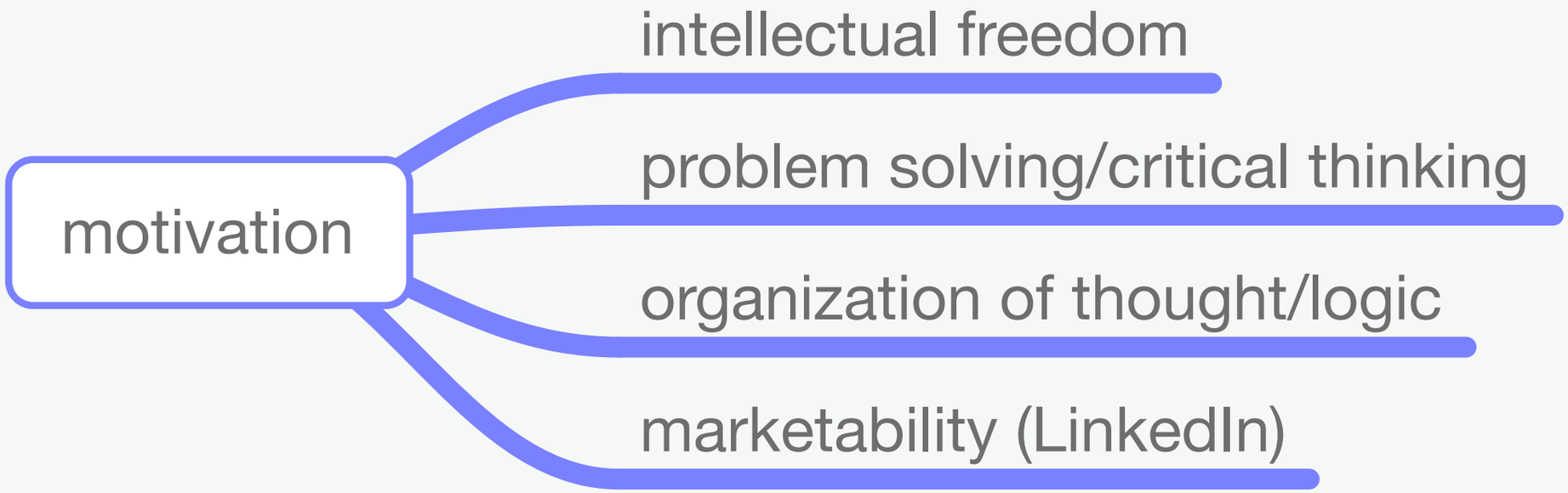
my personal coding story



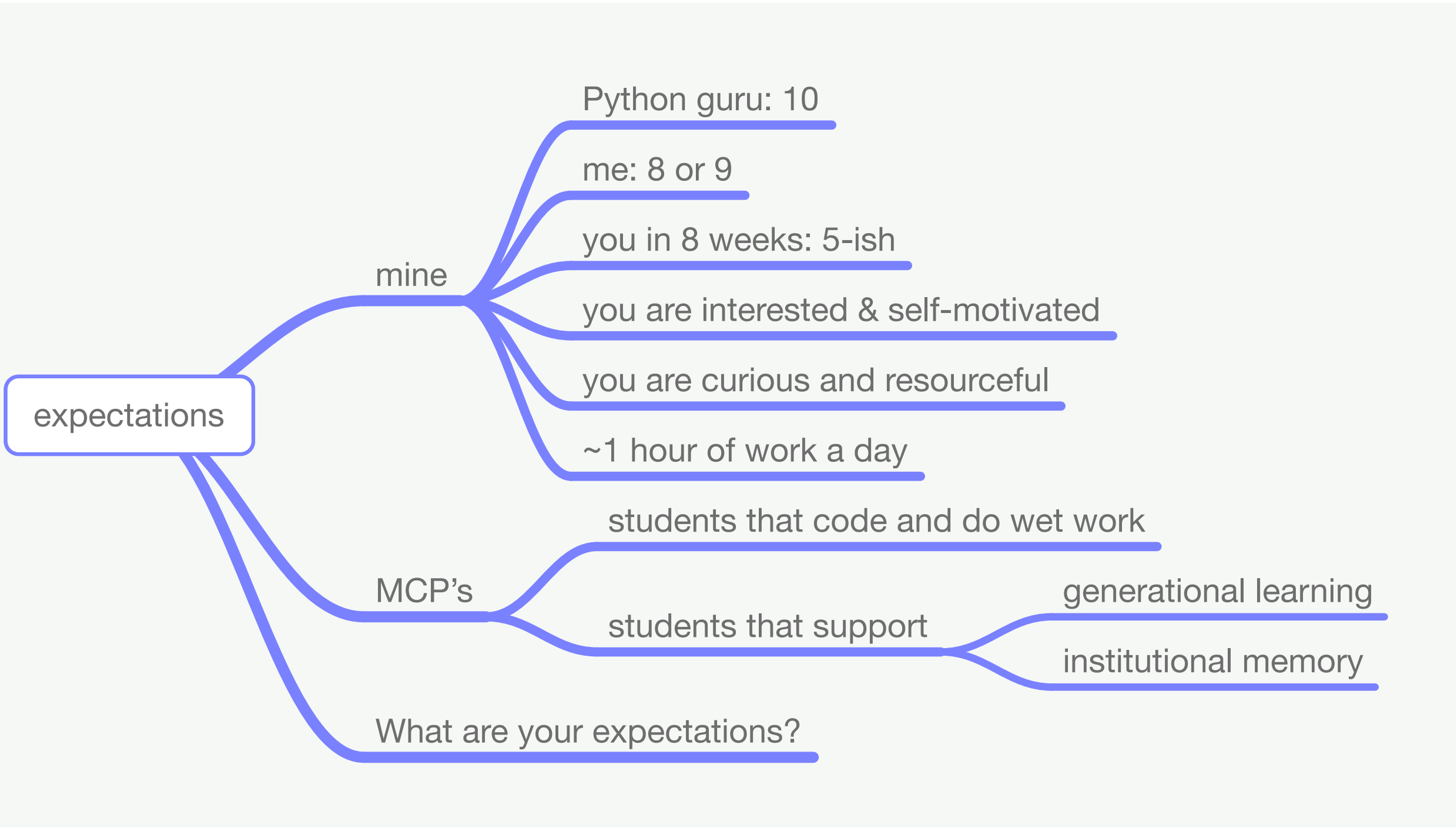
my mission


my mission
advance UM Research Computing
Center for Computational Science
high-performance computing
what we are starting here today
build a cohesive
coding community
analytics culture
big data workforce
provide valuable real-world training

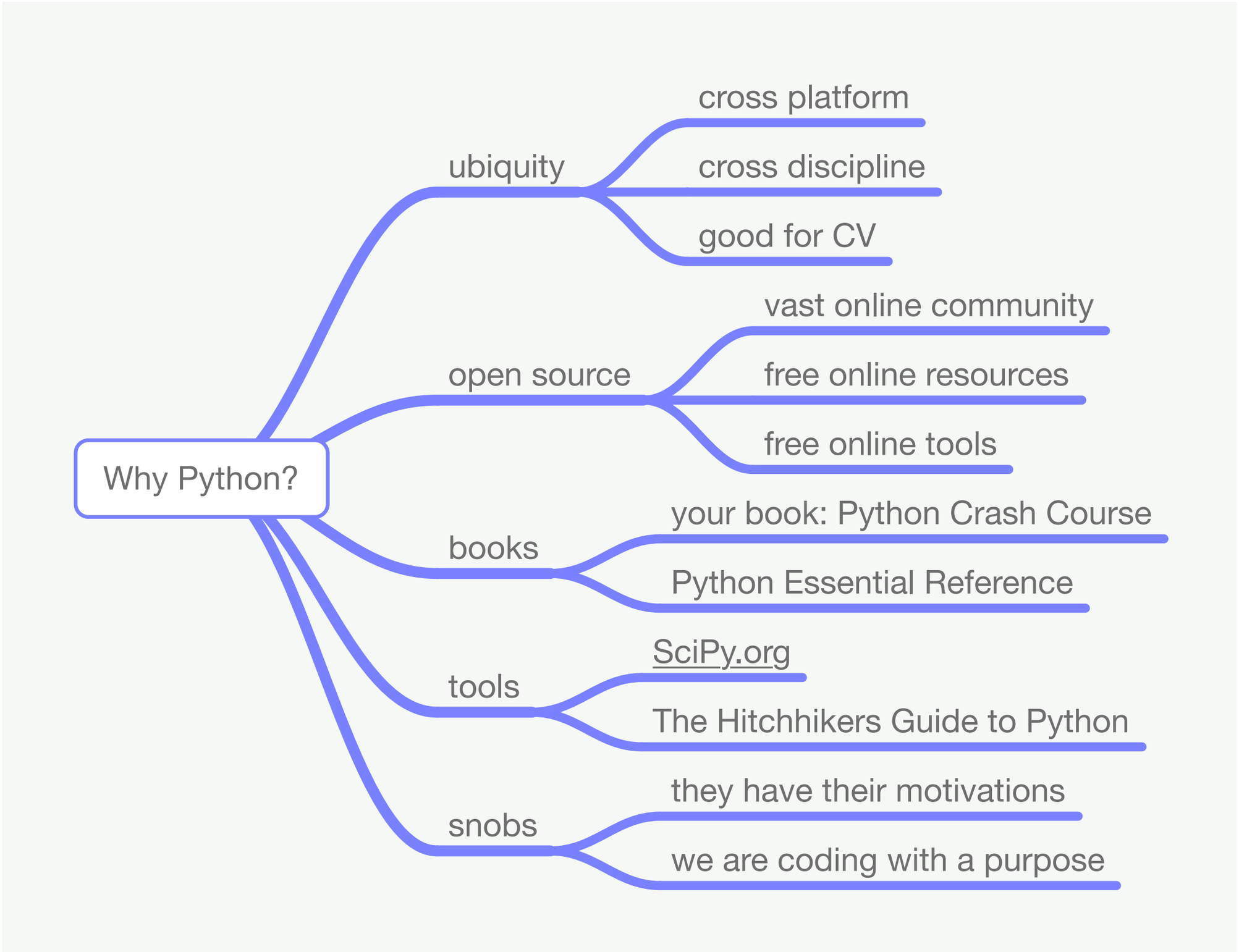
motivations for learning Python



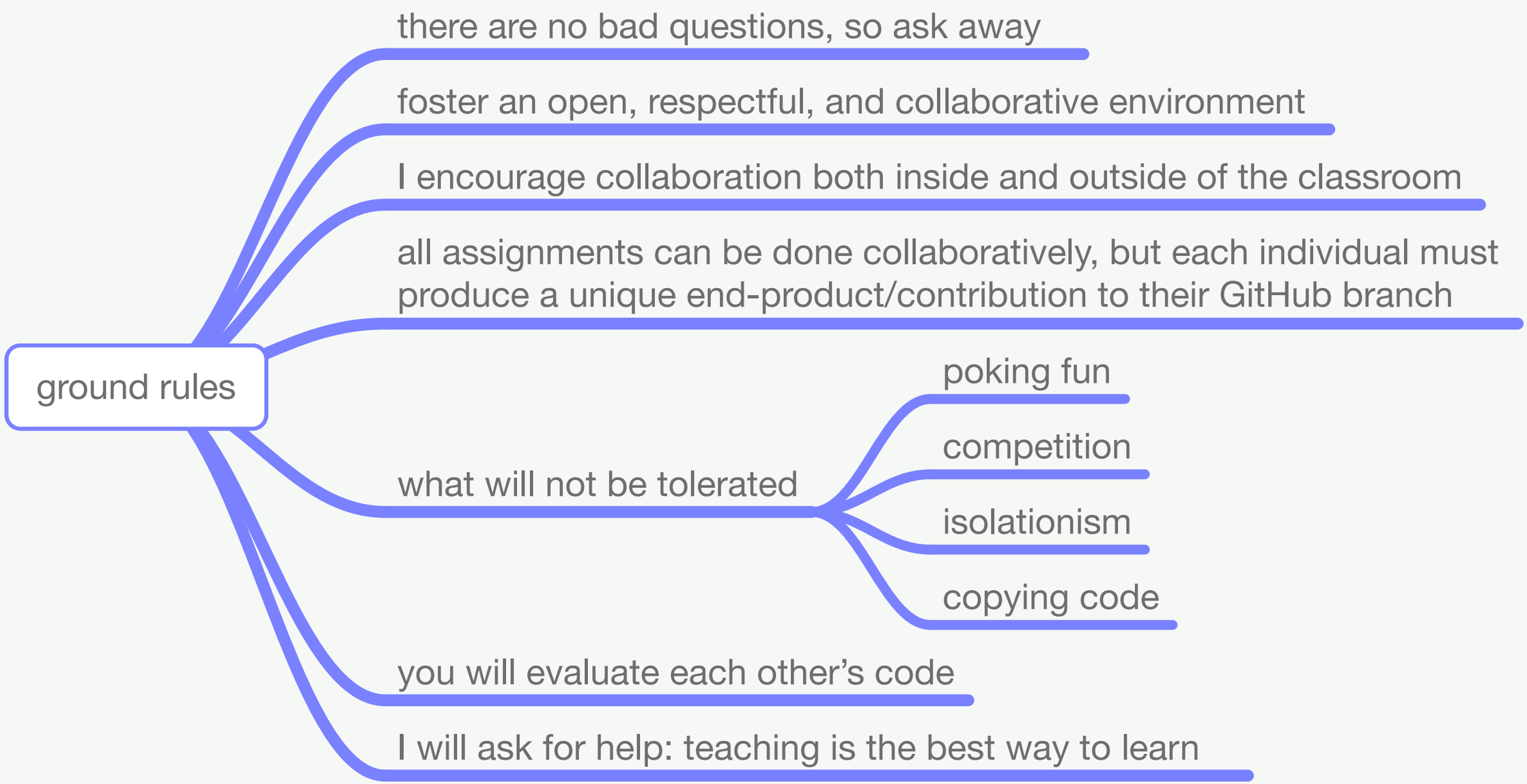
setting expectations



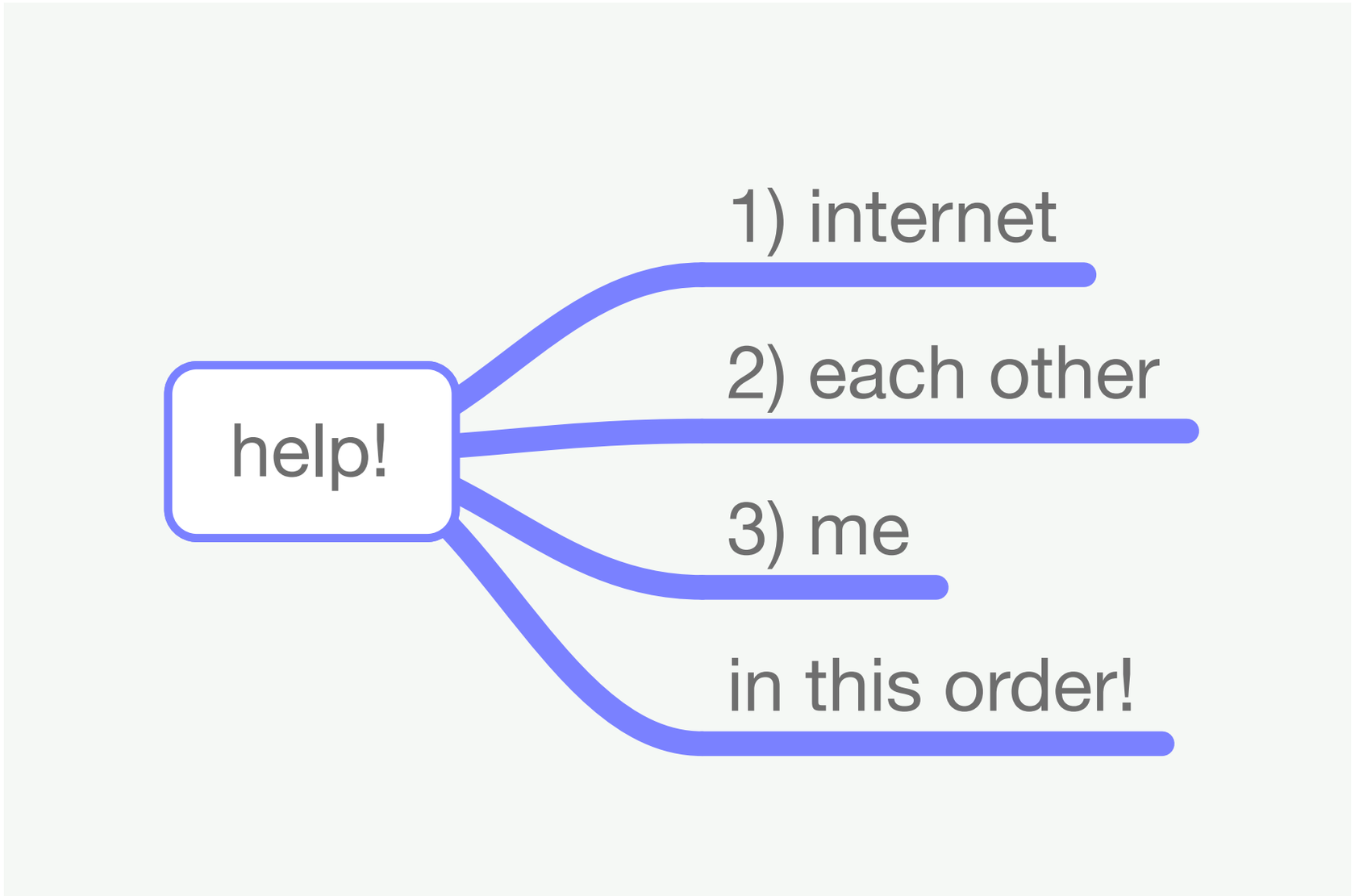
justification



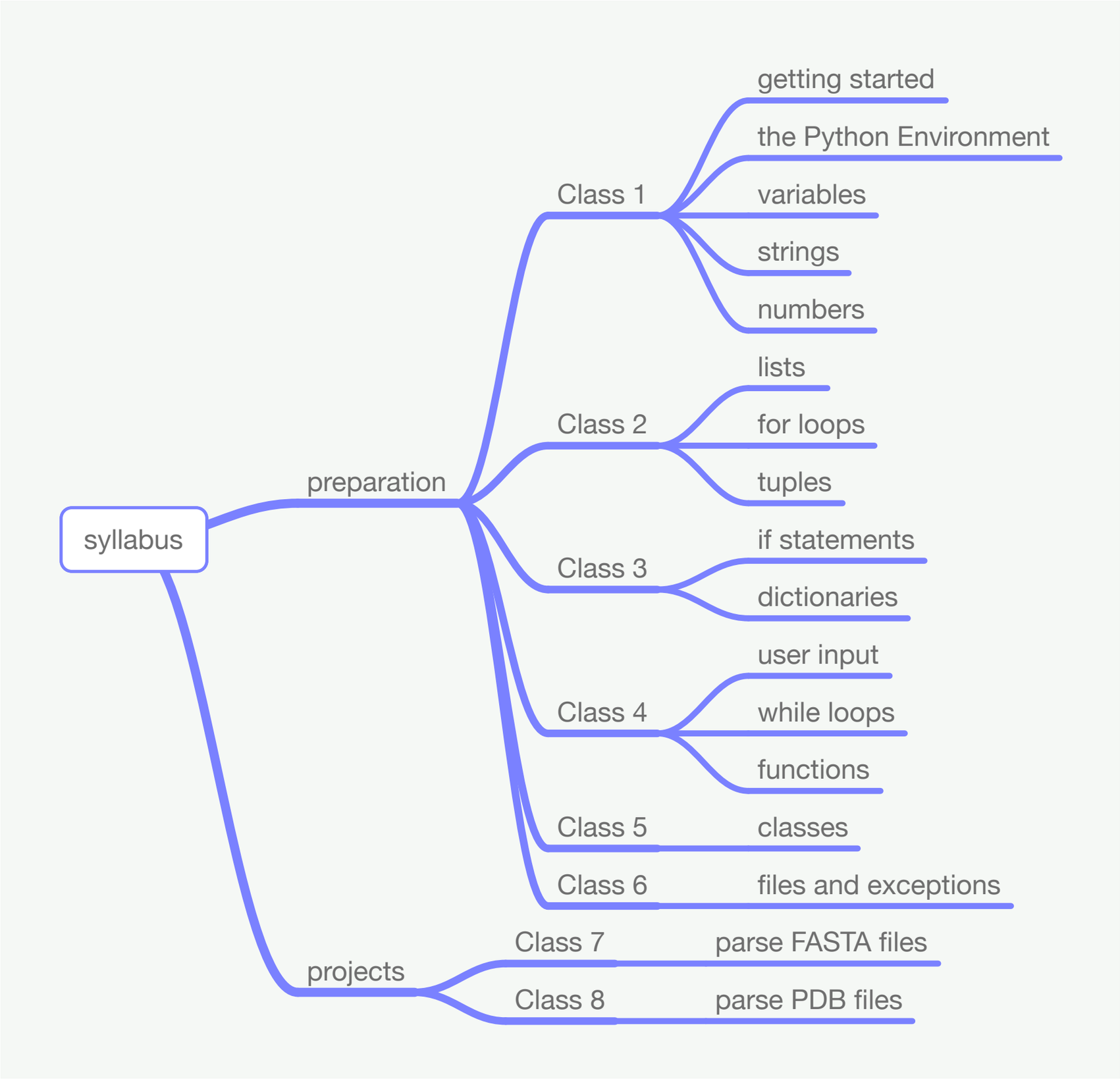
ground rules for class



when you get stuck and need help…



current syllabus



Is your Python environment working?

there may be multiple versions of Python on your computer

my default Python binary →

Apple's Python binaries →

```
Dans-iMac:/ danisom$ sudo find . -path "*bin/python*"
Password:
./Applications/Kivy.app/Contents/Resources/venv/bin/python
./Applications/Kivy.app/Contents/Resources/venv/bin/python2
./Applications/Kivy.app/Contents/Resources/venv/bin/python2.7
./Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX.sdk/usr/bin/python-config
find: ./dev/fd/3: Not a directory
find: ./dev/fd/4: Not a directory
./opt/local/bin/python
./opt/local/bin/python-config
./opt/local/bin/python2.7
./opt/local/bin/python2.7-config
./opt/local/bin/pythonw
./opt/local/bin/pythonw2.7
./opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin/python
./opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin/python-config
./opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin/python2
./opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin/python2-config
./opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7
./opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7-config
./opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin/pythonw
./opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin/pythonw2
./opt/local/Library/Frameworks/Python.framework/Versions/2.7/bin/pythonw2.7
./System/Library/Frameworks/Python.framework/Versions/2.6/bin/python
./System/Library/Frameworks/Python.framework/Versions/2.6/bin/python-config
./System/Library/Frameworks/Python.framework/Versions/2.6/bin/python2.6
./System/Library/Frameworks/Python.framework/Versions/2.6/bin/python2.6-config
./System/Library/Frameworks/Python.framework/Versions/2.6/bin/pythonw
./System/Library/Frameworks/Python.framework/Versions/2.6/bin/pythonw2.6
./System/Library/Frameworks/Python.framework/Versions/2.7/bin/python
./System/Library/Frameworks/Python.framework/Versions/2.7/bin/python-config
./System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2
./System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2-config
./System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7
./System/Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7-config
./System/Library/Frameworks/Python.framework/Versions/2.7/bin/pythonw
./System/Library/Frameworks/Python.framework/Versions/2.7/bin/pythonw2
./System/Library/Frameworks/Python.framework/Versions/2.7/bin/pythonw2.7
./usr/bin/python
./usr/bin/python-config
./usr/bin/python2.6
./usr/bin/python2.6-config
./usr/bin/python2.7
./usr/bin/python2.7-config
./usr/bin/pythonw
./usr/bin/pythonw2.6
./usr/bin/pythonw2.7
```

Note: text with grey background is output from my terminal session in OS X

there may be multiple versions of Python on your computer

my Mac as of this morning:

my default Python from macPorts

```
Dans-iMac:/ danisom$ python
Python 2.7.11 (default, Mar  1 2016, 18:40:10)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Apple’s latest Python (2.7)

```
Dans-iMac:/ danisom$ /usr/bin/python2.7
Python 2.7.10 (default, Jul 30 2016, 19:40:32)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Apple’s additional Python (2.6)

```
Dans-iMac:/ danisom$ /usr/bin/python2.6
Python 2.6.9 (unknown, Jul 30 2016, 19:40:24)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Note: text with grey background is output from my terminal session in OS X

how to identify which Python you are using by default

list the Python binaries on my computer

```
Dans-iMac:/ danisom$ which -a python
/opt/local/bin/python
/usr/bin/python
```

list the default Python binary

```
Dans-iMac:~ danisom$ which python
/opt/local/bin/python
```

Note: text with grey background is output from my terminal session in OS X

Python 3 vs. 2

most code is written in Python 2 (inertia)

the internet is dominated by Python 2

the print function as an example:

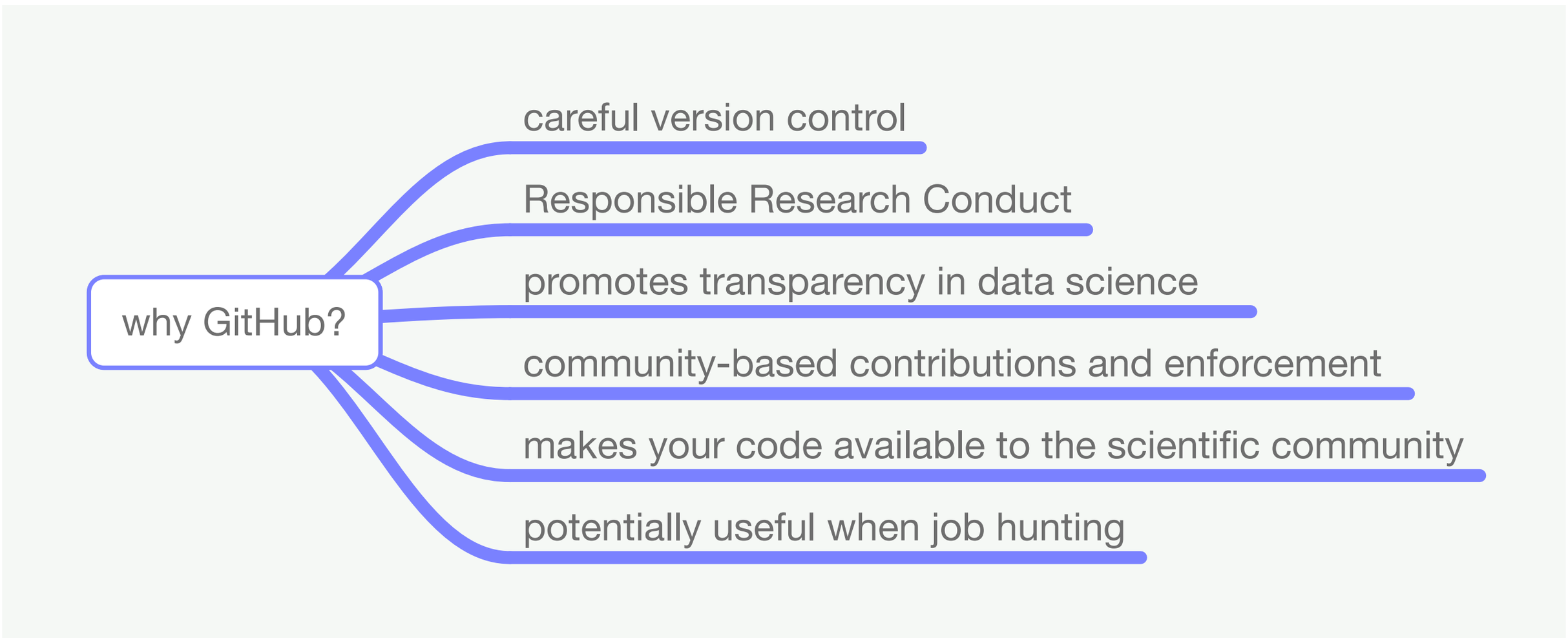
Python 3: print(“I wish Dan would where a white t-shirt”)
Python 2: print “Black t-shirts are a substitute for exercise”

throughout the class,
I will try to point out where there are key differences

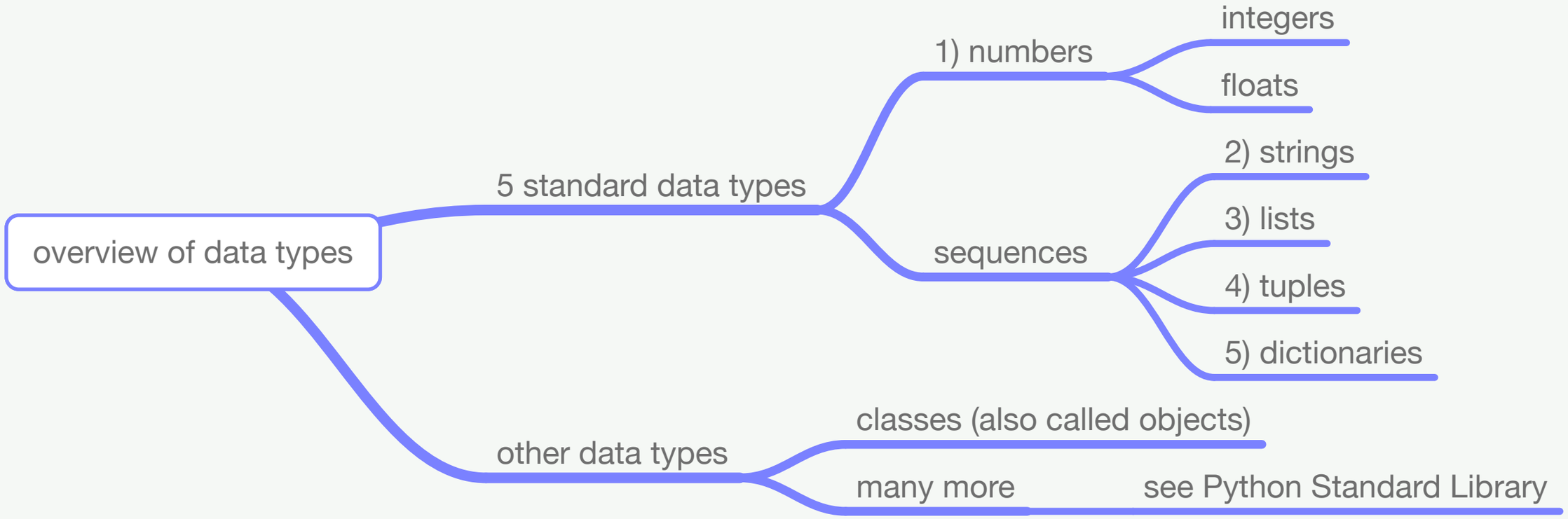
verify that your **Python** works using our **GitHub** repository

1. sign up for a GitHub account
2. download and install GitHub for desktop
3. goto the **UniversityOfMiamiCodingCommunity** repository (repo)
4. goto the **MCP-743-Class1** repo
5. clone the master branch
6. online, select the link to the helloWorld.py file
7. online, create a new branch for the helloWorld.py file
   (name it something like “Custom-hello-world-for-*YourFirstName*”
8. online, view the helloWorld.py file in GitHub desktop
9. Run the Python program from your terminal
10. Edit the helloWorld.py file to include something new to print
11. Name and commit the change as a new version of helloWorld.py
12. Sync Github desktop with your online account

Why GitHub?
it essentially serves as a computational lab notebook



overview of some Python types



#comments are not data types but are important

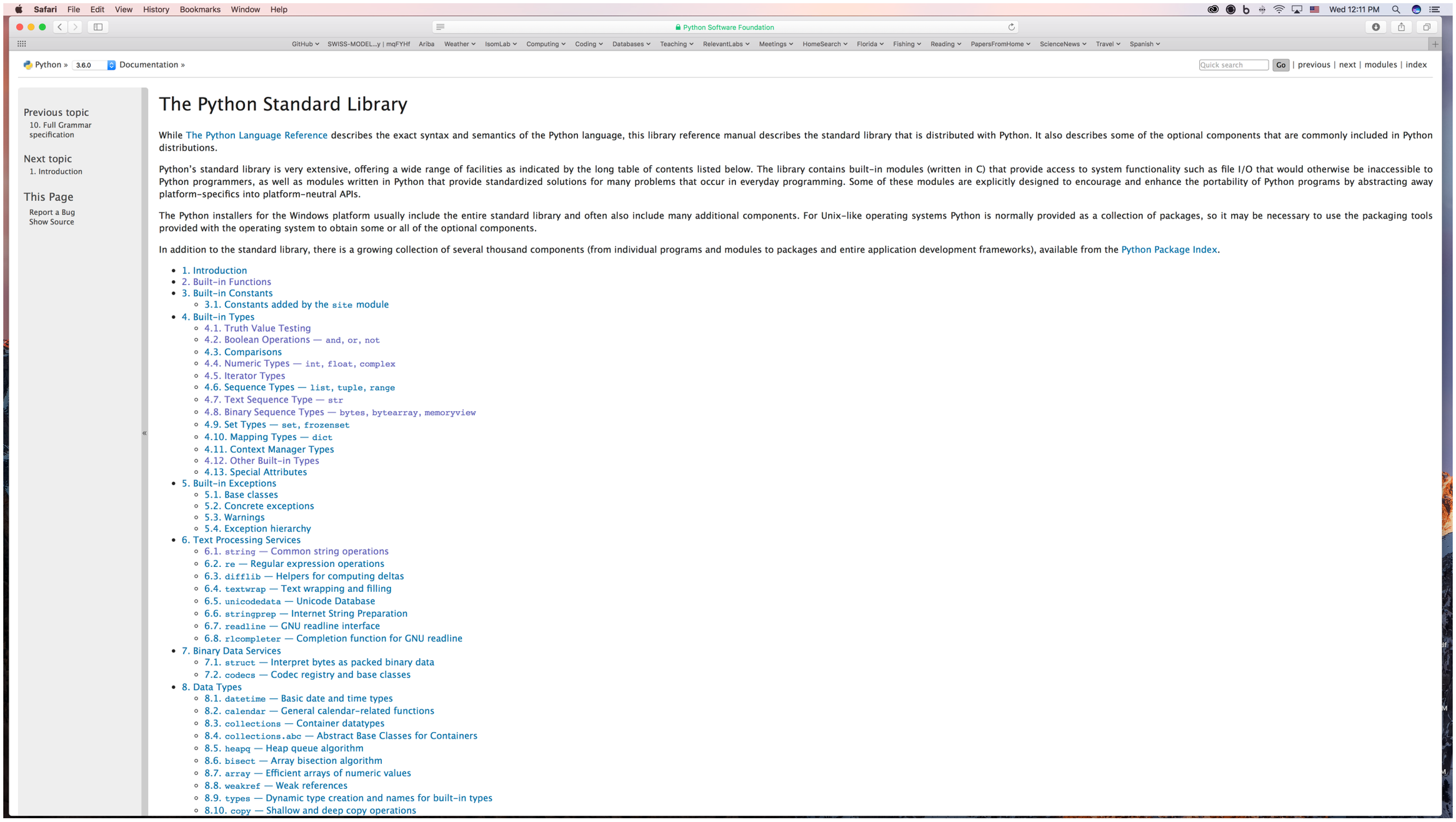
brief review: creating and assigning variables

examples

```
>>> anIntegerVariable = 1
>>>
>>> aFloatVariable = 100.0
>>>
>>> aStringVariable = "you're pathological"
>>>
>>> aListVariable = [1, 2, 3, 4, 5, 6]
>>>
>>> aTupleVariable= (1, 2, 3, 4, 5, 6)
>>>
>>> aDictionaryVariable = {"donald":1, "trump":2, "huge":3, "disaster":4, "wall":5}
```

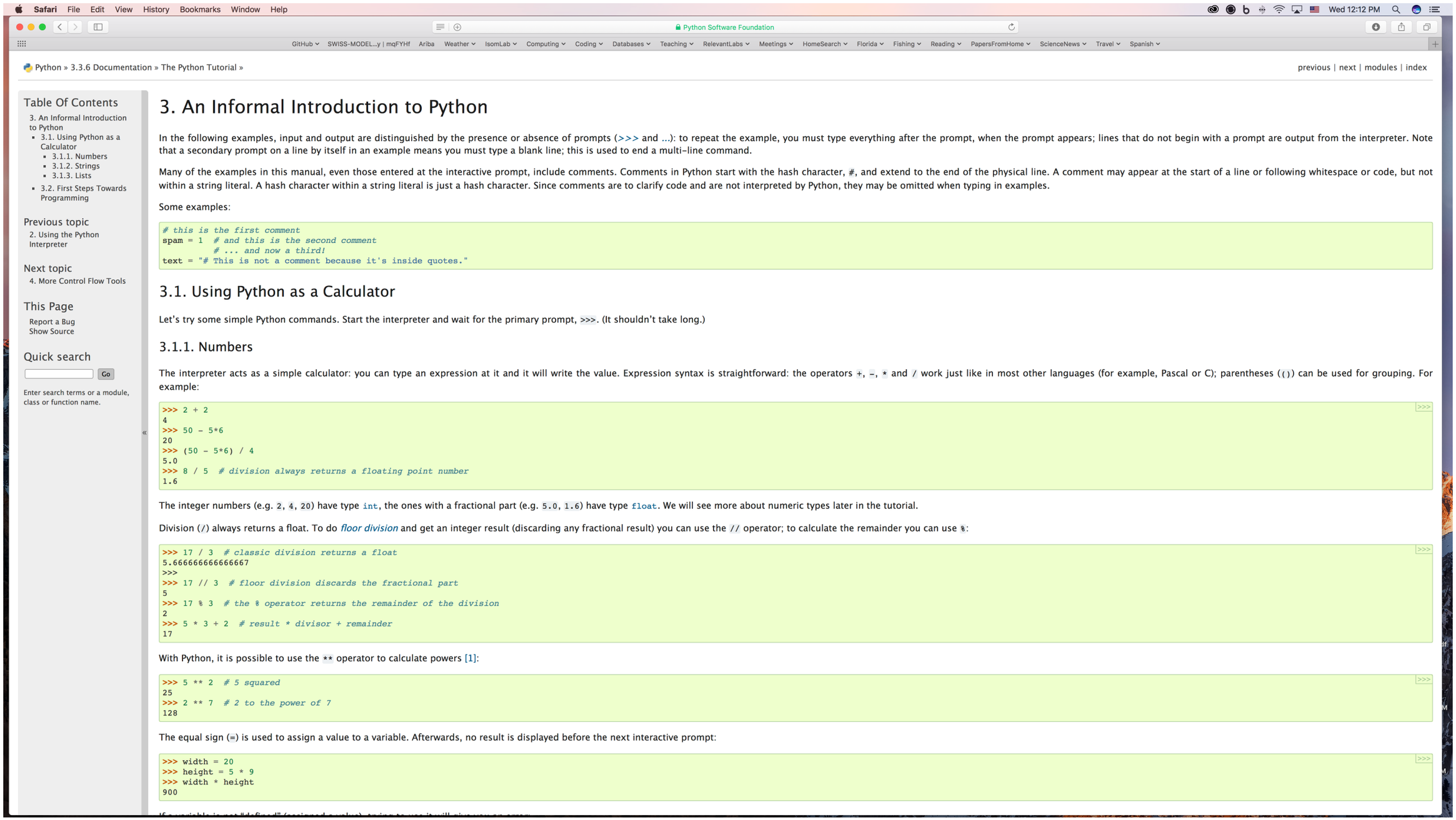
Note: text with grey background is output from my terminal session in OS X

go beyond your book: Python standard library



https://docs.python.org/3/library/index.html

go beyond your book: Python tutorial



Safari File Edit View History Bookmarks Window Help — Wed 12:12 PM

Python Software Foundation

GitHub · SWISS-MODEL...y | mqFYHf · Ariba · Weather · IsomLab · Computing · Coding · Databases · Teaching · RelevantLabs · Meetings · HomeSearch · Florida · Fishing · Reading · PapersFromHome · ScienceNews · Travel · Spanish

Python » 3.3.6 Documentation » The Python Tutorial »

previous | next | modules | index

Table Of Contents

3. An Informal Introduction to Python
- 3.1. Using Python as a Calculator
  - 3.1.1. Numbers
  - 3.1.2. Strings
  - 3.1.3. Lists
- 3.2. First Steps Towards Programming

Previous topic

2. Using the Python Interpreter

Next topic

4. More Control Flow Tools

This Page

Report a Bug
Show Source

Quick search

Go

Enter search terms or a module, class or function name.

3. An Informal Introduction to Python

In the following examples, input and output are distinguished by the presence or absence of prompts (>>> and ...): to repeat the example, you must type everything after the prompt, when the prompt appears; lines that do not begin with a prompt are output from the interpreter. Note that a secondary prompt on a line by itself in an example means you must type a blank line; this is used to end a multi-line command.

Many of the examples in this manual, even those entered at the interactive prompt, include comments. Comments in Python start with the hash character, #, and extend to the end of the physical line. A comment may appear at the start of a line or following whitespace or code, but not within a string literal. A hash character within a string literal is just a hash character. Since comments are to clarify code and are not interpreted by Python, they may be omitted when typing in examples.

Some examples:

```
# this is the first comment
spam = 1  # and this is the second comment
          # ... and now a third!
text = "# This is not a comment because it's inside quotes."
```

3.1. Using Python as a Calculator

Let's try some simple Python commands. Start the interpreter and wait for the primary prompt, >>>. (It shouldn't take long.)

3.1.1. Numbers

The interpreter acts as a simple calculator: you can type an expression at it and it will write the value. Expression syntax is straightforward: the operators +, -, * and / work just like in most other languages (for example, Pascal or C); parentheses (()) can be used for grouping. For example:

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # division always returns a floating point number
1.6
```

The integer numbers (e.g. 2, 4, 20) have type int, the ones with a fractional part (e.g. 5.0, 1.6) have type float. We will see more about numeric types later in the tutorial.

Division (/) always returns a float. To do *floor division* and get an integer result (discarding any fractional result) you can use the // operator; to calculate the remainder you can use %:

```
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>>
>>> 17 // 3  # floor division discards the fractional part
5
>>> 17 % 3  # the % operator returns the remainder of the division
2
>>> 5 * 3 + 2  # result * divisor + remainder
17
```

With Python, it is possible to use the ** operator to calculate powers [1]:

```
>>> 5 ** 2  # 5 squared
25
>>> 2 ** 7  # 2 to the power of 7
128
```

The equal sign (=) is used to assign a value to a variable. Afterwards, no result is displayed before the next interactive prompt:

```
>>> width = 20
>>> height = 5 * 9
>>> width * height
900
```

https://docs.python.org/3.3/tutorial/introduction.html

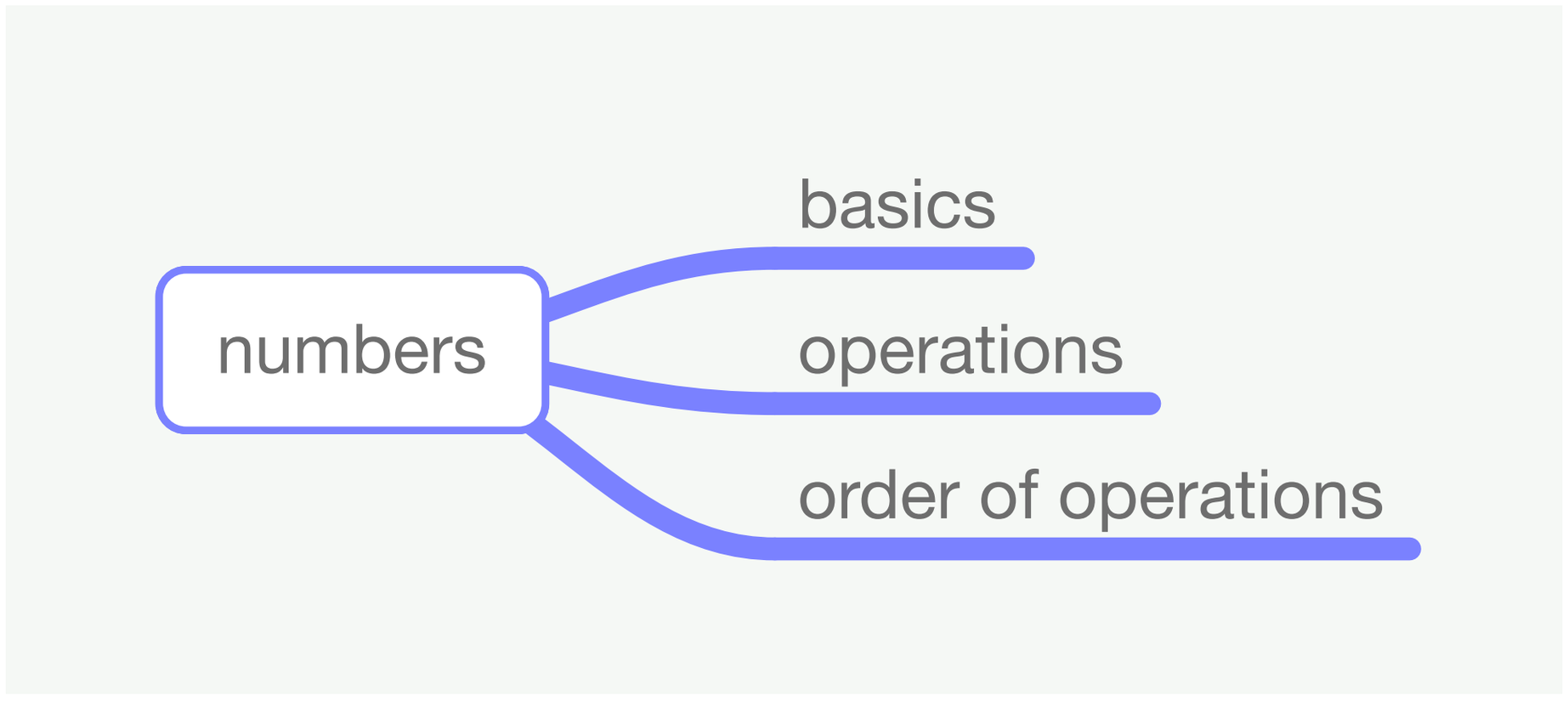
Python built-in functions

		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

Python keywords (avoid as variable names)

FALSE	class	finally	is	return
None	continue	for	lambda	try
TRUE	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

numbers



https://docs.python.org/3.3/tutorial/introduction.html#numbers

numbers Python 3 vs. 2

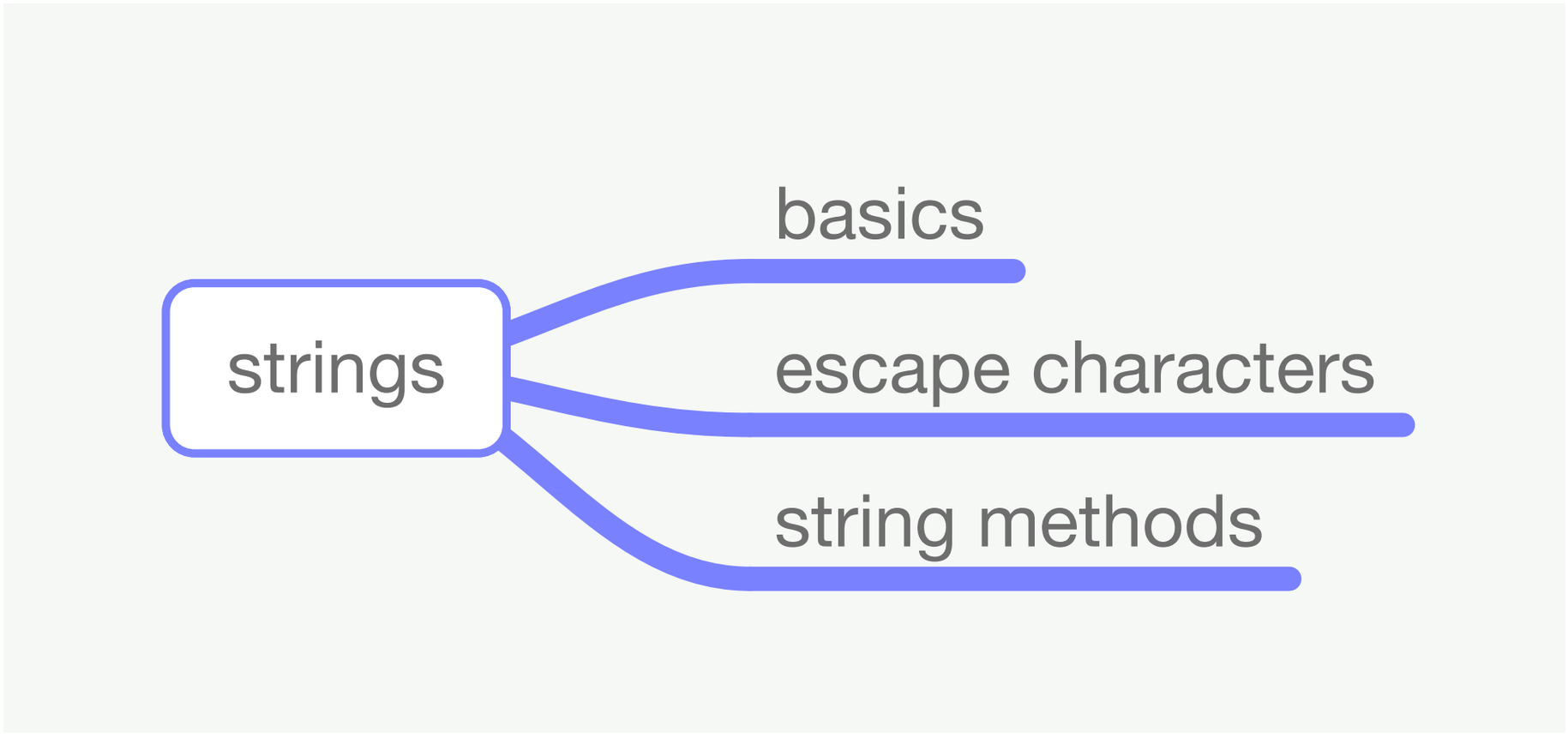
In Python 3
3 / 2 = 1.5

In Python 2
3 / 2 = 1

To get equivalent result in Python 2,
at least one number must be a float.
3.0 / 2 = 1.5

https://docs.python.org/3.3/tutorial/introduction.html#numbers

strings



https://docs.python.org/3.3/tutorial/introduction.html#strings

now I shall riff…

escape characters

Backslash notation	Hexadecimal character	Description
\a	0x07	Bell or alert
\b	0x08	Backspace
\cx		Control-x
\C-x		Control-x
\e	0x1b	Escape
\f	0x0c	Formfeed
\M-\C-x		Meta-Control-x
\n	0x0a	Newline
\nnn		Octal notation, where n is in the range 0.7
\r	0x0d	Carriage return
\s	0x20	Space
\t	0x09	Tab
\v	0x0b	Vertical tab
\x		Character x
\xnn		Hexadecimal notation, where n is in the range 0.9, a.f, or A.F

- all you really need to know

preparation for Class 2

- **Class 2**

  - **Assignment**
    - Read Chapters 3 & 4
    - Write 3 Programs
      - program1.py
        - In a single .py file submit 2 equations of your choice to GitHub
        - See my example on GitHub (pv=nRT)
      - program2.py
        - Create a list of integers
        - Loop over that list of integers
        - Print each integer
      - program3.py
        - Create a list of integers
        - Loop over the list of integers
        - Sum the integers and print the summed result

- **Topics**
  - Lists
  - For Loops
  - Tuples