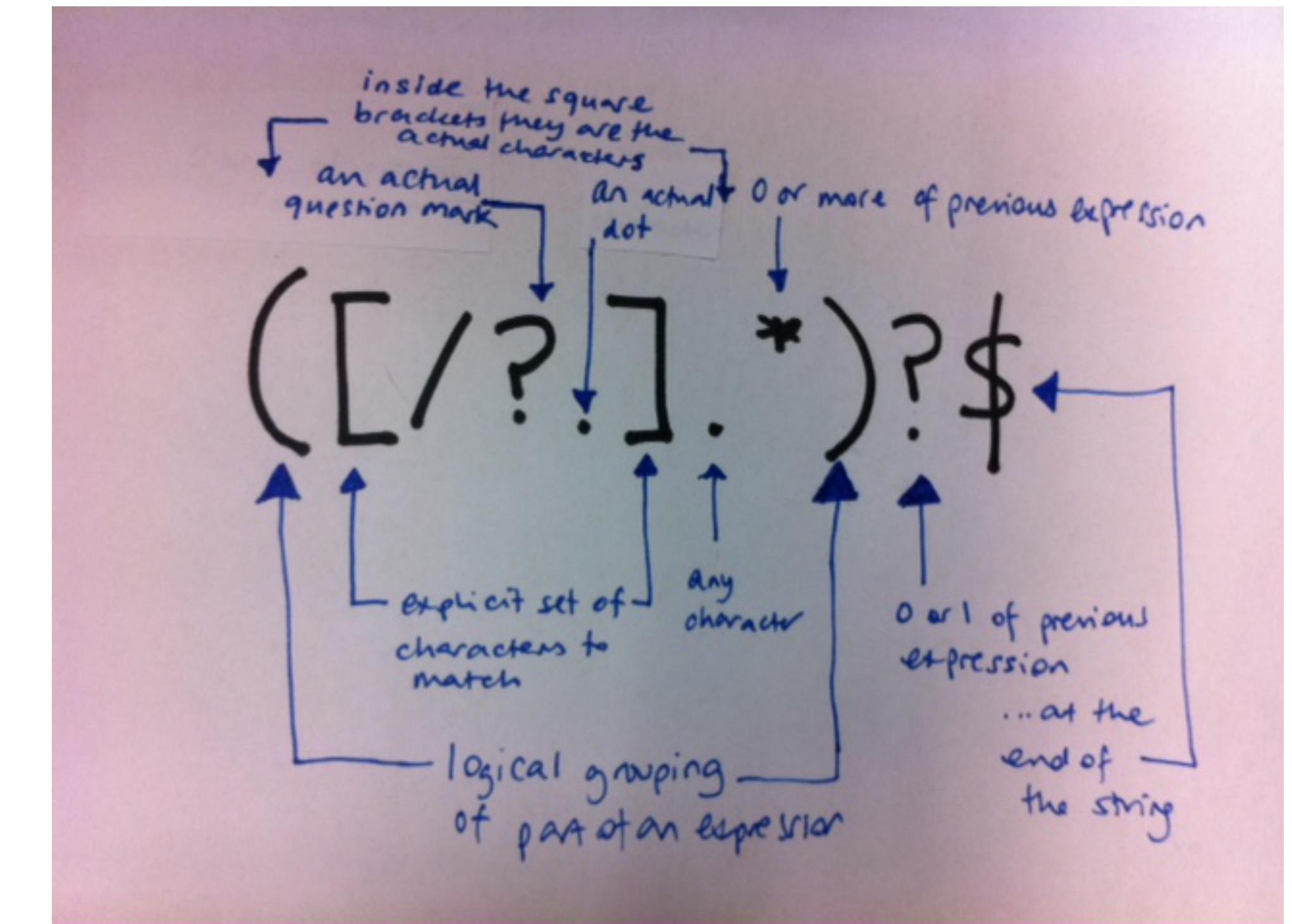
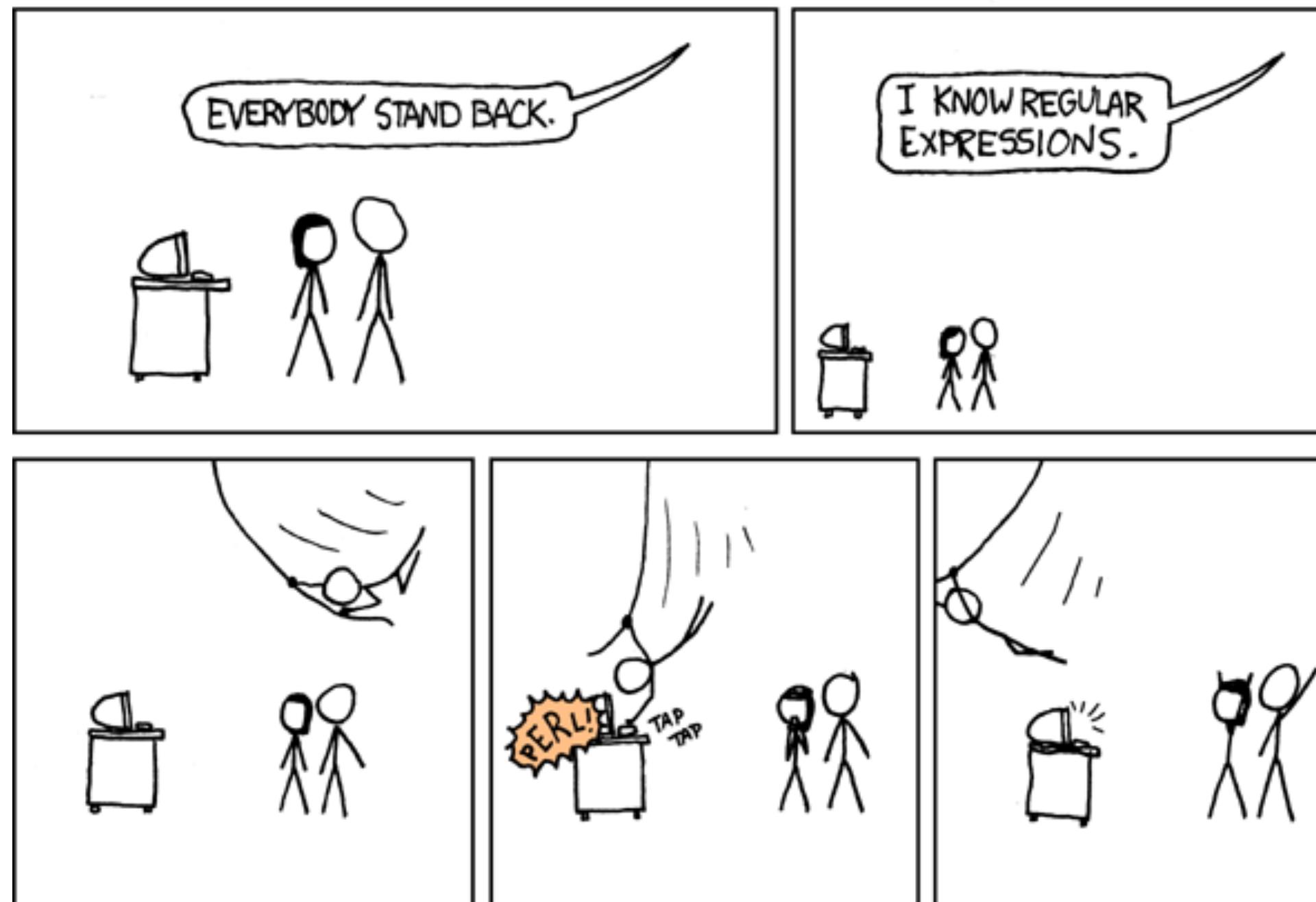


# Intro & Regular Expressions

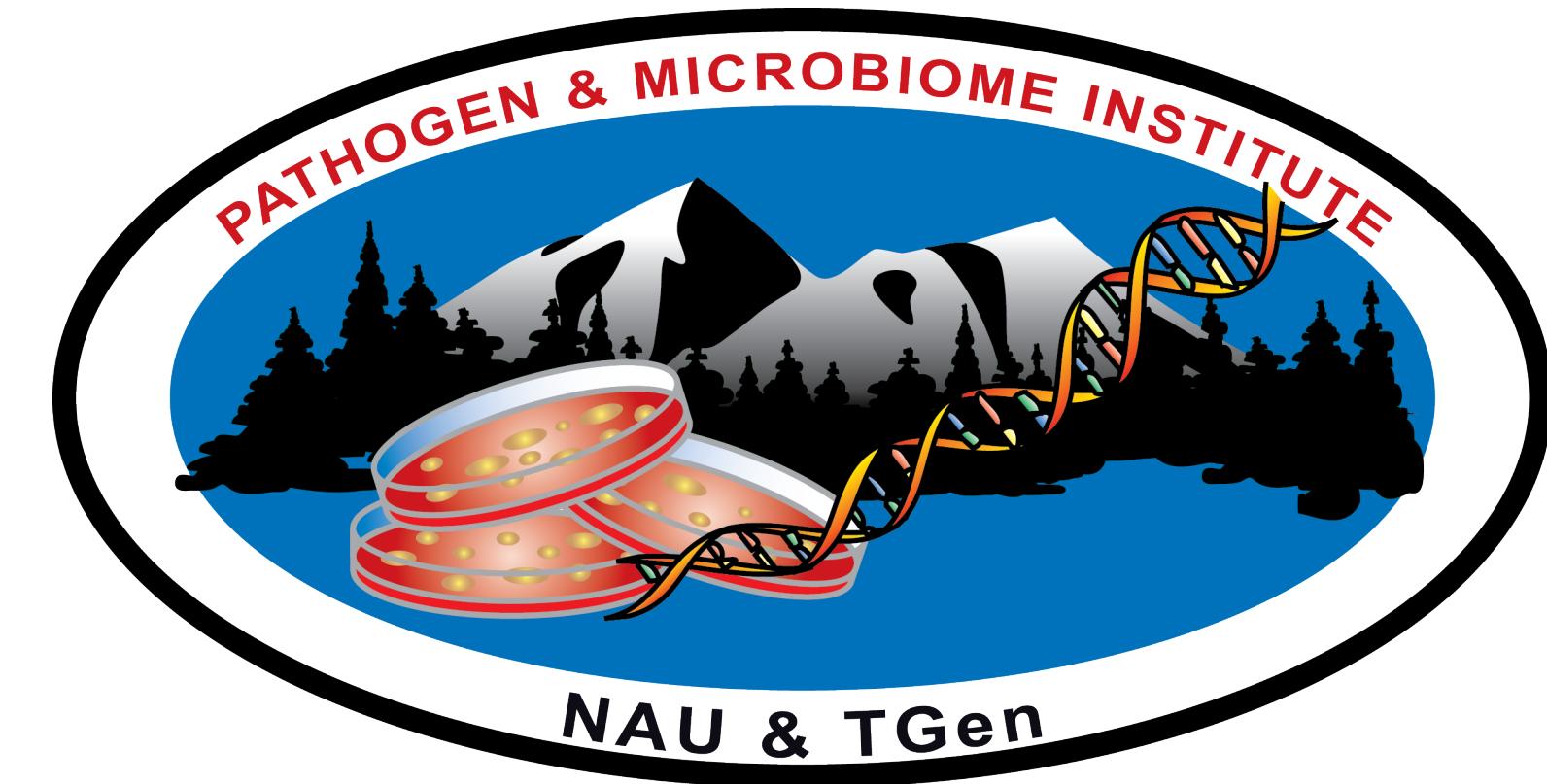
Spring 2022  
PCfB Class 1  
January 14, 2022



# Outline

- Introductions
- Course organization
- Plain text files
- Regular expressions

## PhD - Evolutionary genetics



## PostDoc - Pathogen genomics



**USAMRIID**

United States Army  
Medical Research Institute  
of Infectious Diseases

Biodefense solutions to protect our nation



**Assistant Professor**  
**Dept of Biological Sciences**  
**Pathogen and Microbiome Institute**

# Intros

- 1. Your name**
- 2. Your research focus**
- 3. What you hope to get from this class**
- 4. (Optional) Pronouns**

Course  
organization

## What this course is:

- Intro to general computing techniques broadly applicable to many research-related tasks

## What it isn't:

- A bioinformatics class

# Most course materials on



[https://github.com/jtladner/PracticalComputing\\_Spring2022](https://github.com/jtladner/PracticalComputing_Spring2022)

# “Pulling” GitHub updates

# Syllabus on Bb Learn

The screenshot shows the Bb Learn course interface for 'BIO-599 (1211-4804) CONTEMPORARY DEVELOPMENTS (Spring 2021 M16) 007 Topic - PRACTICAL COMPUTING IN BIOLOGY'. The left sidebar lists course navigation options: Course Content, Syllabus and Schedule, Assignments, Discussions, My Grades, Course Messages, Announcements, Tests/Quizzes, and NAU Help. The main content area displays the 'Syllabus and Schedule' page, featuring the NAU logo and a banner placeholder for 'SYLLABUS & SCHEDULE'. A file icon with a document and a download arrow is shown below the banner.

BIO-599 (1211-4804)

CONTEMPORARY  
DEVELOPMENTS (Spring  
2021 M16) 007 Topic -  
PRACTICAL COMPUTING  
IN BIOLOGY

Course Content

Syllabus and Schedule

Assignments

Discussions

My Grades

Course Messages

Announcements

Tests/Quizzes

NAU Help

Syllabus and Schedule

Build Content Assessments Tools Partner Content

banner placeholder

NAU

SYLLABUS & SCHEDULE

Syllabus & Schedule

# Assignments submitted via Bb Learn

The screenshot shows the Bb Learn interface. On the left, a sidebar displays course information: BIO-599 (1211-4804), CONTEMPORARY DEVELOPMENTS (Spring 2021 M16) 007 Topic - PRACTICAL COMPUTING IN BIOLOGY. Below this are links for Course Content, Syllabus and Schedule, Assignments (which is selected and highlighted in blue), and Discussions.

The main content area is titled "Assignments". It features a navigation bar with "Build Content", "Assessments", "Tools", and "Partner Content".

A specific assignment is listed:

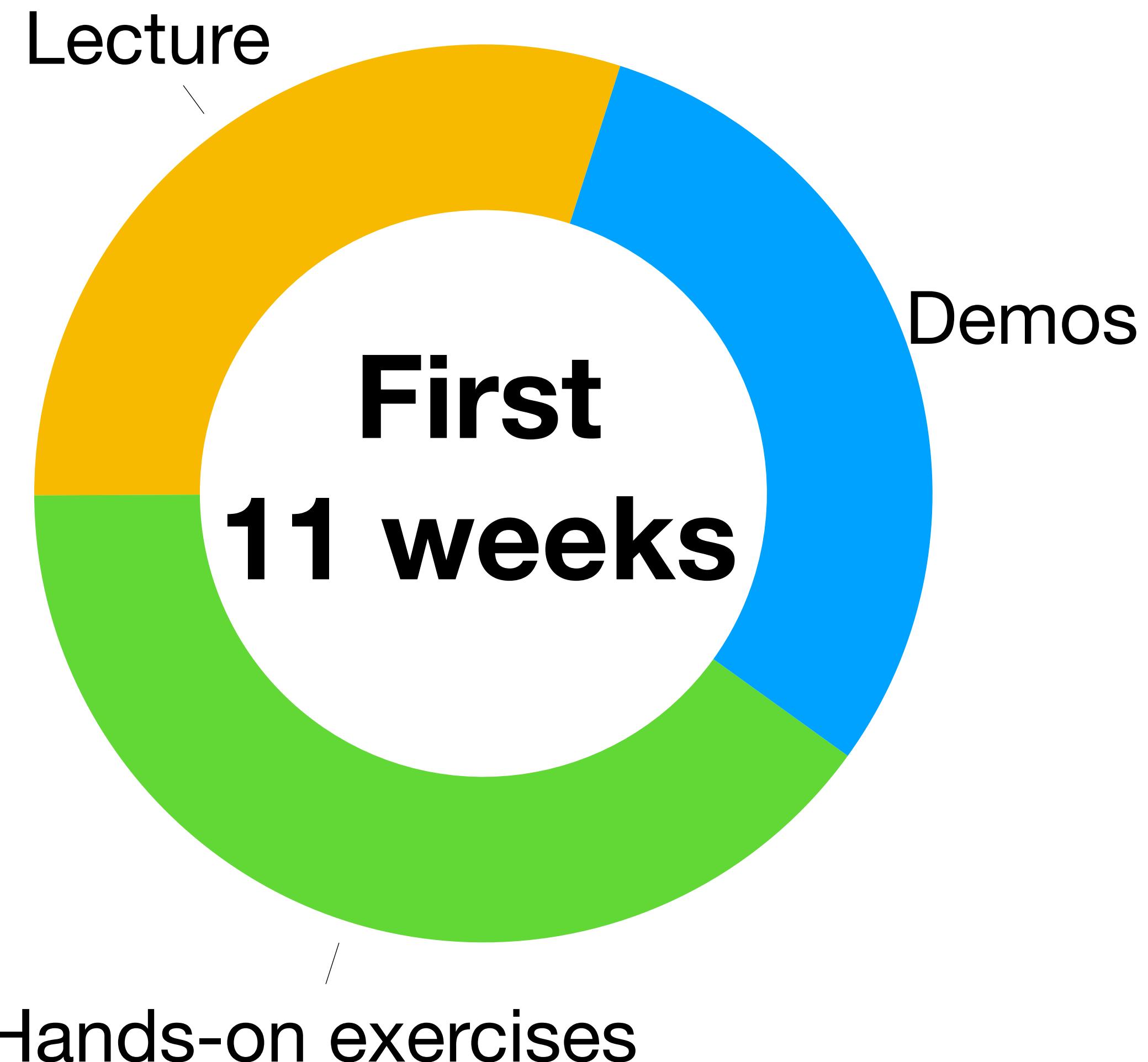
- Week 01 - Regular Expressions** (with a dropdown arrow)
- Attached Files: A file icon with a orange drop symbol, a word document icon labeled "Assignment\_01.docx" (with a dropdown arrow), and a download icon labeled "(17.936 KB)".
- Description: Assignment for Week 01 - Regular Expressions.

# Required text



- Haddock, S. H. D. and Dunn, C. W. (2010). Practical Computing for Biologists. Sinauer Associates
- <http://practicalcomputing.org/>
- Reading must be complete **PRIOR** to class

# Class organization



## Individual projects (Last 5 weeks)

- Individual coding projects
- Topic of your choice
- 3 work weeks
- 2 weeks for presentations

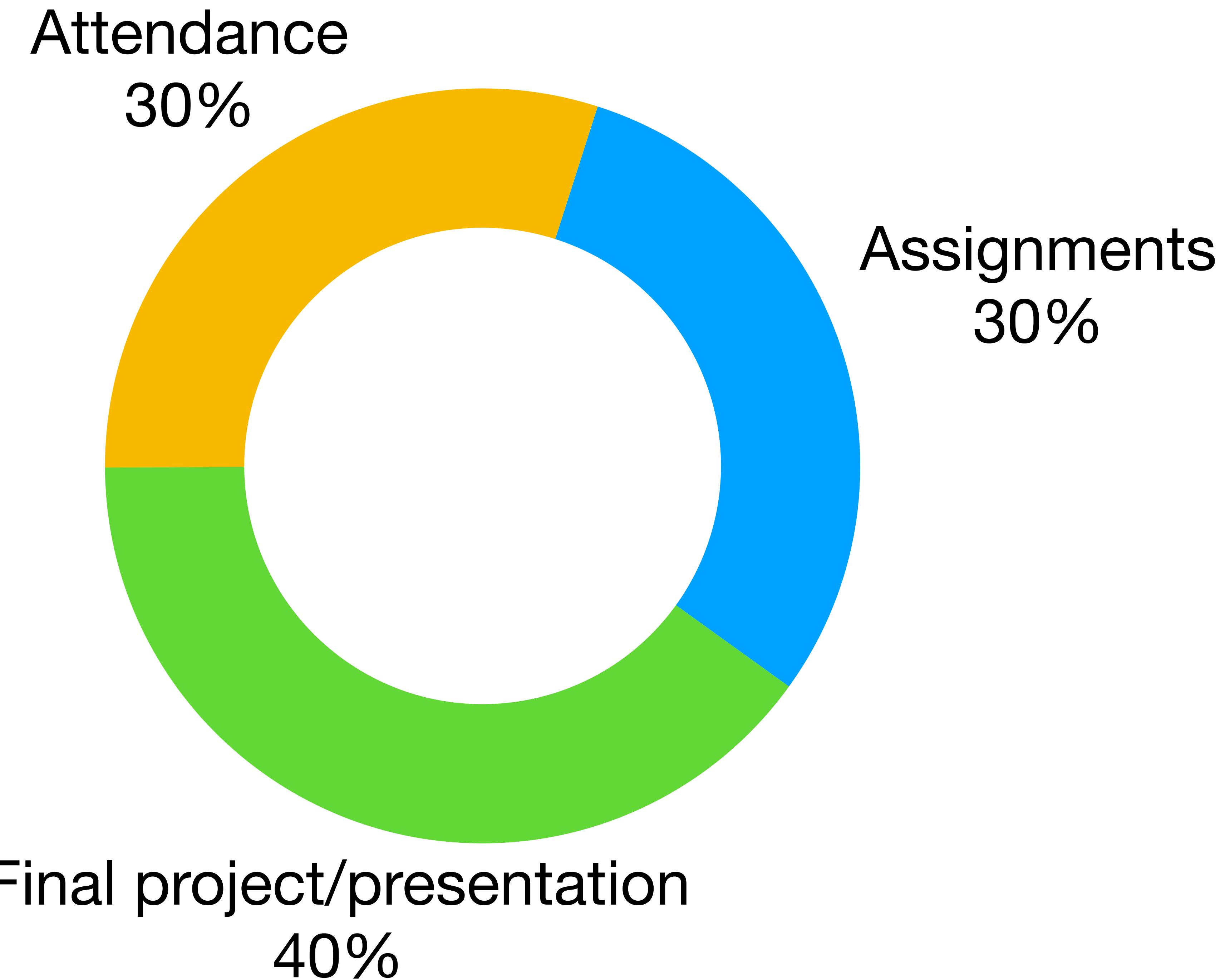
# Assignments

- One assignment per week (weeks 1-11)
- Focus on hands-on time in class (may need to complete outside of class)
- Always due by 11:59 pm on Thursday
- Directions on GitHub, Answer sheet on Bb Learn
- Partial credit for revisions

# Breakout groups for hands-on exercises

- Main room: questions/troubleshooting w/ professor
- Quiet workspace: for independent work
- Discussion spaces: for discussion among students

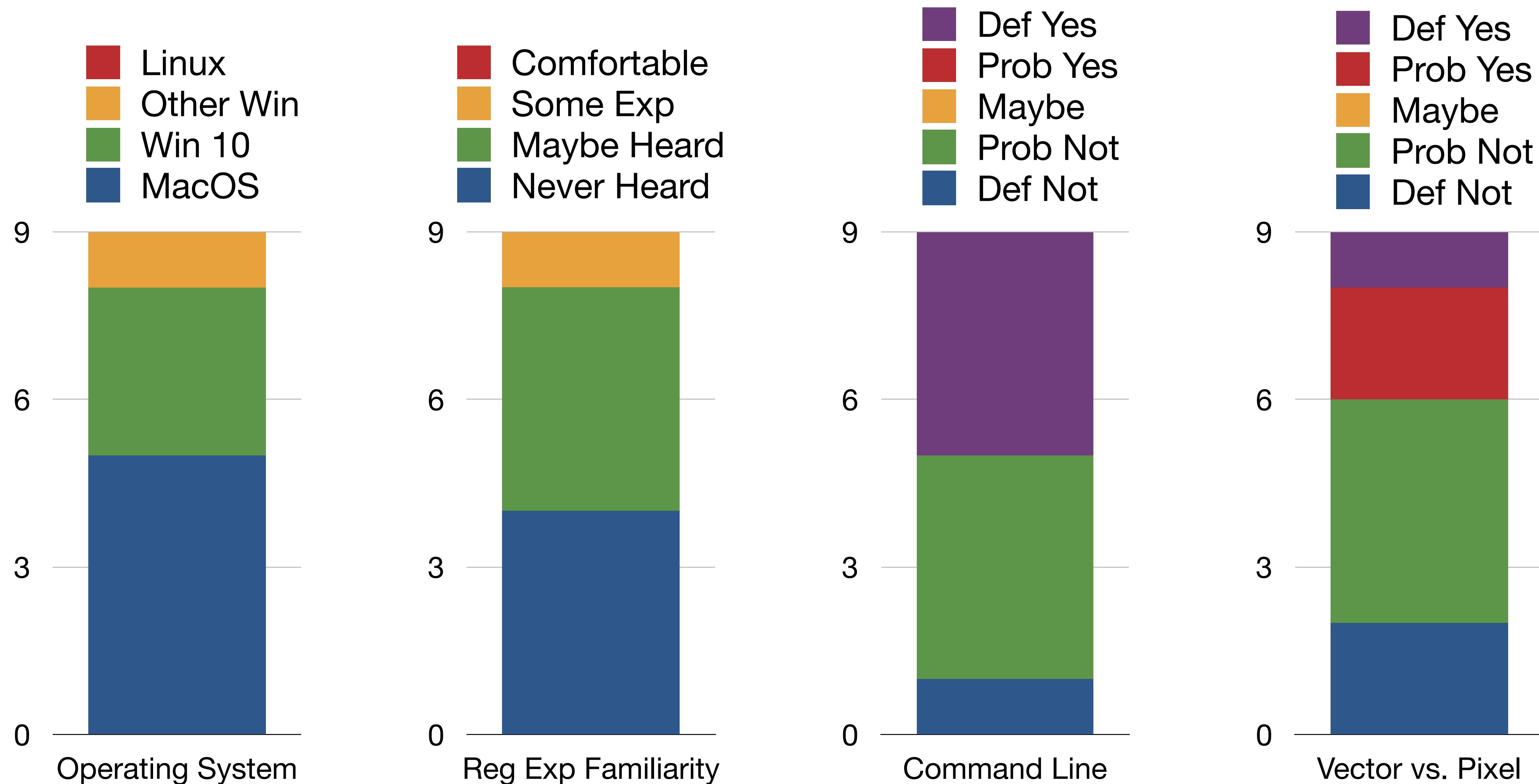
# Grading



# Final project - deadlines

Week	Date	Topic	Reading
1	1/14	Intro, Setup & Regular Expressions	PCfB: Ch. 1-3
2	1/21	The Shell - Part 1	PCfB: Ch. 4-5
3	1/28	The Shell - Part 2	PCfB: Ch. 6, 21
4	2/4	Python Programming - Part 1	PCfB: Ch. 7-8 <a href="#">Jupyter Tutorial</a>
5	2/11	Python Programming - Part 2	PCfB: Ch. 9
6	2/18	Python Programming - Part 3	PCfB: Ch. 10-11
7	2/25	Python Programming - Part 4	PCfB: Ch. 12
8	3/4	Debugging, Combining Methods	PCfB: Ch. 13-14
9	3/11	Graphical concepts: vectors vs. pixels	PCfB: Ch. 17-19
10	3/18	Making Figures in Python - Part 1	<a href="#">Matplotlib overview</a>
11	3/25	Making Figures in Python - Part 2 (*Project proposal due)	
12	4/1	Work/Troubleshoot Day #1	
13	4/8	Work/Troubleshoot Day #2	
14	4/15	Work/Troubleshoot Day #3	
15	4/22	Project Presentations - Part 1	
16	4/29	Project Presentations - Part 2	
Finals	5/2	*Final project due	

# Pre-class survey



# Computer setup

- Text Editor
- GitHub Repository
- Command line terminal

[https://github.com/jtladner/PracticalComputing\\_Spring2022/  
tree/master/Getting%20Started](https://github.com/jtladner/PracticalComputing_Spring2022/tree/master/Getting%20Started)

# Plain text

## files

# Plain text file

- Pure sequence of character codes
- No formatting (e.g., text size, color, font, spacing)
- Human and machine readable
- Standardized

# Which of these formats are NOT plain text?

**Excel (.xlsx)**

**html**

**OpenOffice (.odf)**

**Google Sheet**

**text (.txt)**

**fasta**

**markdown**

**xml**

**Google Doc**

**nexus**

**json**

**Word (.doc)**

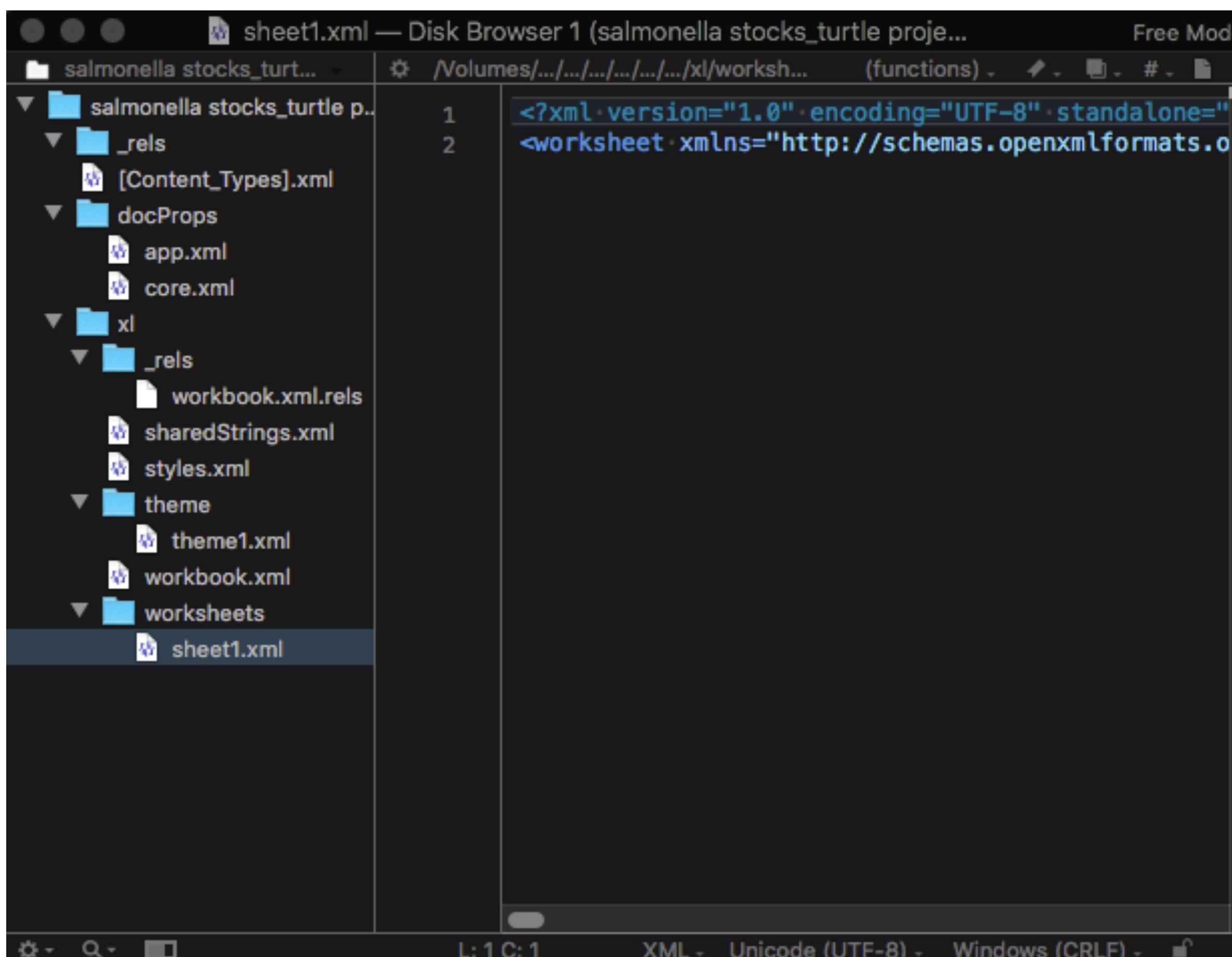
**rich text (.rtf)**

**python script (.py)**

**tab-separated (.tsv)**

# Viewing non-plain text in text editor

.xlsx/.docx



sheet1.xml — Disk Browser 1 (salmonella stocks\_turtle proje... Free Mode

/Volumes/.../.../.../.../.../.../xl/worksh... (functions) #

salmonella stocks\_turt...

salmonella stocks\_turtle p...

\_rels

[Content\_Types].xml

docProps

app.xml

core.xml

xl

\_rels

workbook.xml.rels

sharedStrings.xml

styles.xml

theme

theme1.xml

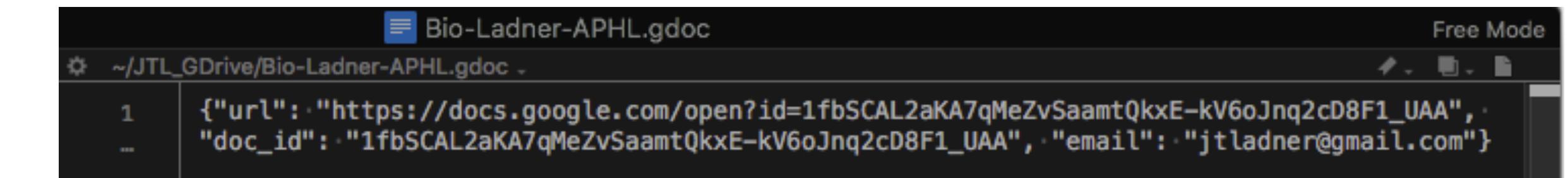
workbook.xml

worksheets

sheet1.xml

L: 1 C: 1 XML - Unicode (UTF-8) - Windows (CRLF) -

Google Doc

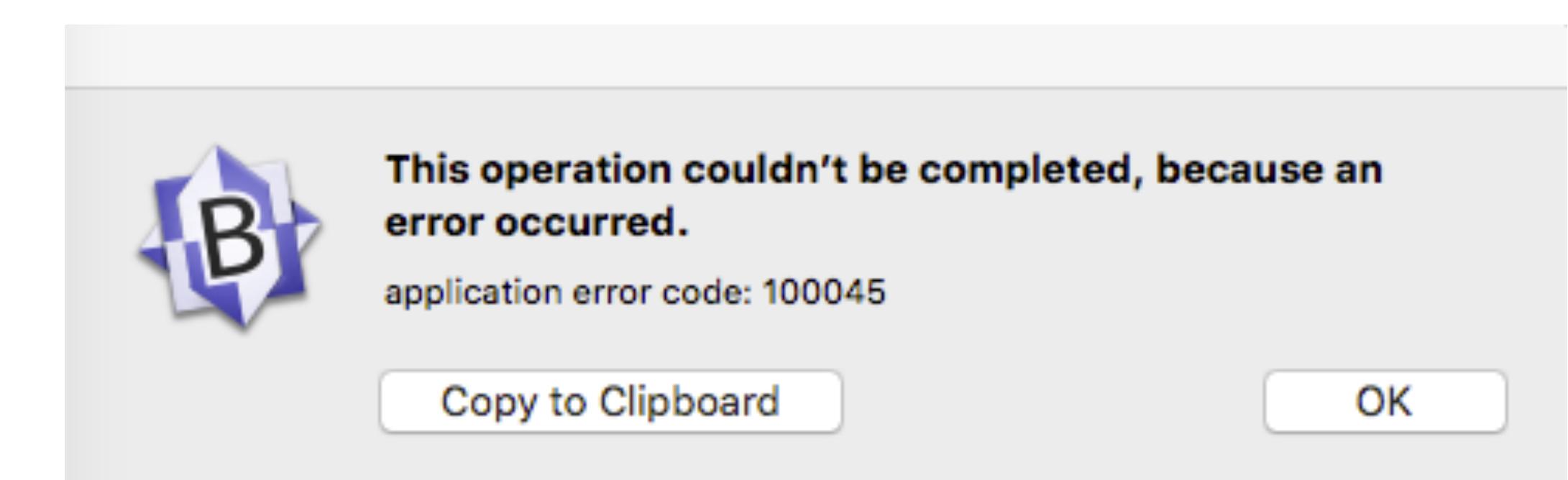


Bio-Ladner-APHL.gdoc — Free Mode

~/JTL\_GDrive/Bio-Ladner-APHL.gdoc

1 {"url": "https://docs.google.com/open?id=1fbSCAL2aKA7qMeZvSaamtQkxE-kV6oJnq2cD8F1\_UAA", "doc\_id": "1fbSCAL2aKA7qMeZvSaamtQkxE-kV6oJnq2cD8F1\_UAA", "email": "jtladner@gmail.com"}

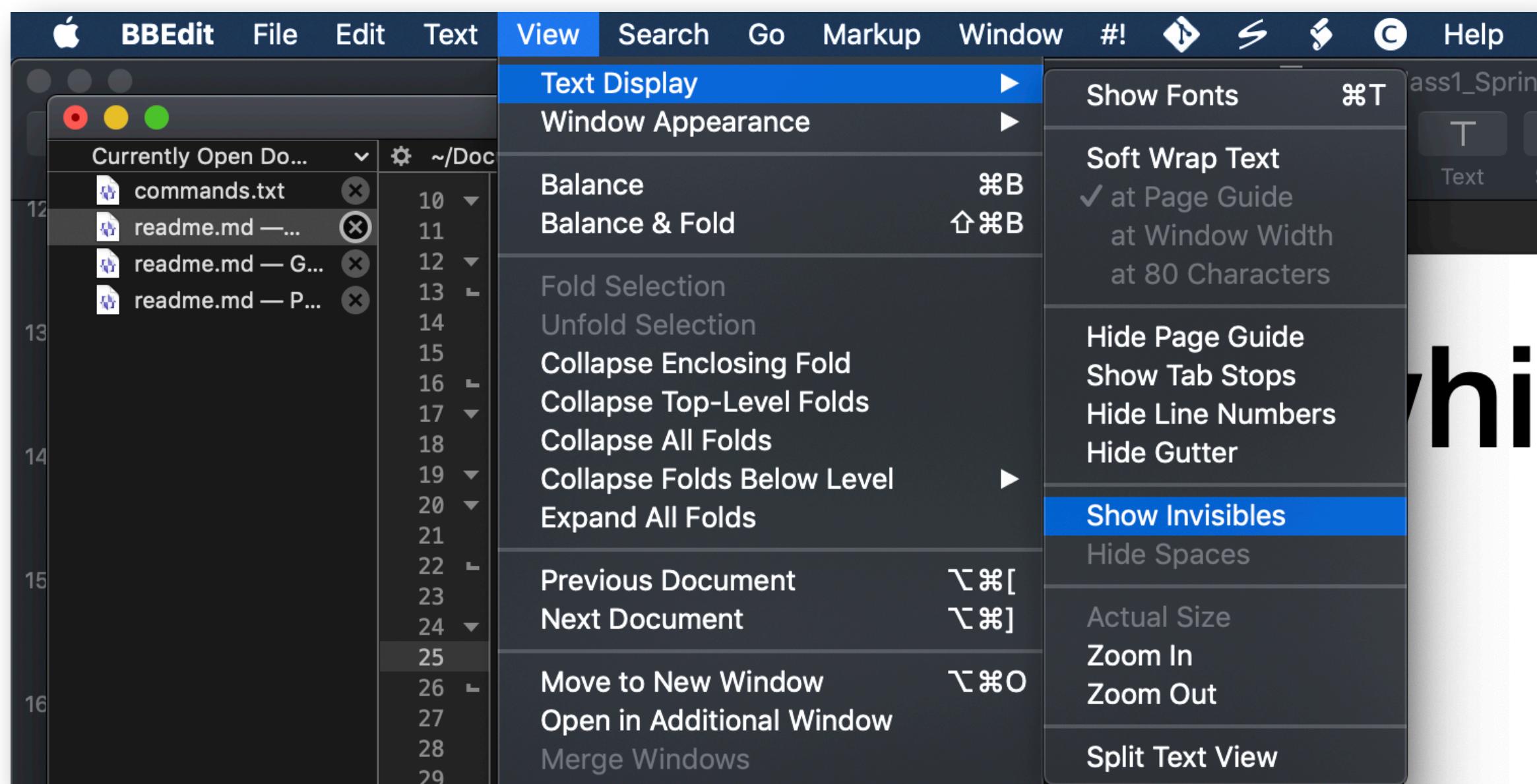
Google Sheet



# Whitespace

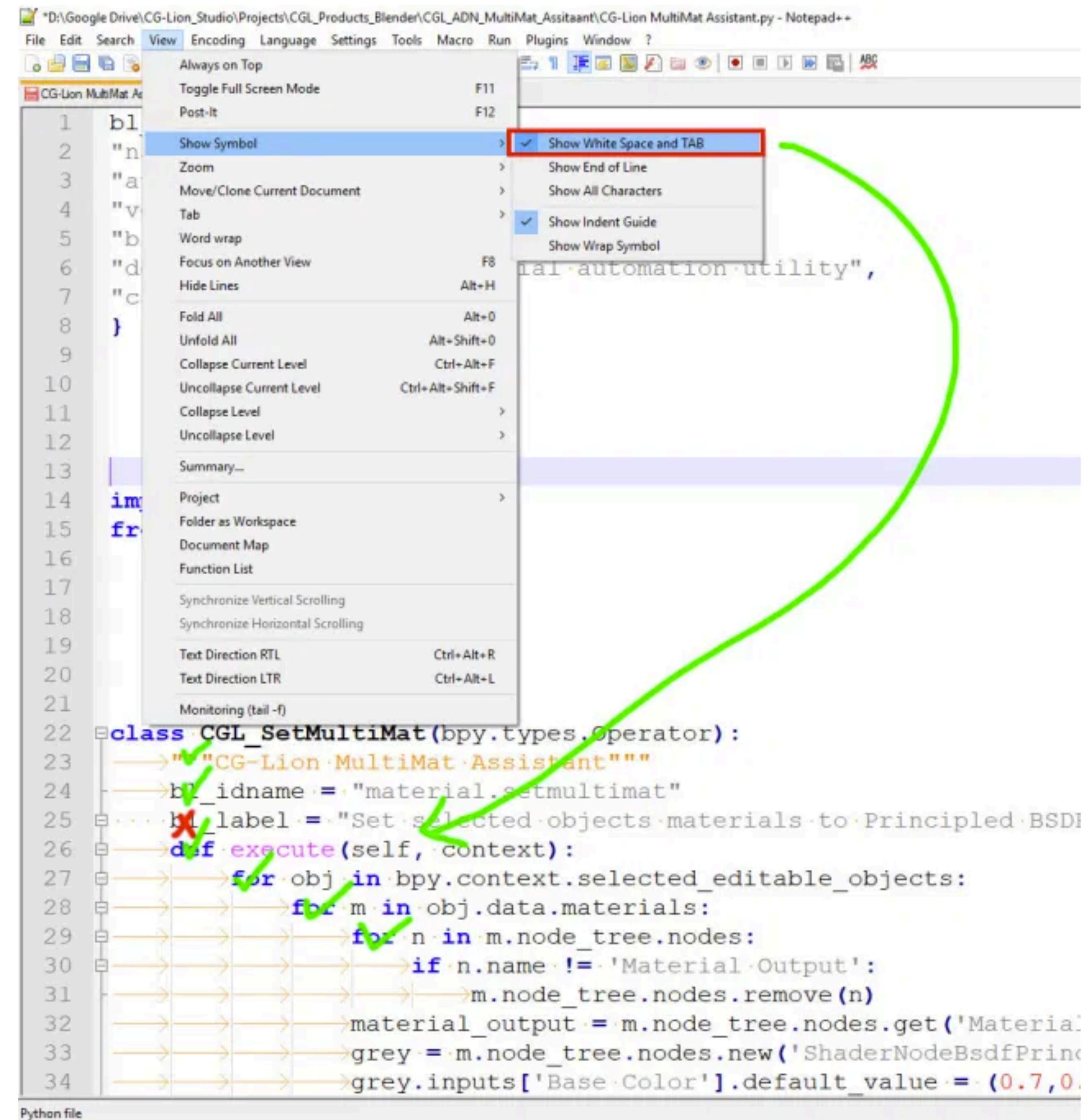
- Space
- Tab
- End of line

# Visualizing white space (BBEdit)



```
10  ### Prep for next class
11
12  1. Open your command line interface and type this command f
13  ``echo $SHELL``
14
15  If the response is not "/bin/bash", let me know.
16
17  ### Assignment
18
19  1. Go to [RegexOne](https://regexone.com/) and complete the
20      - Although the interface will allow you to only match a
21      - Keep track of your solutions in the table provided in
```

# Visualizing white space (Notepad++)



The screenshot shows the Notepad++ interface with the 'View' menu open. The 'Show White Space and TAB' option is highlighted with a red box and checked. A green circle highlights this same option in the menu. The main text area displays Python code for a Blender operator, with yellow vertical lines indicating whitespace and tabs.

```
*D:\Google Drive\CG-Lion_Studio\Projects\CGI_Products_Blender\CGL_ADN_MultiMat_Assistaant\CG-Lion MultiMat Assistant.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Always on Top F11
Toggle Full Screen Mode F12
Post-it
Show Symbol > ✓ Show White Space and TAB
Zoom > Show End of Line
Move/Clone Current Document > Show All Characters
Tab > ✓ Show Indent Guide
Word wrap > Show Wrap Symbol
Focus on Another View F8
Hide Lines Alt+H
Fold All Alt+O
Unfold All Alt+Shift+O
Collapse Current Level Ctrl+Alt+F
Uncollapse Current Level Ctrl+Alt+Shift+F
Collapse Level >
Uncollapse Level >
Summary...
Project >
Folder as Workspace
Document Map
Function List
Synchronize Vertical Scrolling
Synchronize Horizontal Scrolling
Text Direction RTL Ctrl+Alt+R
Text Direction LTR Ctrl+Alt+L
Monitoring (tail -f)
class CGL_SetMultiMat(bpy.types.Operator):
    """CG-Lion MultiMat Assistant"""
    bl_idname = "material.setmultimat"
    bl_label = "Set selected objects materials to Principled BSDF"
    def execute(self, context):
        for obj in bpy.context.selected_editable_objects:
            for m in obj.data.materials:
                for n in m.node_tree.nodes:
                    if n.name != 'Material Output':
                        m.node_tree.nodes.remove(n)
                        material_output = m.node_tree.nodes.get('Material Output')
                        grey = m.node_tree.nodes.new('ShaderNodeBsdfPrincipled')
                        grey.inputs['Base Color'].default_value = (0.7, 0.7, 0.7, 1)
```

# BEdit

The screenshot shows the BBEdit interface with a file named 'readme.md' open. The file contains a multi-step assignment for a class. It includes instructions for preparing the environment, using a command-line interface, and completing a challenge on RegexOne. It also provides a Fasta format example and steps for editing regular expressions and uploading a final file. The text is in a monospaced font with some syntax highlighting.

```
## Prep for next class
1. Open your command line interface and type this command
``echo $SHELL``

If the response is not "/bin/bash", let me know.

### Assignment
1. Go to [RegexOne](https://regexone.com/) and complete
   - Although the interface will allow you to only match
   - Keep track of your solutions in the table provided
2. We will now start using the text editor on your computer
   1. Open "EBOV.phy" in your text editor.
   2. Use a series of find/replace queries to convert the file to FASTA format.
      - Fasta format example:
        >name1
        seq1
        >name2
        seq2
      - Edit your regular expression(s) to also change the file to FASTA format.
      - Edit your regular expression(s) to remove all 'N' characters.
      - Upload final fasta-formatted file to Bb Learn along with your assignment.
3. Open "HastingsBirdList\_2007\_.txt" in your text editor
   1. Design a single search and replace query that utilizes regular expressions to convert the file to FASTA format.
```

# Notepad++

The screenshot shows the Notepad++ interface with two tabs open. The top tab, titled 'new 1.txt', displays text with various indentation levels: one tab indent and two tab indents. The bottom tab, titled '\*new 4 - Notepad++', displays text with CRLF line endings.

**new 1.txt**

```
1 This is a line without any tab or spaces.
2
3 This line contains two spaces but no tab.
4
5 → This line is indicating one tab indent.
6
7 →→ This line is indicating two tab indents.
```

**\*new 4 - Notepad++**

```
1 This is a line of text CRLF
2 This is another line of text CRLF
3
```

# End of line characters differ by OS

- Line feed (LF) - Mac OSX, Linux
- Carriage return (CR) - Mac OS9 and earlier
- Carriage return + line feed (CRLF) - Windows

# Regular expressions

# Regular expressions

(a.k.a. regex, regexp)

- Powerful find and replace toolkit
- Understood by many text editors, programming languages and even search engines
- Power comes from wildcard operators

\d

\w

\s



\w+

\w\*

\w?

[ABC]

[^ABC]

[A-C]

(ABC)

(AB)C

((AB)C)

Λ

\$

# Tips

- Try PCfB methodology
  - copy target text into search dialog
  - replace text with wildcards, piece by piece
- Be as specific as possible
- Build in redundancies

# Regexp reference tables

## Wildcards

\w	Letters, numbers and _
.	Any character except \n \r
\d	Numerical digits
\t	Tab
\r	Return character. Also used as the generic end-of-line character in TextWrangler
\n	Line-feed character. Also used as the generic end-of-line character in Notepad++
\s	Space, tab, or end of line
[A-Z]	A single character of the ranges indicated in square brackets
[^A-Z]	A single character including all characters <i>not</i> in the brackets. Note that this will include \n unless otherwise specified, and may cause you to match across lines
\	Used to escape punctuation characters so they are searched for as themselves, not interpreted as wildcards or special symbols
\\	The \ symbol itself, escaped

## Boundaries

^	Match the start of the line, i.e., the position before the first character
\$	Match the last position before the end-of-line character

## Quantifiers, used in combination with characters and wildcards

+	Look for the longest possible match of one or more occurrences of the character, wildcard, or bracketed character range immediately preceding. The match will extend as far as it can while still allowing the entire expression to match.
*	As above, matches as many of the previous character to occur, but allows for the character not to occur at all if the match still succeeds
?	Modifies greediness of + or * to match the shortest possible match instead of longest
{}	Specify a range of numbers to repeat the match of the previous character. For example: \d{2,4} matches between 2 and 4 digits in a row [AC]{4,} matches 4 or more of the letter A or C in a row
	<b>Capturing and replacing</b>
( )	Capture the search results between the parentheses for use in the replacement term
\1 \$1	Substitute the contents of the matched into the replacement term, in numerical order. Syntax depends on the text editor or language that you are using.

**Questions about the  
reading?**



# RegexOne

Learn Regular Expressions with simple, interactive exercises.

## Exercise 1: Matching Characters

Task	Text	
------	------	--

Match	abcd <ins>efg</ins>	✓
-------	---------------------	---

Match	abcde	✓
-------	-------	---

Match	abc	✗
-------	-----	---

[ e-g ]+

Continue >

*Solve the above task to continue on to the next problem, or read the [Solution](#).*

## Exercise 1: Matching Characters

Task	Text	
------	------	--

Match	abcdefg	✓
-------	---------	---

Match	abcde	✓
-------	-------	---

Match	abc	✓
-------	-----	---

\w+

Continue >

*Solve the above task to continue on to the next problem, or read the [Solution](#).*

# “Prep for next class”

## Class 1 - Jan. 14th 2022

- In this first class we will:
  - Discuss the syllabus and course organization/expectations
  - Troubleshoot computer setup problems
  - Learn to use regular expressions to edit text files

### Required Reading (Must be completed ahead of time)

Practical Computing for Biologists, Chapters 1-3

### Prep for next class

1. Open your command line interface and type this command followed by 'Enter': `echo $SHELL`

If the response is not "/bin/bash" or "/bin/zsh", let me know.

1 | 43·18971~  
2 | LIBR4748\_2015-06-29·NNNNNNNNNNNNNNNAGAAGAATTTAGGATCTTGTGTGCGAATAA  
3 | LIBR4783\_2015-06-30·NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNATAA  
4 | LIBR4808\_2015-07-01·NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNTTGTGTGCGAATAA  
5 | LIBR4866\_2015-07-02·NNNNNNNNNNNNNNNAGAATTTAGGATCTTGTGTGCGAATAA  
6 | LIBR5046\_2015-07-07·NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNATAA  
7 | LIBR5047\_2015-07-07·NN  
8 | LIBR5081\_2015-07-08·NN  
9 | LIBR5307\_2015-07-12·NNNNNNNNNNNGAAAGAAGAATTTAGGATCTTGTGTGCGAATAA  
10 | BF-KN\_2014-08-24·NNNNNNNNNNNNNNNNNAATTTAGGATCTTGTGTGCGAATAACTA

1 | >LIBR4748~  
2 | AGAAGAATTTAGGATCTTGTGTGCGAATAACTATGAGGAAGATTAAATAATTTCCTCTCATTGAAATTATATCGGAATT  
3 | >LIBR4783~  
4 | ATAACTATGAGGAAGATTAAATAATTTCCTCTCATTGAAATTATATCGGAATTAAATTGAAATTGTTACTGTAATCATACT  
5 | >LIBR4808~  
6 | TTTGTGTGCGAATAACTATGAGGAAGATTAAATAATTTCCTCTCATTGAAATTATATCGGAATTAAATTGAAATTGTTACTG  
7 | >LIBR4866~  
8 | AGAATTTTAGGATCTTGTGTGCGAATAACTATGAGGAAGATTAAATAATTTCCTCTCATTGAAATTATATCGGAATTAA  
9 | >LIBR5046~  
10 | ATAACTATGAGGAAGATTAAATAATTTCCTCTCATTGAAATTATATCGGAATTAAATTGAAATTGTTACTGTAATCATACT  
11 | >LIBR5047~  
12 | GGAAGATTAAATAATTTCCTCTCATTGAAATTATATCGGAATTAAATTGAAATTGTTACTGTAATCATACTGGTTGTTCT  
13 | >LIBR5081~  
14 | ACCTGGTTGTTCAGAGCCATATCACCAAGATAGAGAACACCTAGGTCTCCGGAGGGGGCAAGGGCATCAGTGTGCTCAGTT  
15 | >LIBR5307~  
16 | GAAAGAAGAATTAGGATCTTGTGTGCGAATAACTATGAGGAAGATTAAATAATTTCCTCTCATTGAAATTATATCGGA  
17 | >BF-KN~  
18 | AATTAGGATCTTGTGTGCGAATAACTATGAGGAAGATTAAATAATTTCCTCTCATTGAAATTATATCGGAATTAAAT

**Text editor**

**regex demos**

# Start (email)

Sample ID	sample collection date	Gender	Age	Location		Sex(F/M)	Age(years)
N27	22.04.2020	100010117153	F	52	Trondelag		
N28	22.04.2020	100010117157	M	51	Trondelag		
N29	22.04.2020	100010117161	M	31	Trondelag		
N30	20.04.2020	121252.43310	M	67	Trondelag		
N31	21.04.2020	121097.39802	F	22	Trondelag		
N32	14.04.2020	100010126959	F	57	Trondelag		
N33	Oslo	20.03.2020	17.04.2020	COVID-19 convalescent	J000920011268	F	30
N34	Oslo	22.03.2020	17.04.2020	COVID-19 convalescent	J000920011287	F	47
N35	Oslo	09.03.2020	17.04.2020	COVID-19 convalescent	J000920011293	M	35
N36	Oslo	13.03.2020	17.04.2020	COVID-19 convalescent	J000920011322	F	53
N37	Oslo	09.03.2020	17.04.2020	COVID-19 convalescent	J000920011324	M	38
N38	Oslo	25.03.2020	17.04.2020	COVID-19 convalescent	J000920011341	F	50
N39	Oslo	25.03.2020	17.04.2020	COVID-19 convalescent	J000920011353	F	78
N40	Oslo	23.03.2020	17.04.2020	COVID-19 convalescent	J000920011348	F	58
N41	Oslo	11.03.2020	16.04.2020	COVID-19 convalescent	J000920011072	M	52
N42	Oslo	27.03.2020	16.04.2020	COVID-19 convalescent	J000920011091	F	70
N43	Oslo	11.03.2020	16.04.2020	COVID-19 convalescent	J000920011095	F	36

**End  
(tsv)**

SampleID	SampleCollectionDate	UnkID	Gender	Age	Location
N27	2020-04-22	100010117153	F	52	Trondelag
N28	2020-04-22	100010117157	M	51	Trondelag
N29	2020-04-22	100010117161	M	31	Trondelag
N30	2020-04-20	121252.43310	M	67	Trondelag
N31	2020-04-21	121097.39802	F	22	Trondelag
N32	2020-04-14	100010126959	F	57	Trondelag
N33	2020-03-20	J000920011268	F	30	Oslo
N34	2020-03-22	J000920011287	F	47	Oslo
N35	2020-03-09	J000920011293	M	35	Oslo
N36	2020-03-13	J000920011322	F	53	Oslo
N37	2020-03-09	J000920011324	M	38	Oslo
N38	2020-03-25	J000920011341	F	50	Oslo
N39	2020-03-25	J000920011353	F	78	Oslo
N40	2020-03-23	J000920011348	F	58	Oslo
N41	2020-03-11	J000920011072	M	52	Oslo
N42	2020-03-27	J000920011091	F	70	Oslo
N43	2020-03-11	J000920011095	F		Oslo