Lecture 03 – Scripting with Bash

Prof. Brendan Kochunas

NERS/ENGR 570 - Methods and Practice of Scientific Computing (F22)

COLLEGE OF ENGINEERING
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES
UNIVERSITY OF MICHIGAN

Outline

- ~~Lecture 2 and Lab 1 assignment review~~
- Review Survey Results
- Continued hands on of Fortran, C, and C++
- Introduction to Shell/Bash Scripting
- Homework 1
- ~~Introduction to Python~~ (No time this year!)
    - Slides still included

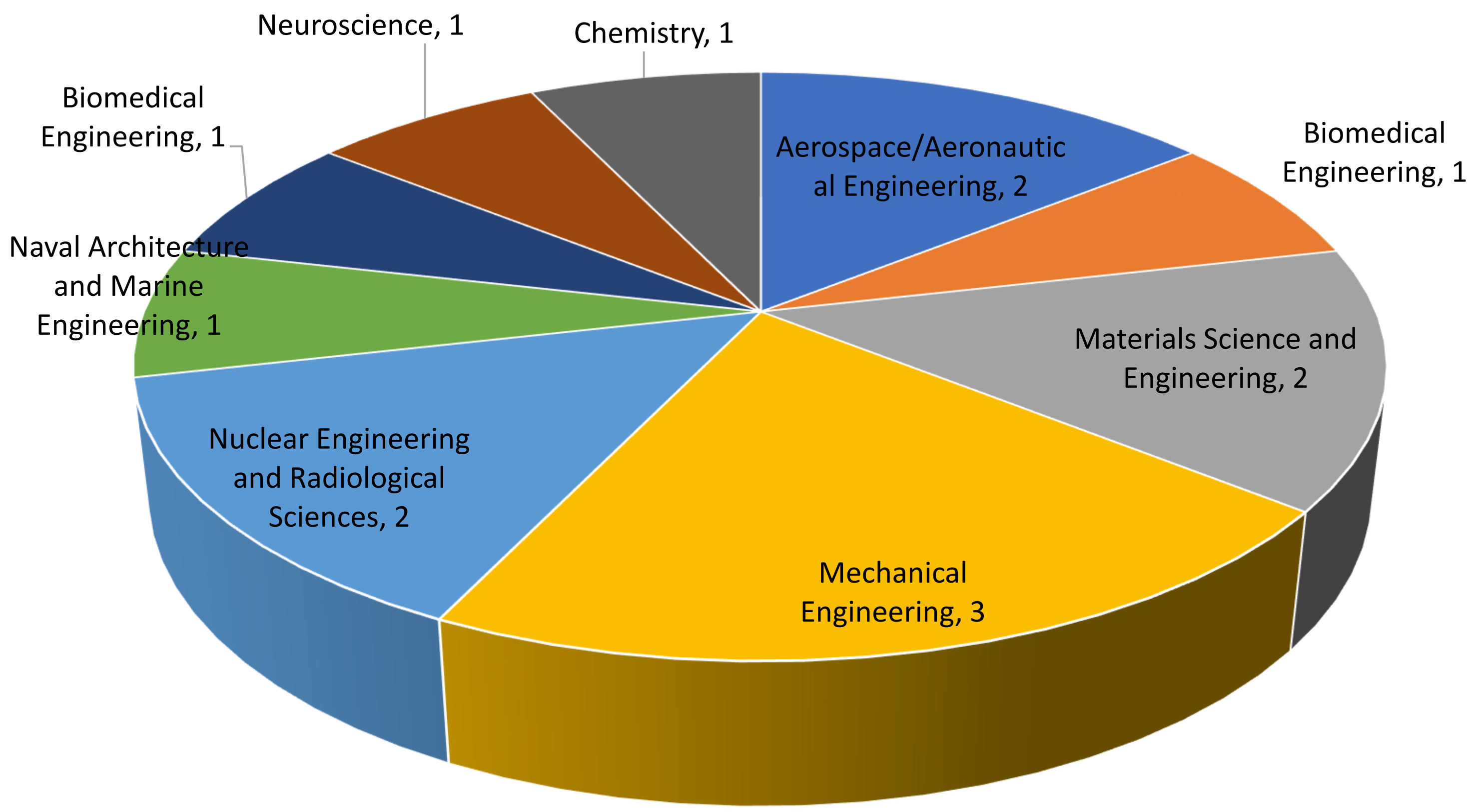
Learning Objectives: By the end of Today's Lecture you should be able to…

- (*Skill*) Declare variables that are matrices, and assign or edit their values in Fortran, C, and C+
- (*Knowledge*) explain the basic level of current expertise of you and your peers in Scientific Computing
- (*Skill*) Declare variables in bash
- (*Skill*) use execution constructs in bash
  - e.g. loops and branching constructs
- (*Knowledge*) explain what must done for homework 1

Survey says?

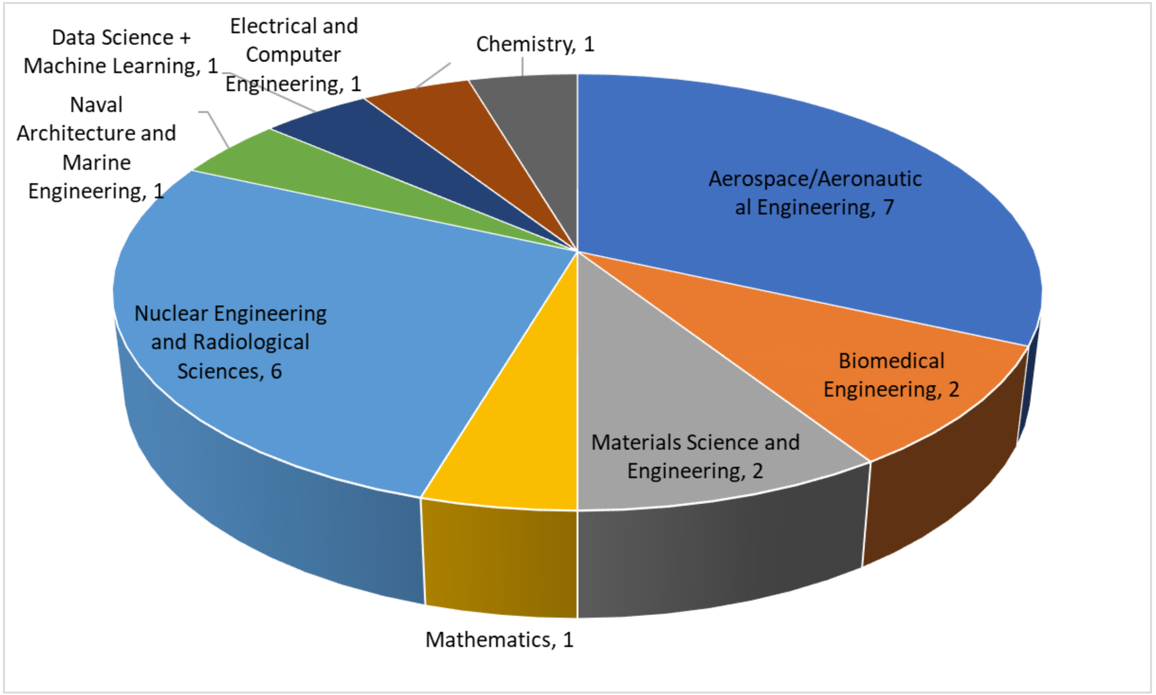
…only 14 students replied

Field of Study

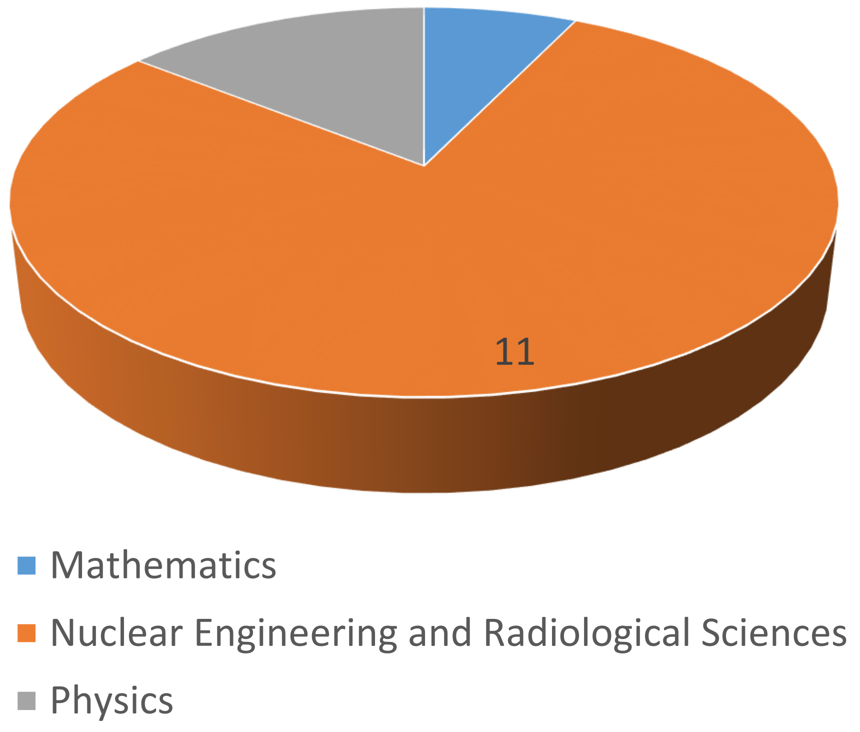


Survey Says…

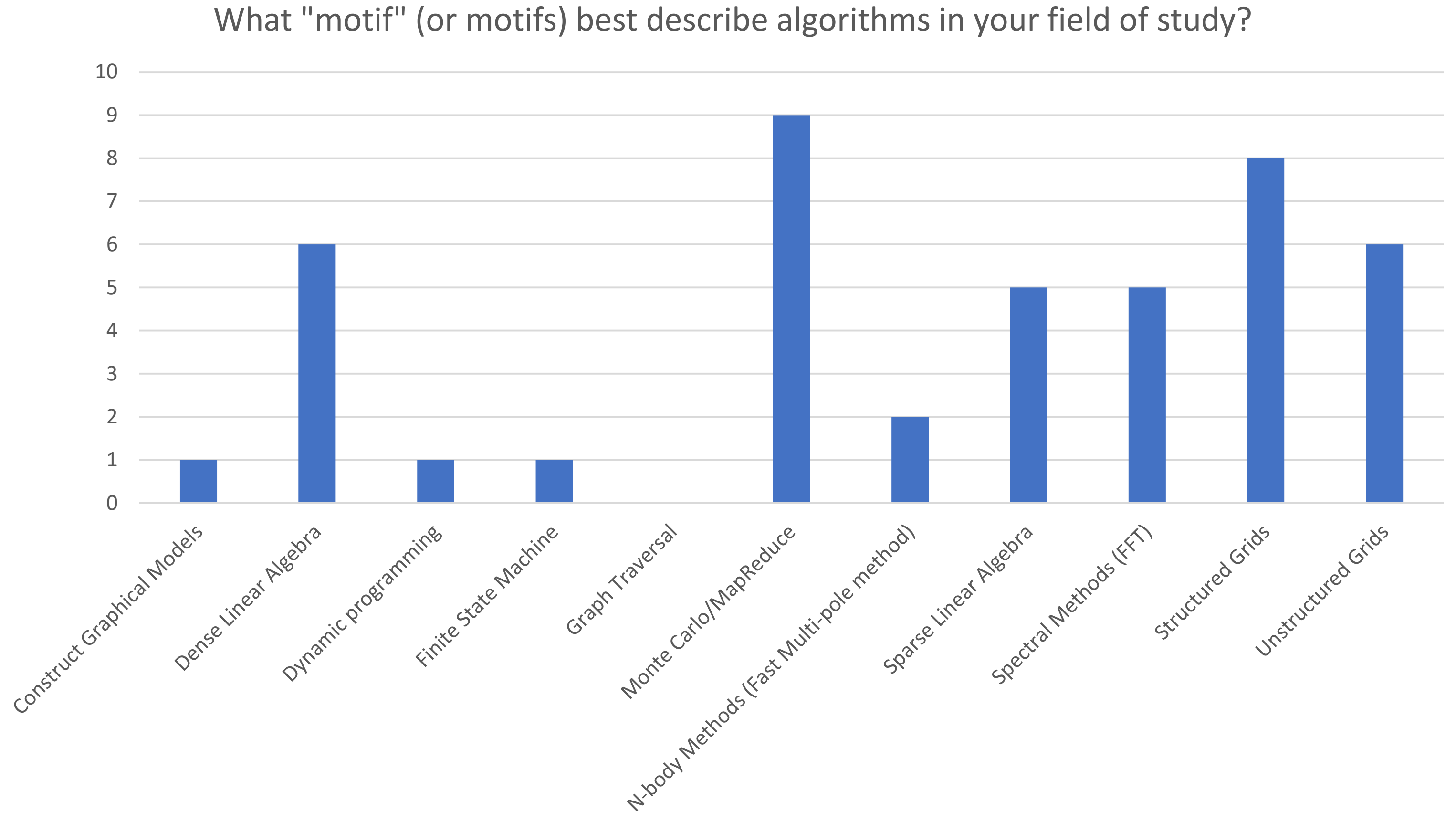
2021



2020

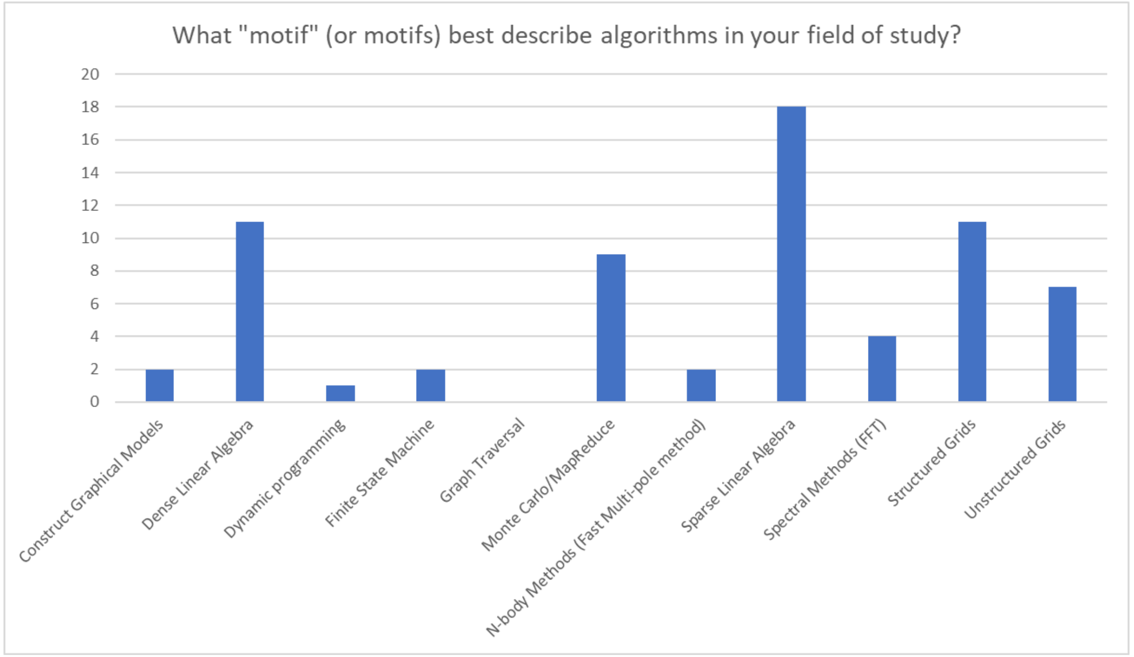


Motifs

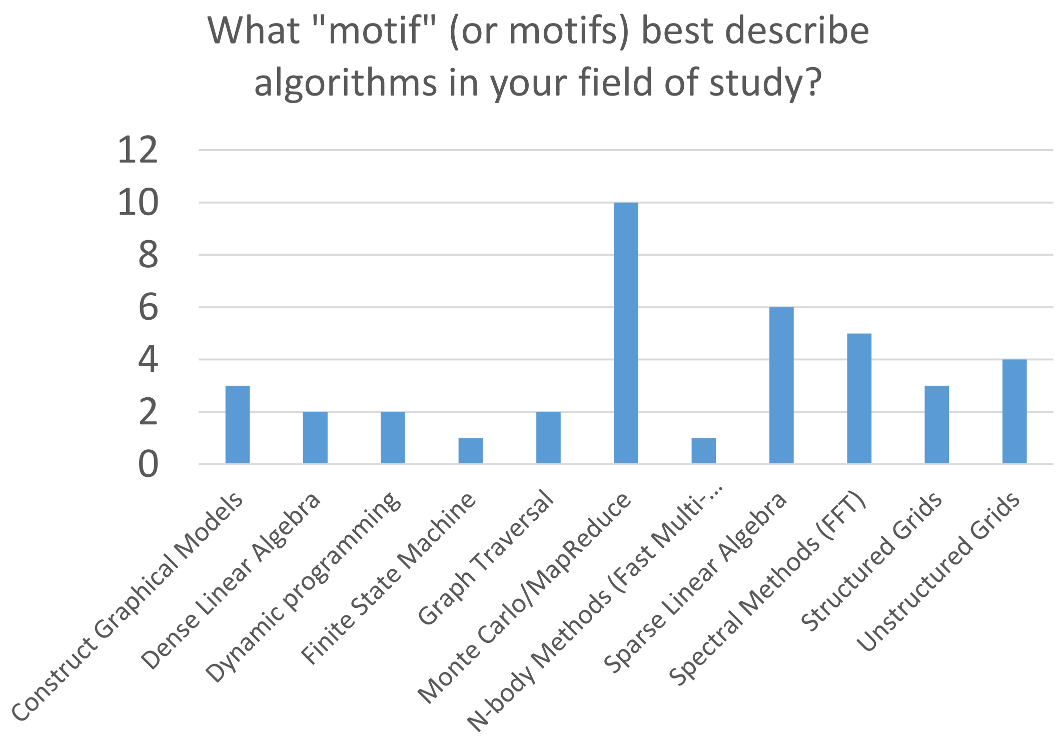

What "motif" (or motifs) best describe algorithms in your field of study?
10
9
8
7
6
5
4
3
2
1
0
Construct Graphical Models
Dense Linear Algebra
Dynamic programming
Finite State Machine
Graph Traversal
Monte Carlo/MapReduce
N-body Methods (Fast Multi-pole method)
Sparse Linear Algebra
Spectral Methods (FFT)
Structured Grids
Unstructured Grids

Motifs

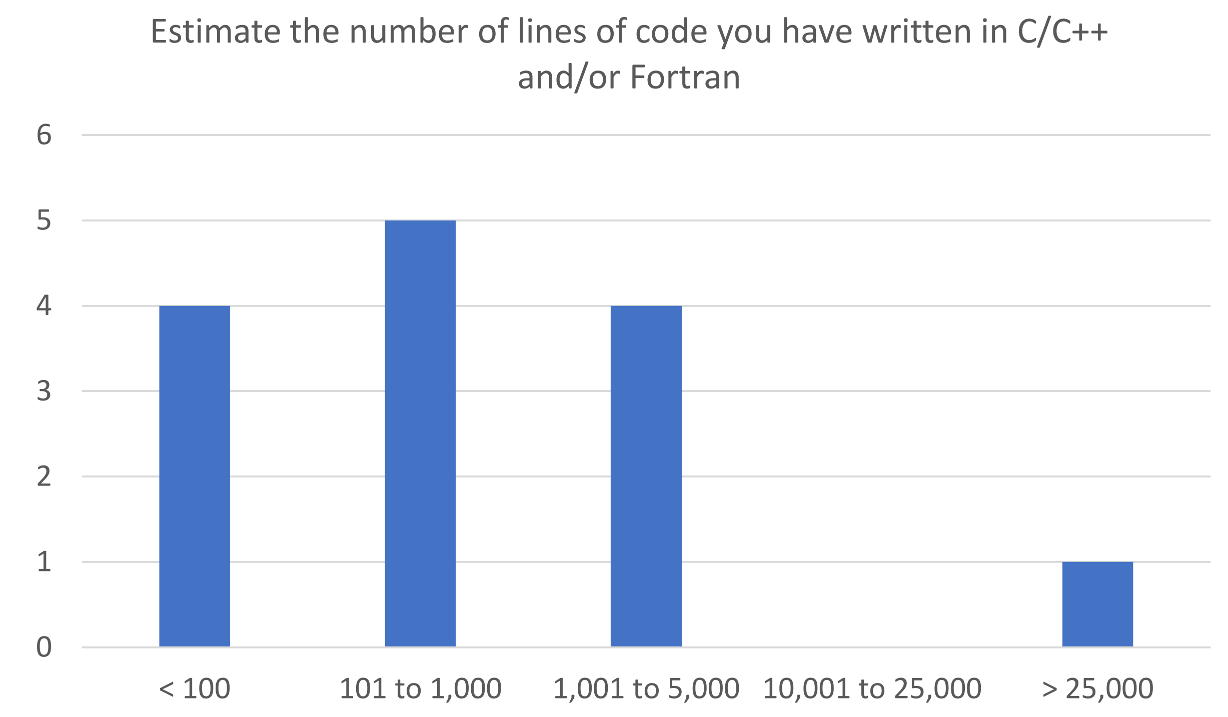
2021



2020

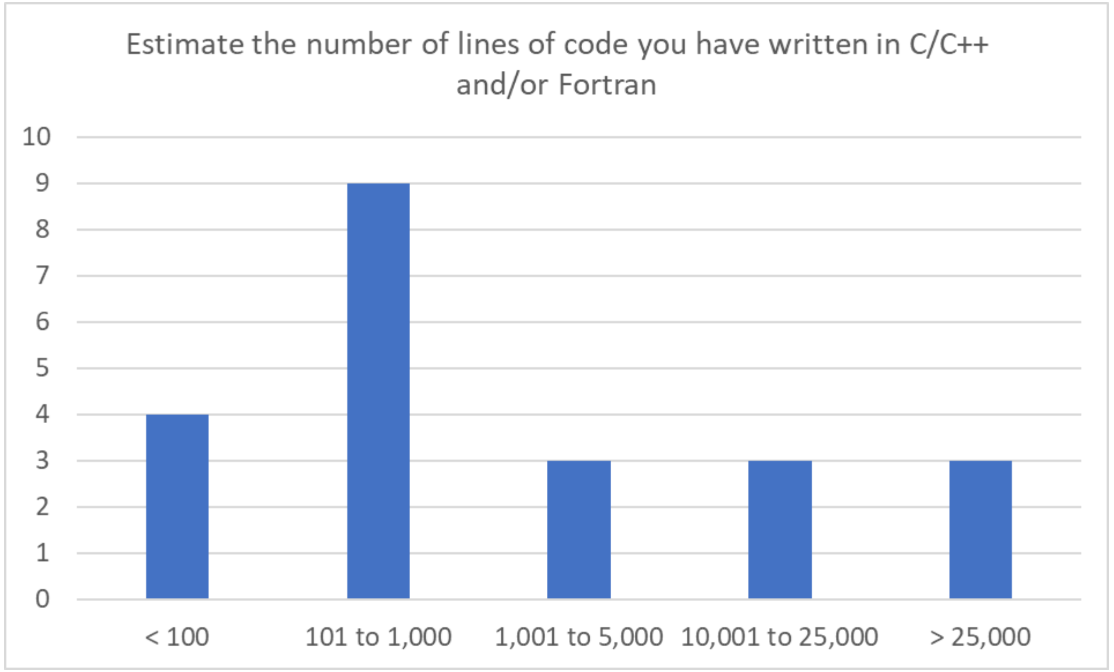


Lines of Code

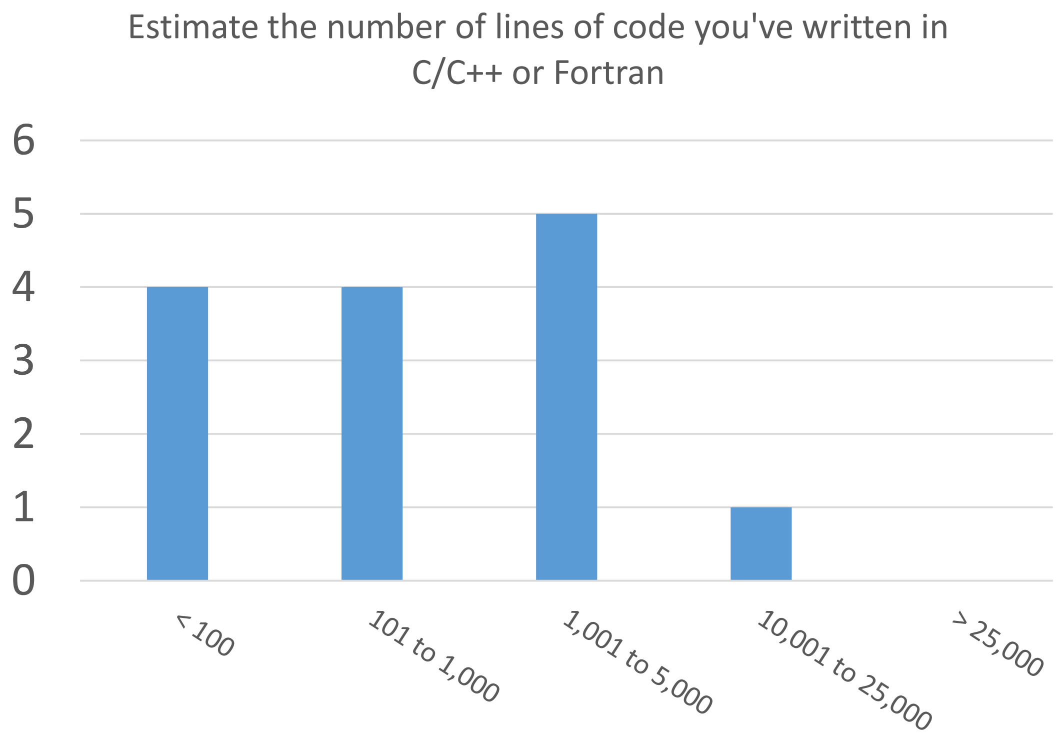


Lines of Code

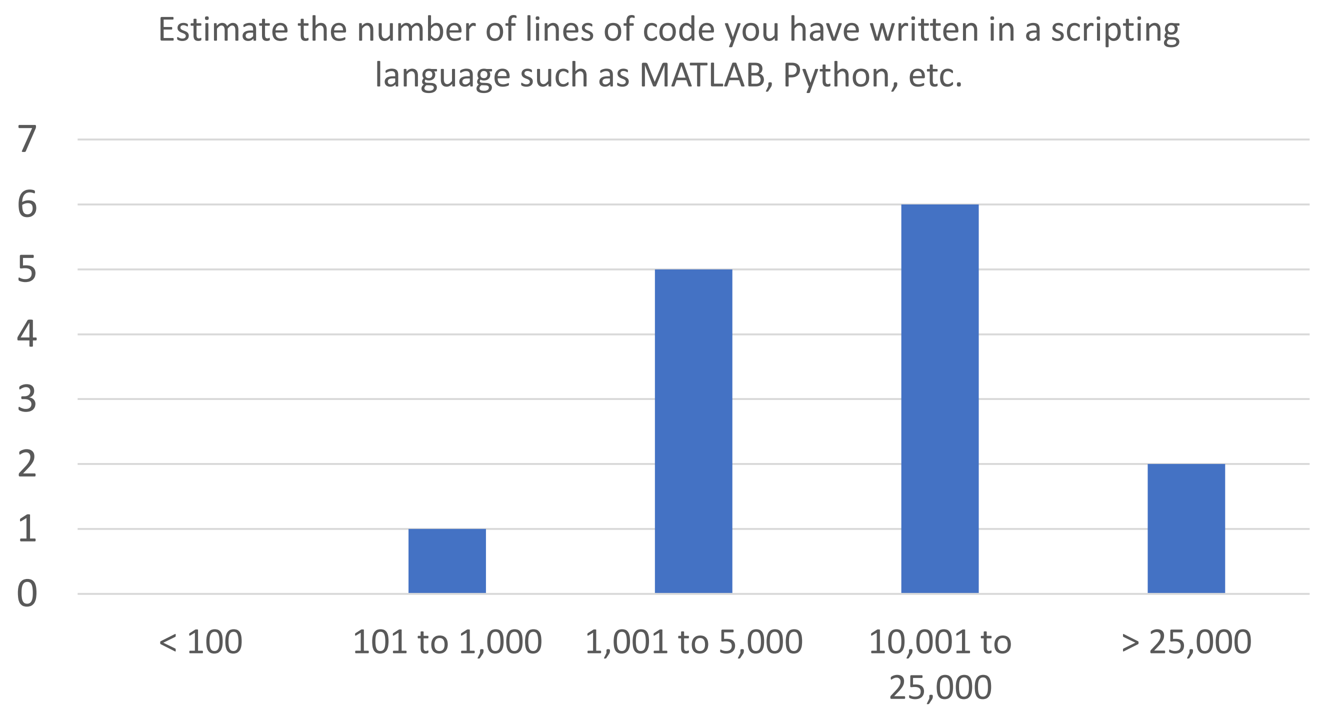
2021



2020

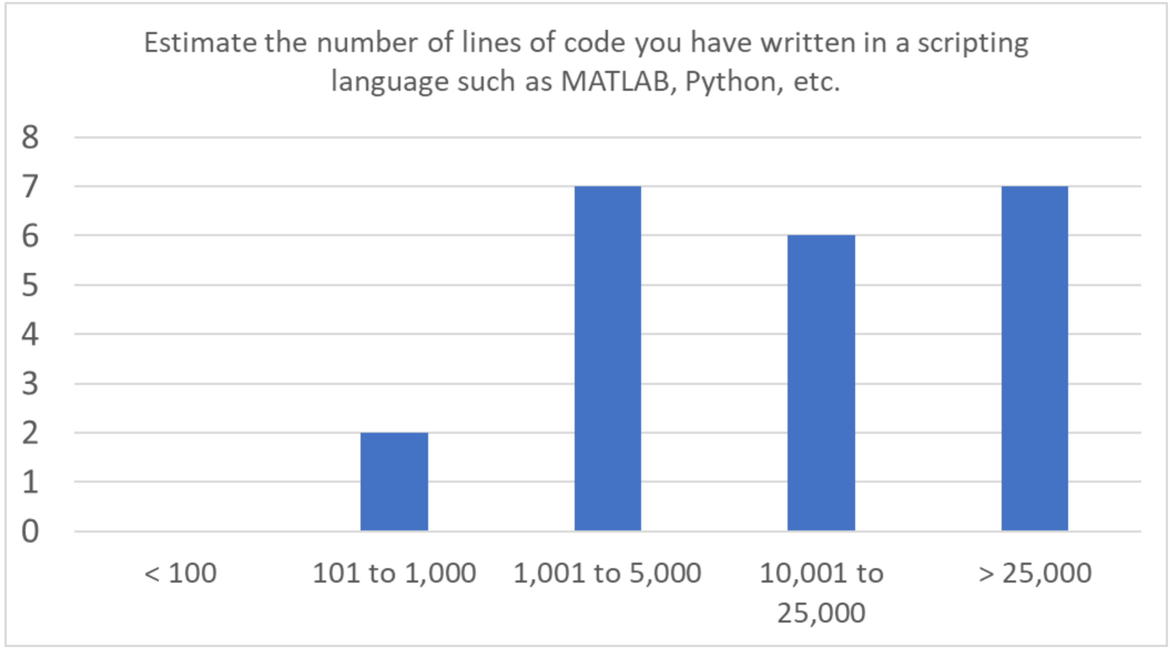


Scripting Lines of Code

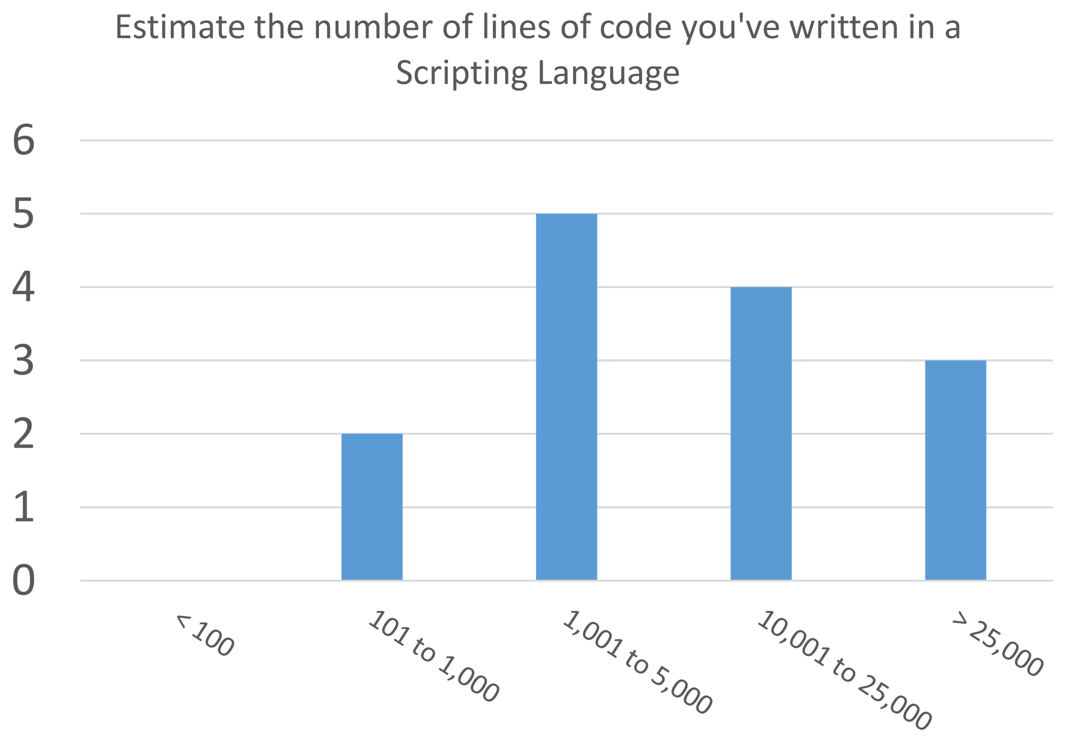


Scripting Lines of Code

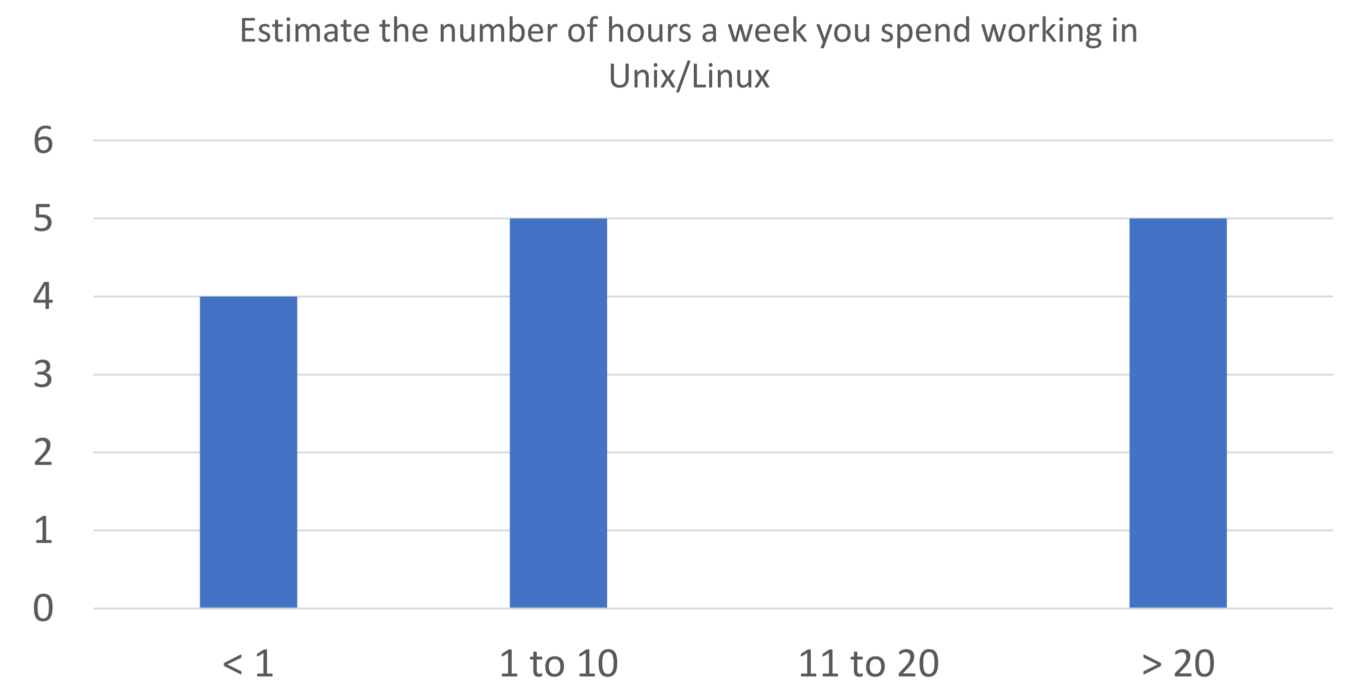
2021



2020

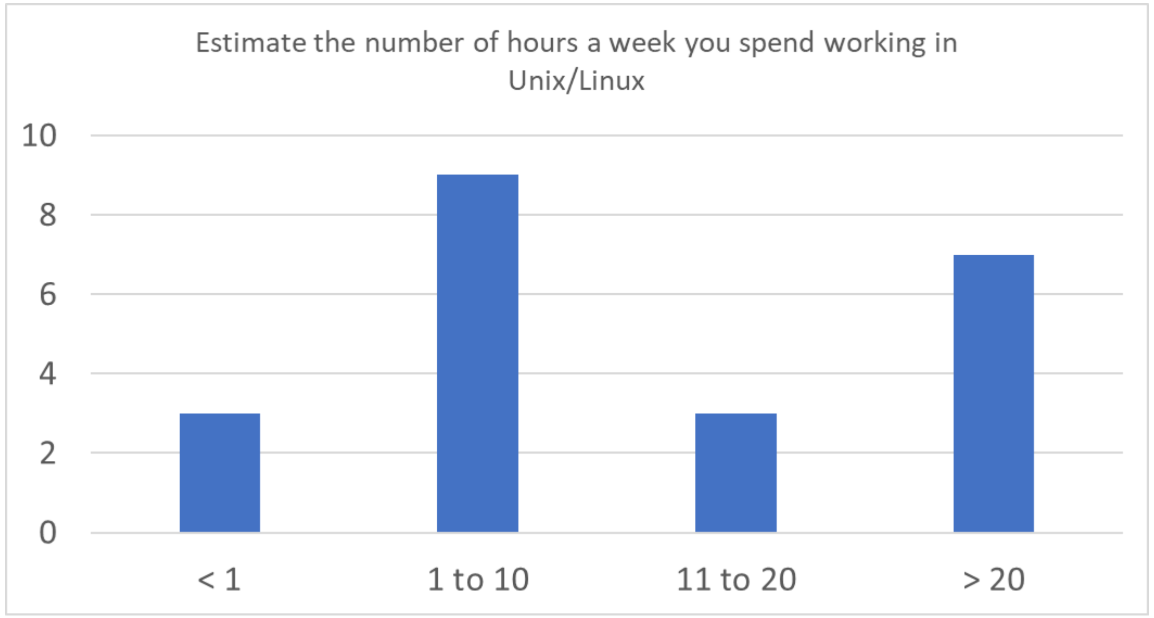


How much you Linux?

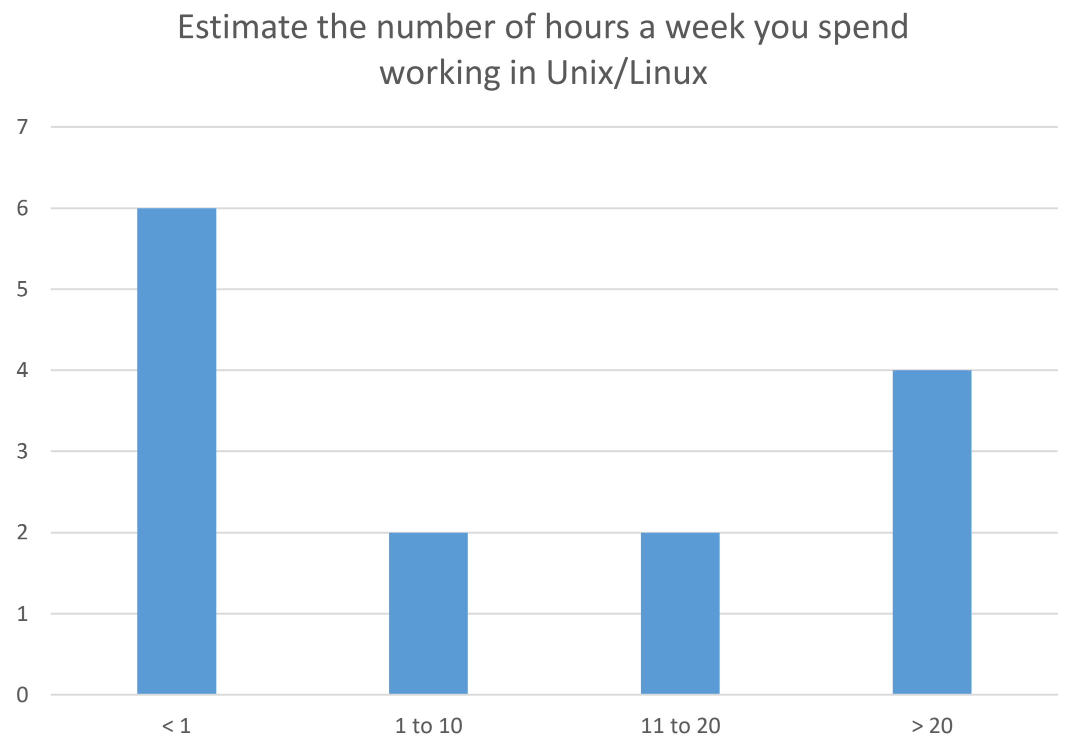


How much you Linux?

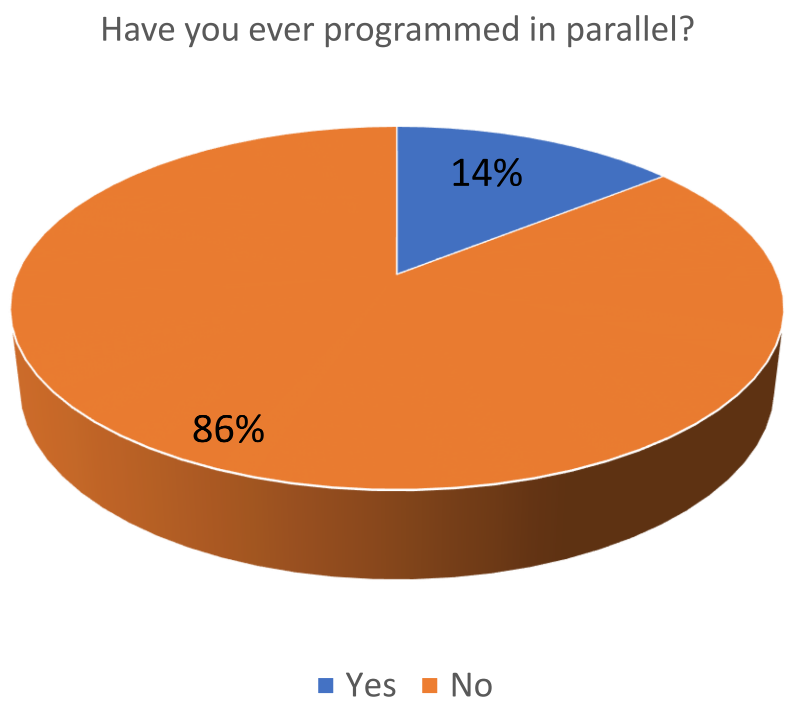
2021



2020

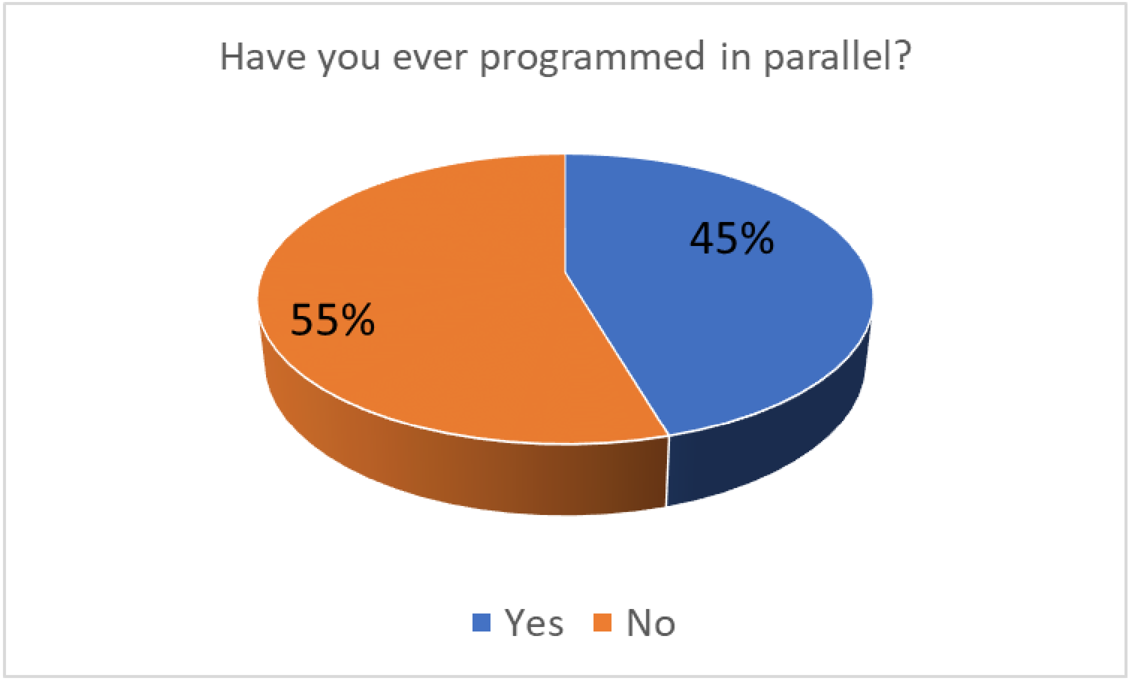


Have you ever programmed in parallel?

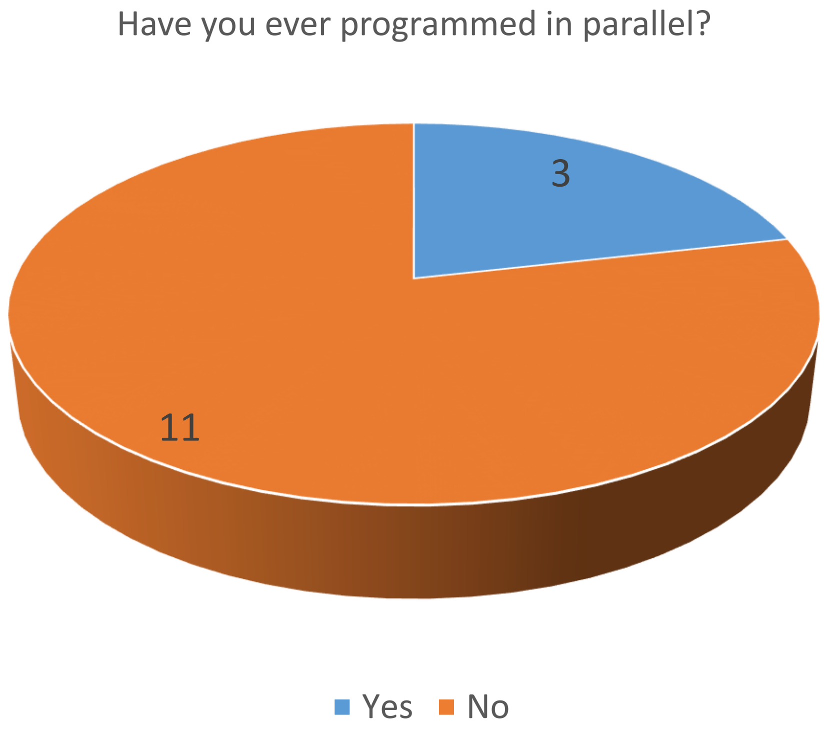


Have you ever programmed in parallel?

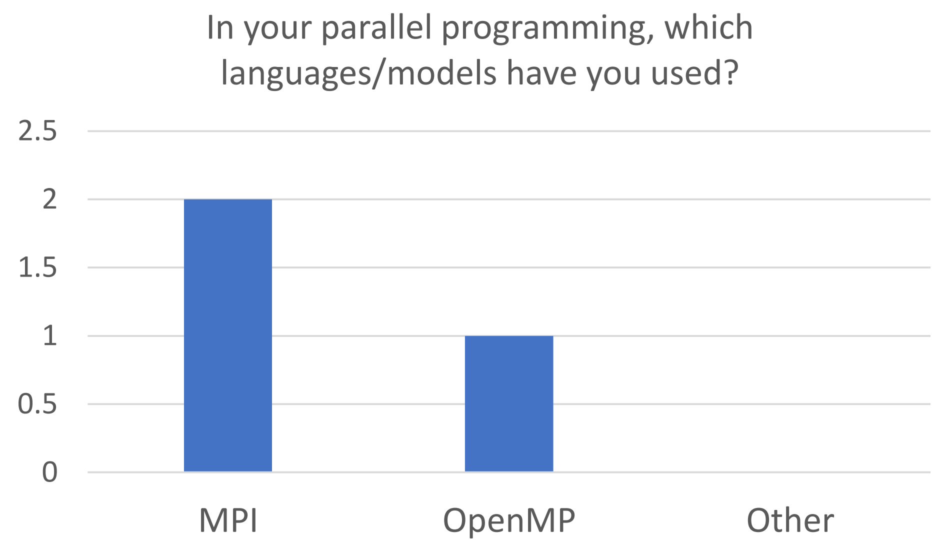
2021



2020

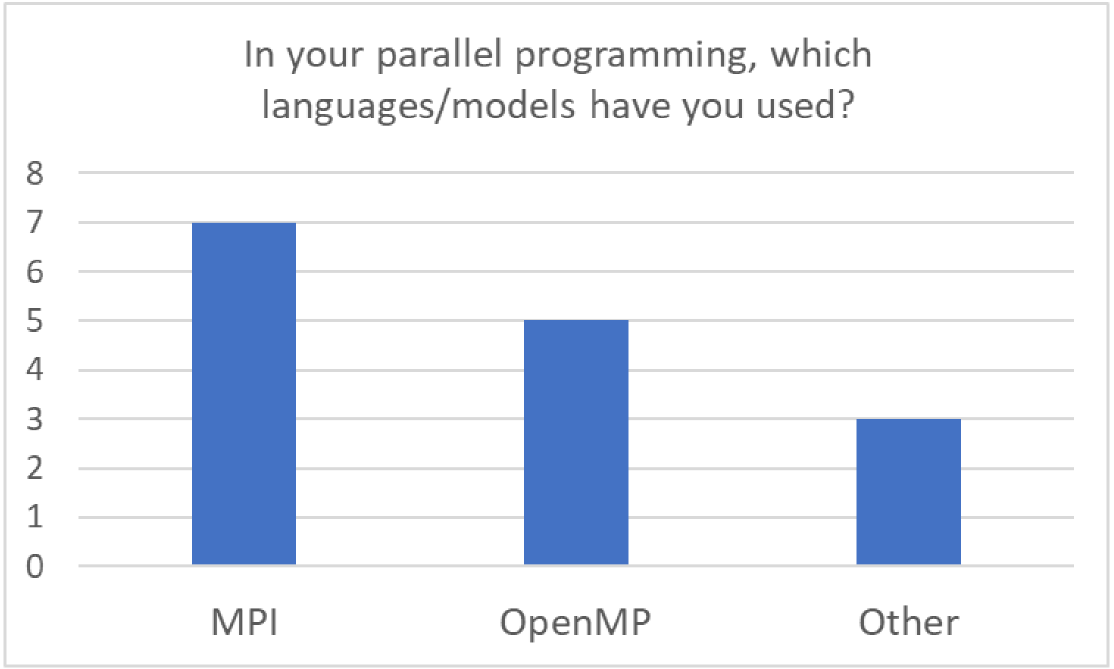


How do you parallel?

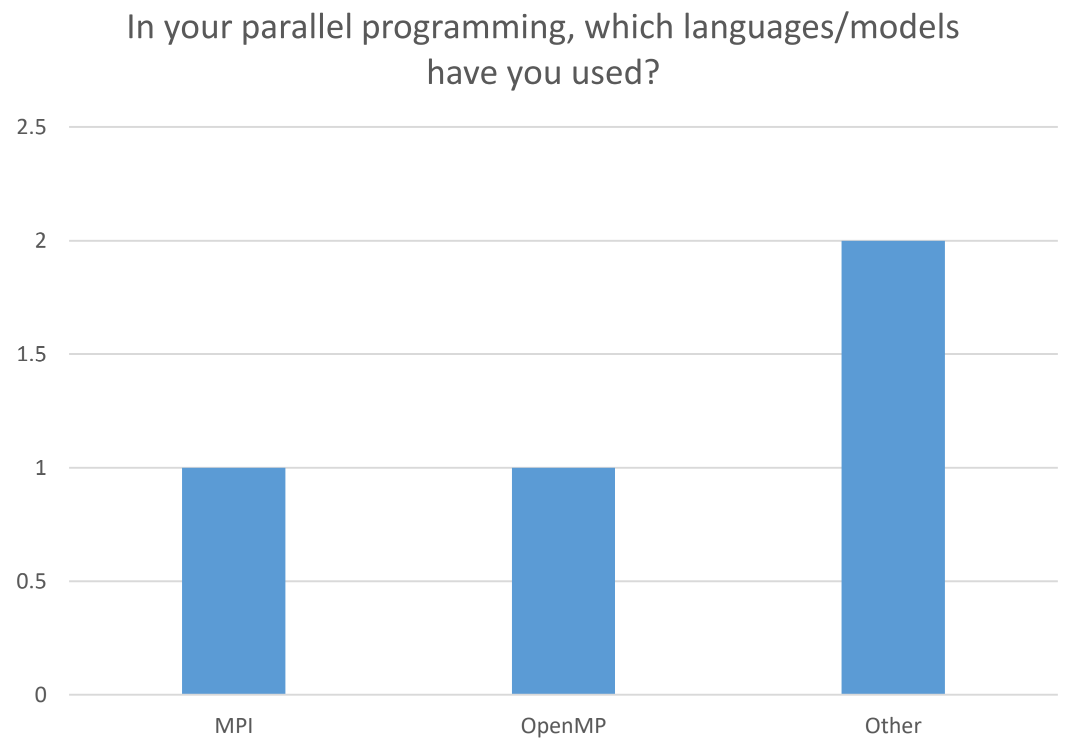


How do you parallel?

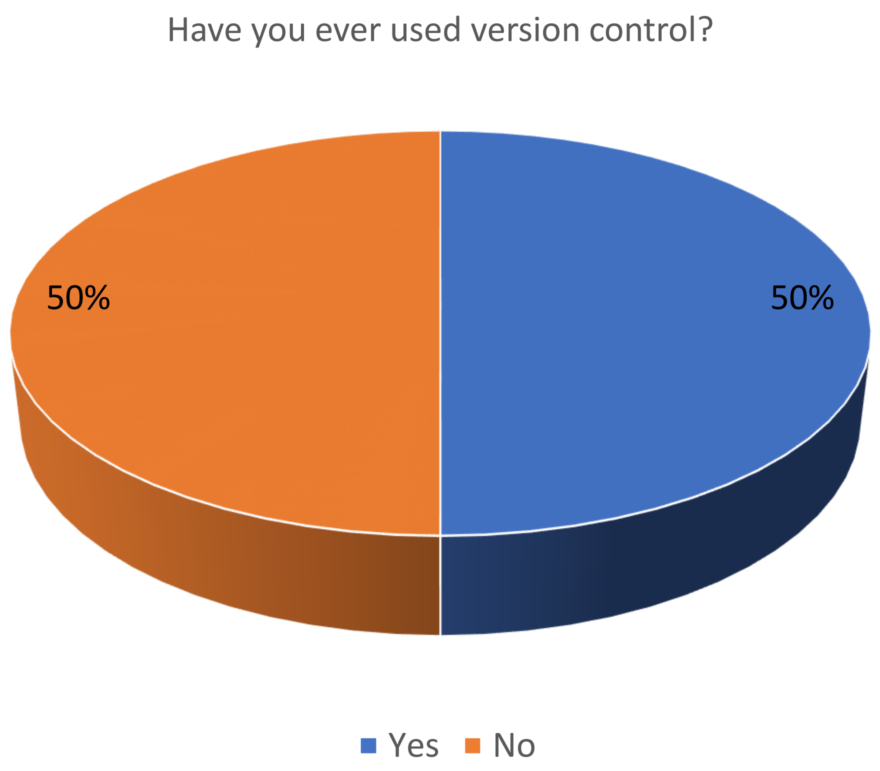
2021



2020

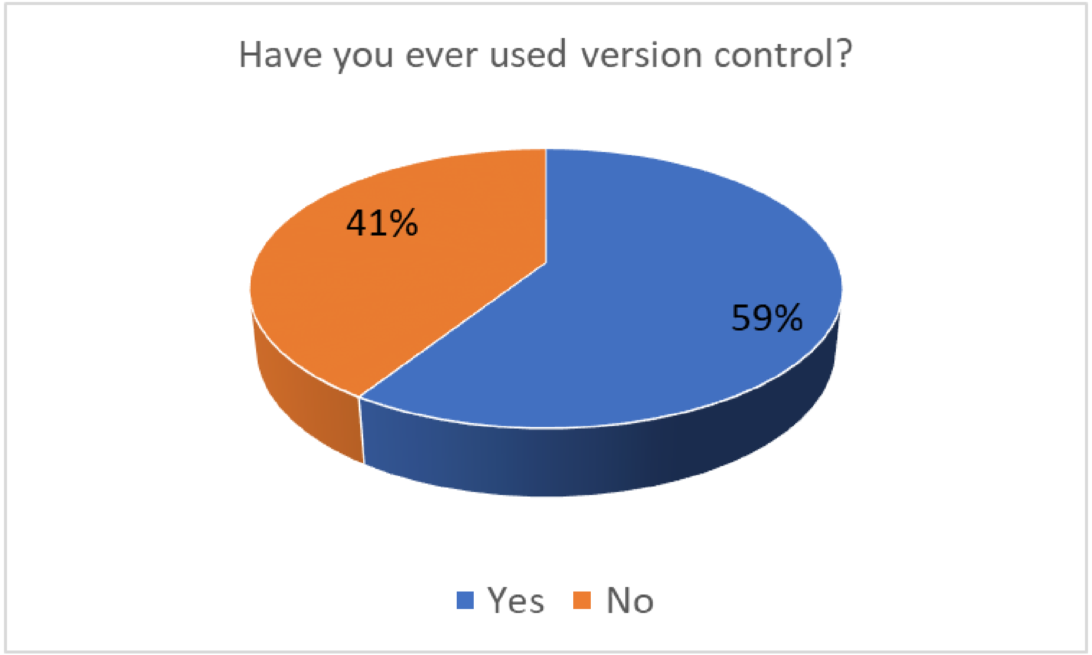


Version Control?

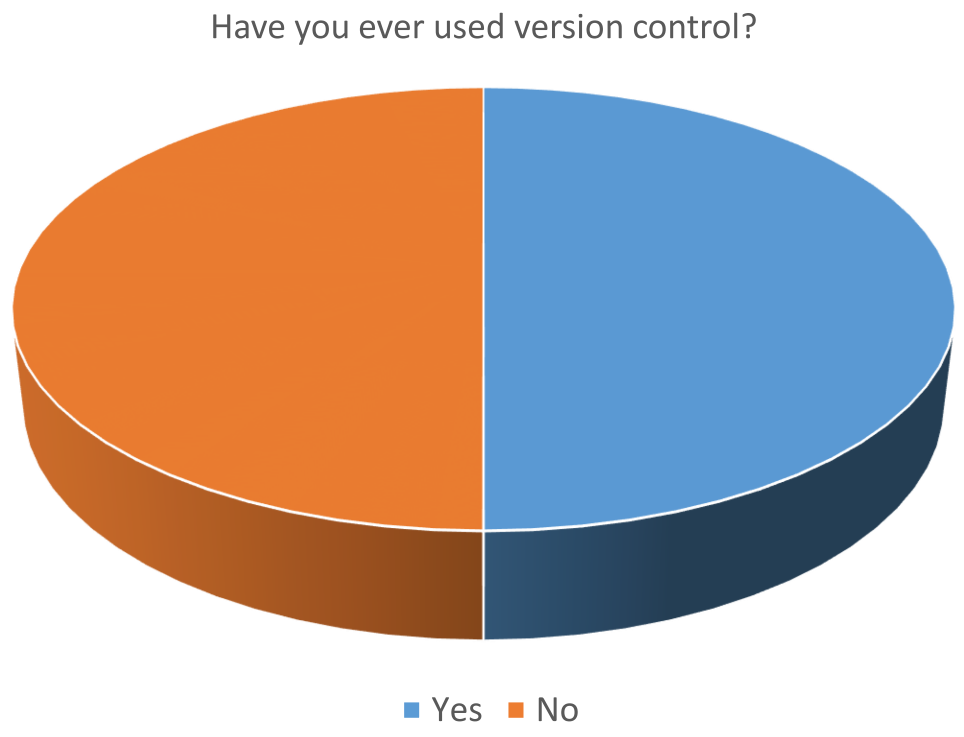


Version Control

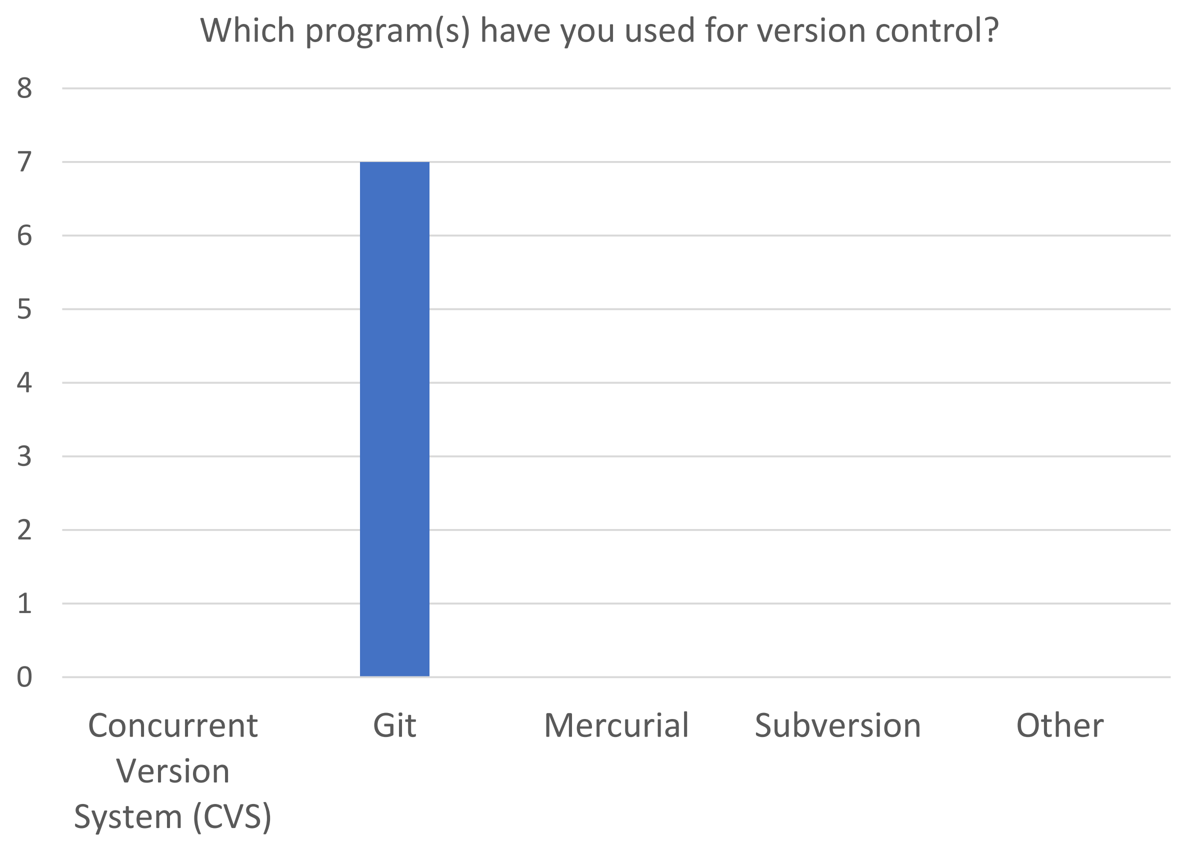
2021



2020

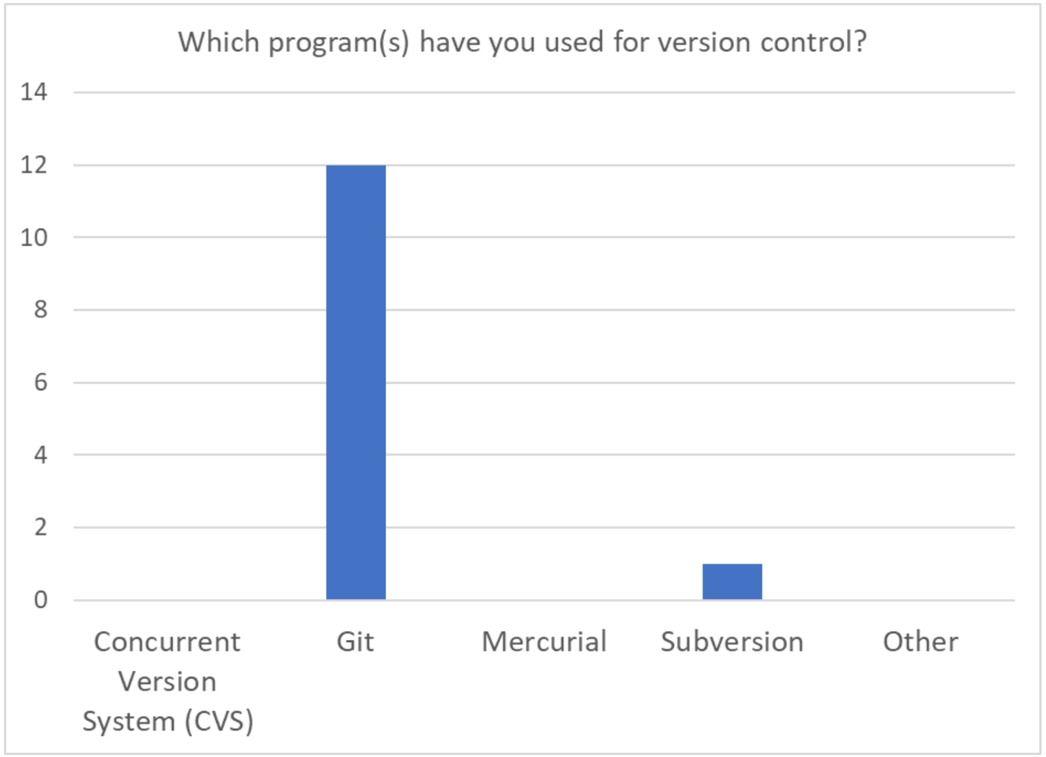


Version Control Programs

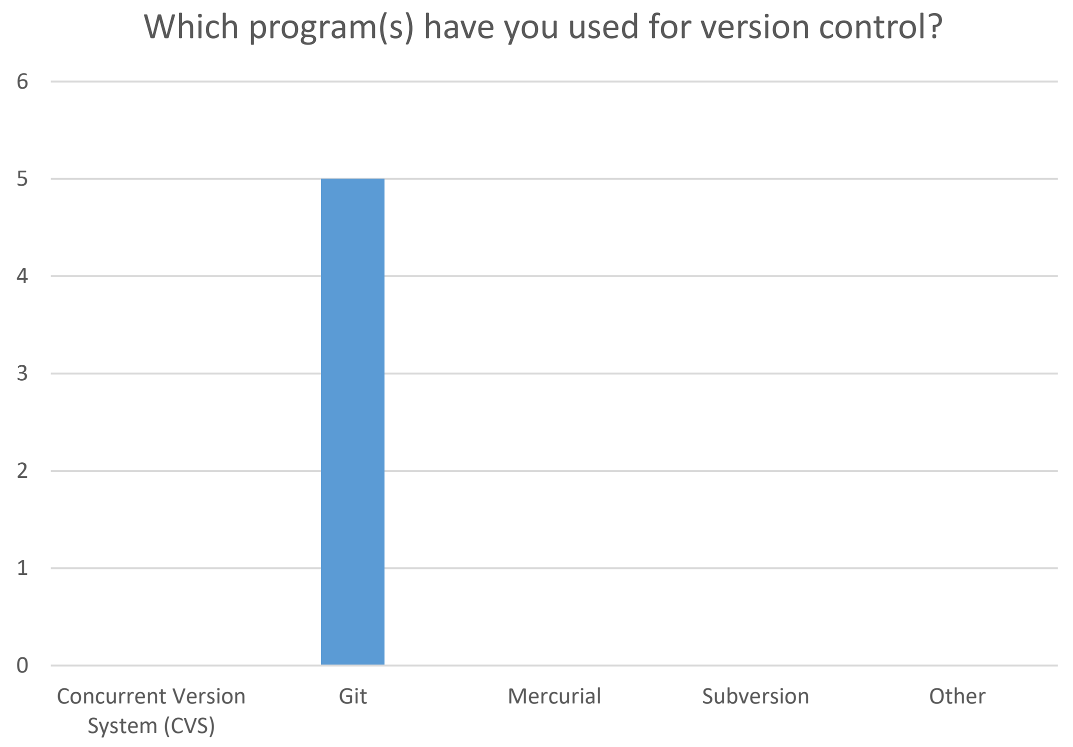


Survey Says..

2021

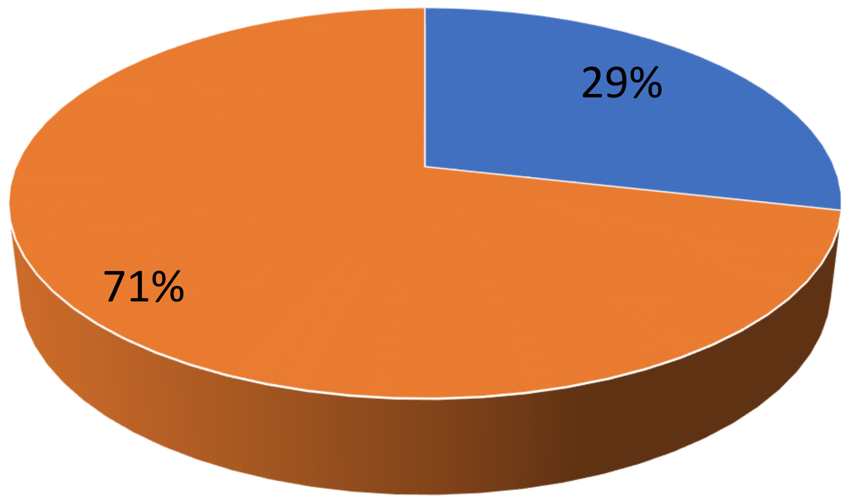


2020



Testing

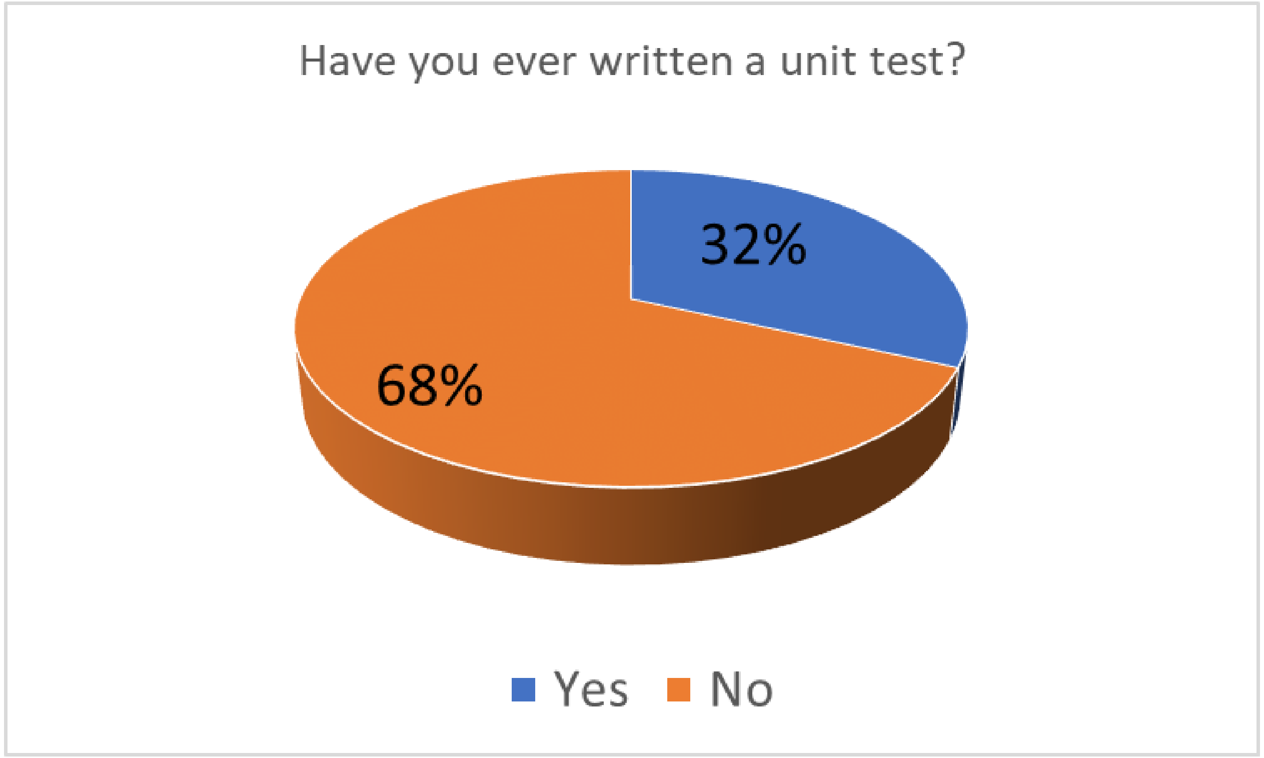
Have you ever written a unit test?



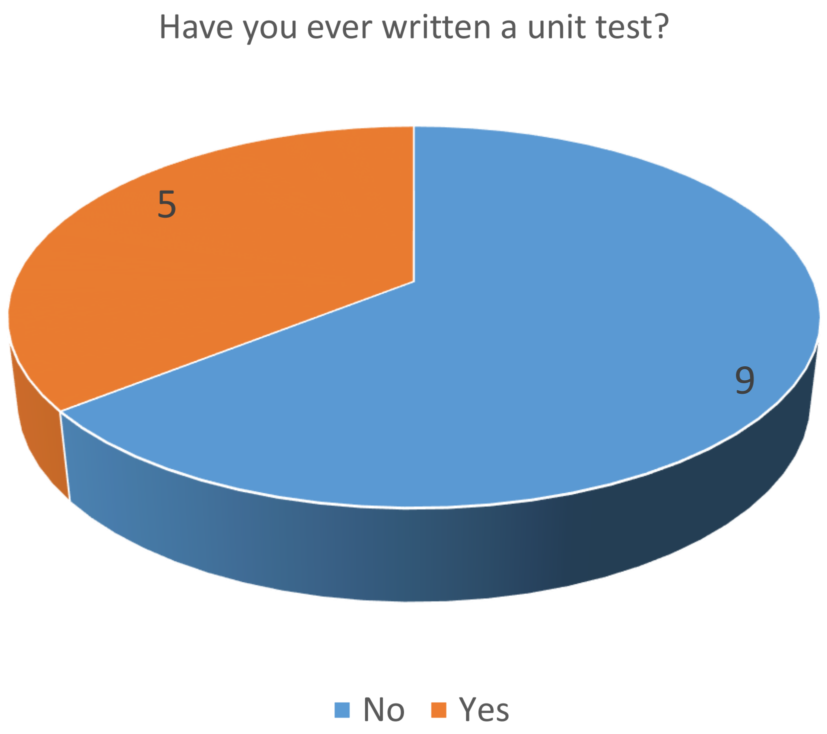
🟦 Yes 🟧 No

Survey Says…

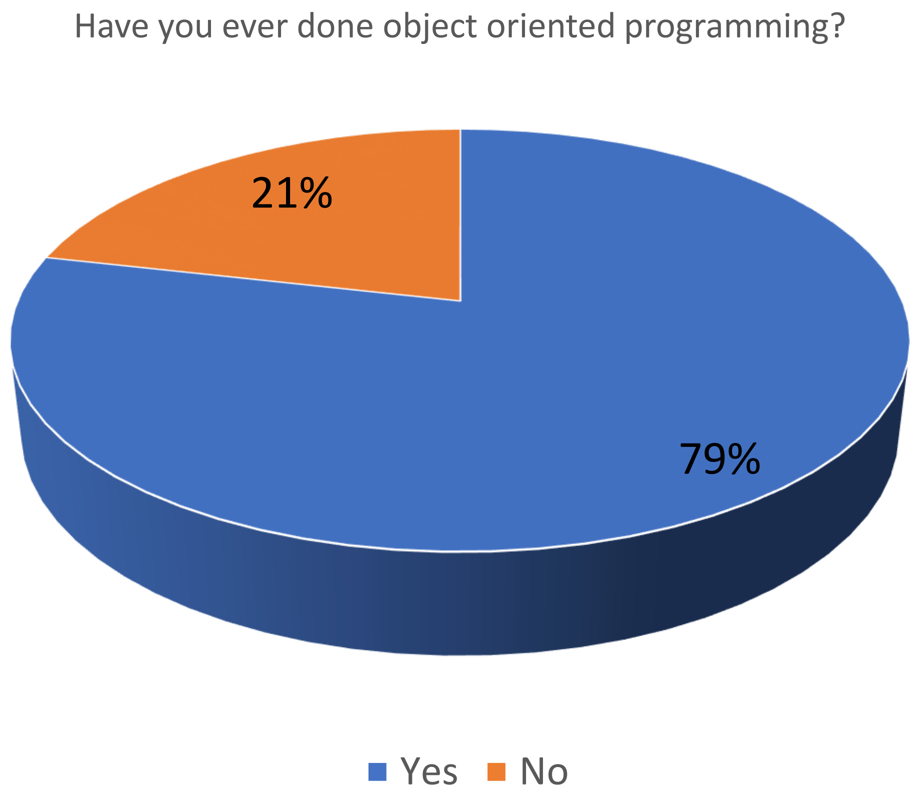
2021



2020

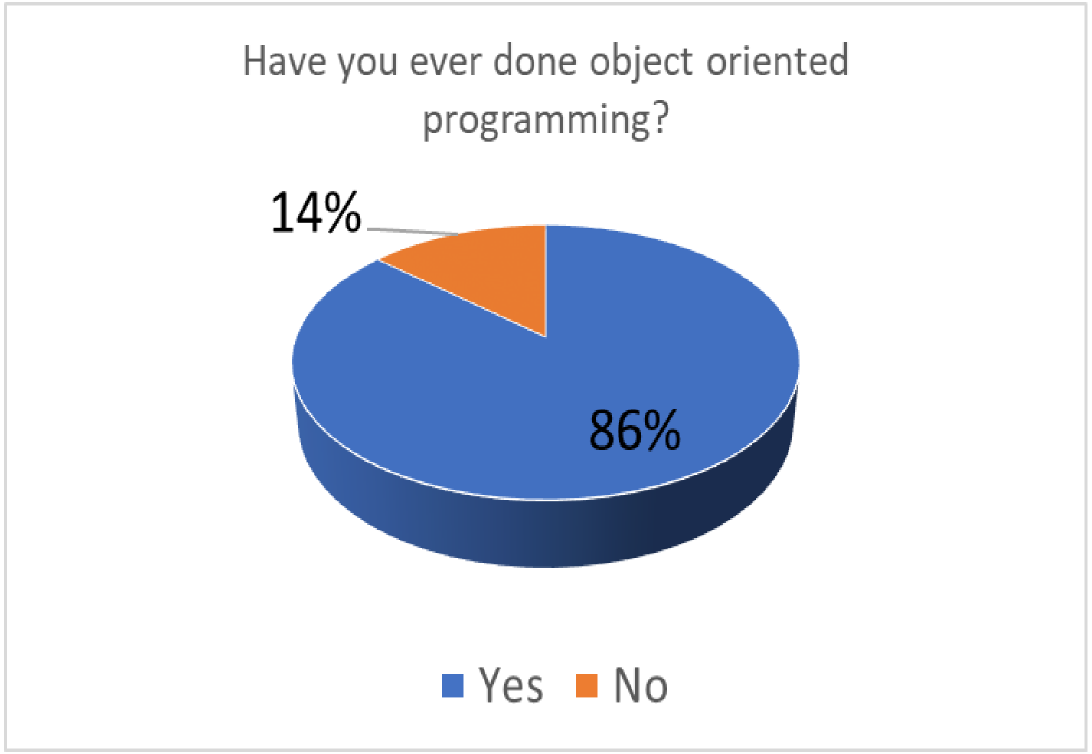


Object Oriented Programming

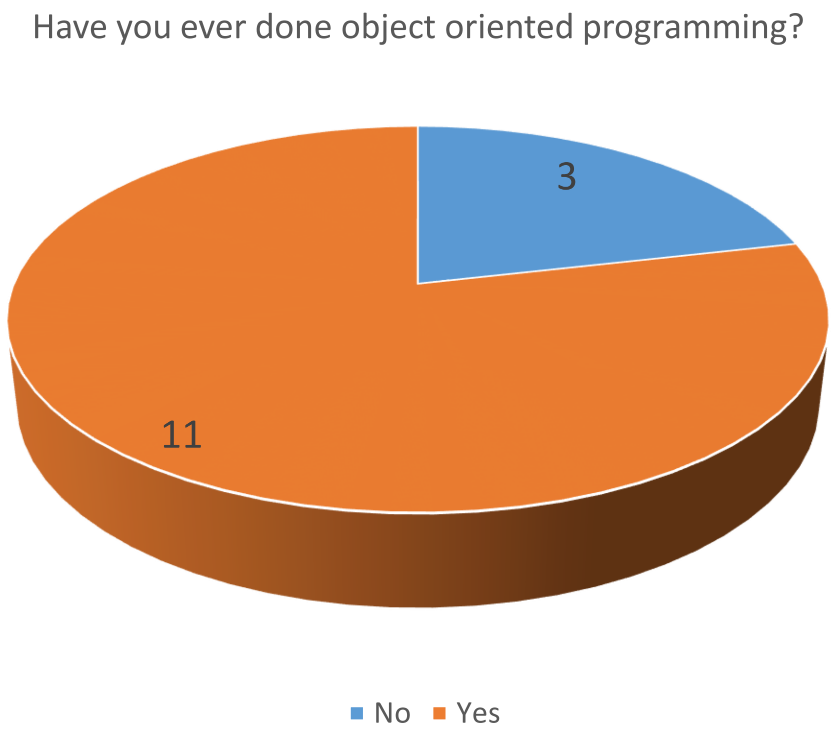


Survey Says…

2021

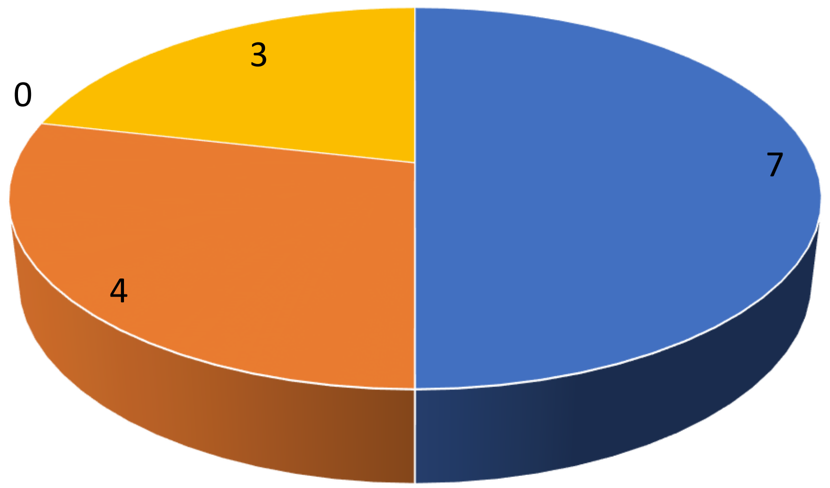


2020



Largest Project

What is the largest software project, in terms of active developers, that you have worked on?

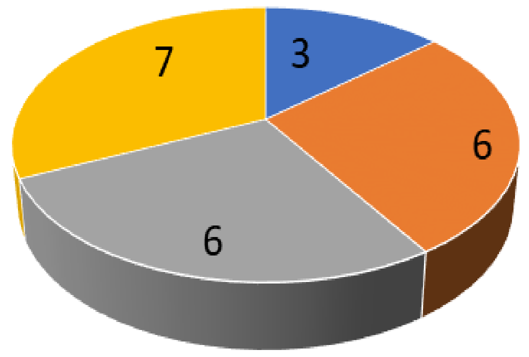


🟦 1 (myself) 🟧 2 ⬜ 3 to 5 🟨 > 5

Largest Project

2021

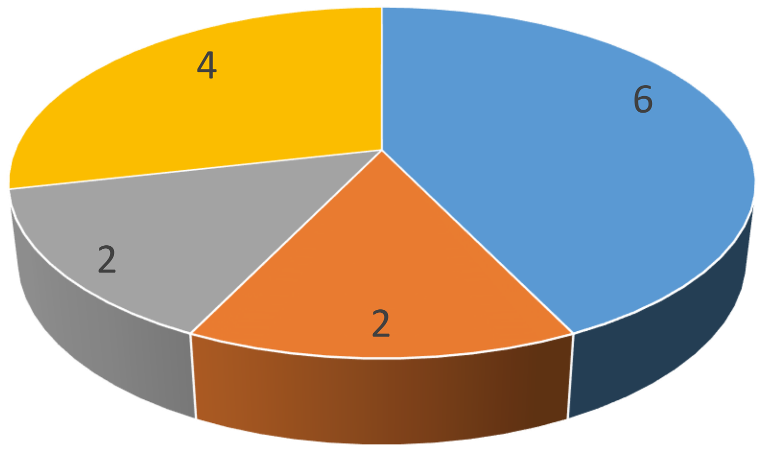
What is the largest software project, in terms of active developers, that you have worked on?



■ 1 (myself) ■ 2 ■ 3 to 5 ■ > 5

2020

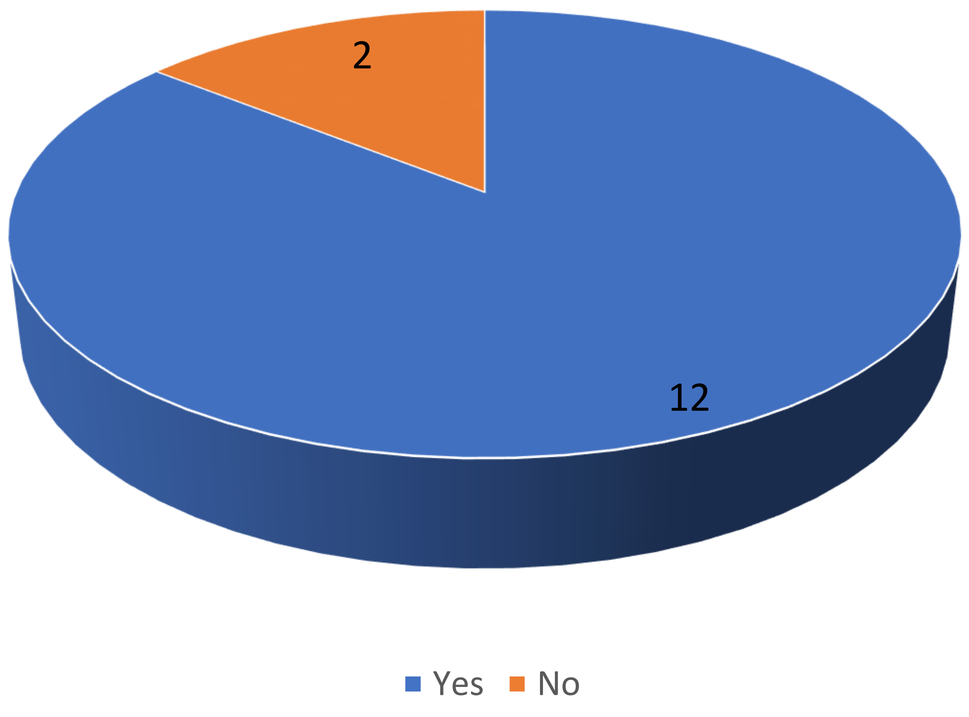
What is the largest software project, in terms of active developers, that you have worked on?



■ 1 (myself) ■ 2 ■ 3 to 5 ■ > 5

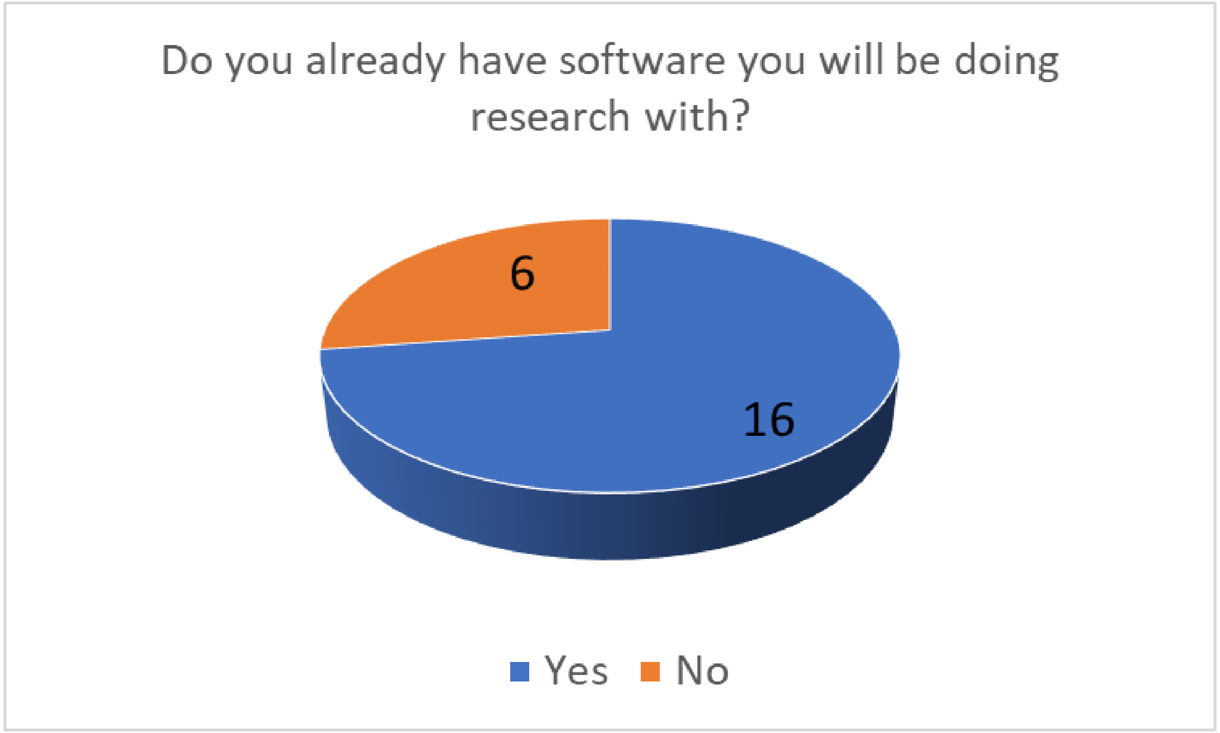
Do you have a research project?

Do you already have software you will be doing research with?

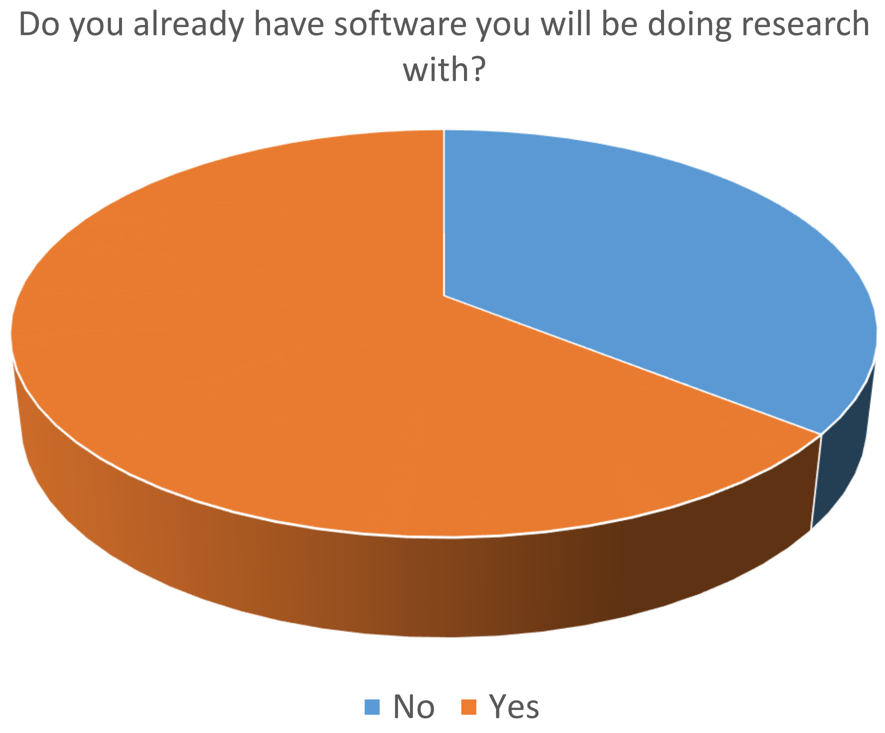


Do you have a research project?

2021

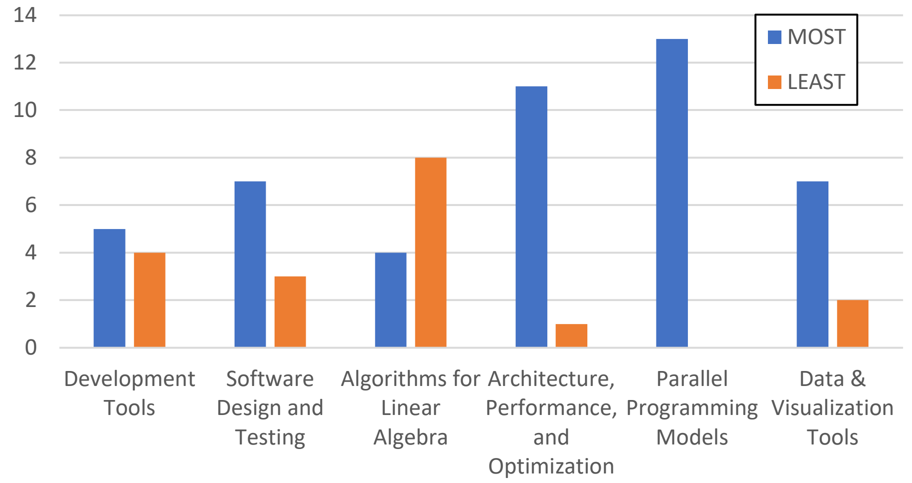


2020

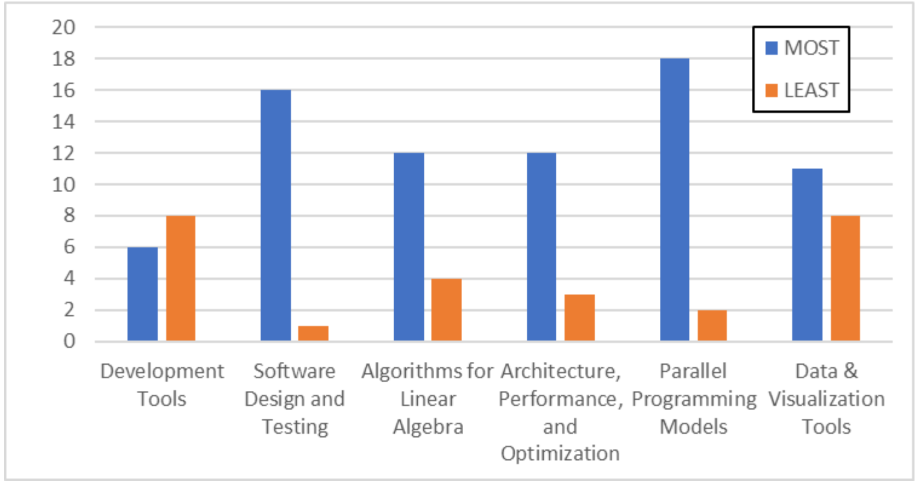


Most/Least Interested Topics

2022

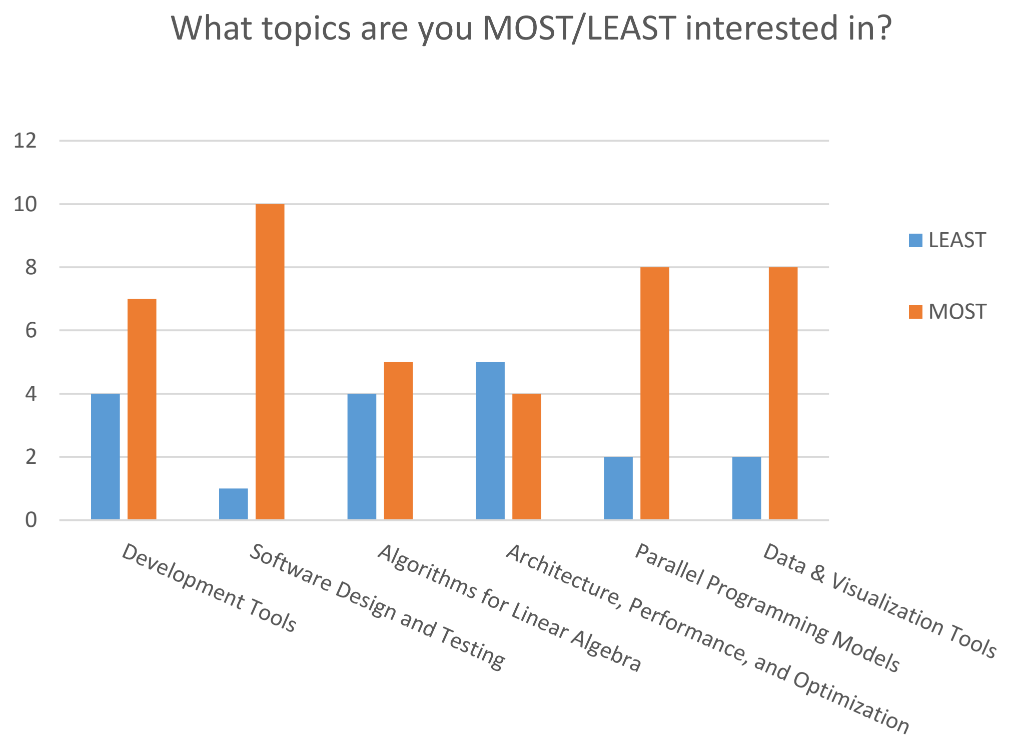


2021



- Topics not listed
  - Algorithms for digital signal processing
  - Computing for data science, computing with python
  - Some Machine Learning perspective as well
  - Image related topics

2020 Topics



- What other topics?
  - Machine Learning
  - GPUs
  - Pair Programming
  - Portability
  - Working on an HPC

What do I want to get out of this course?

- Gain more confidence in coding from scratch for my own computational projects
- learned formally the most efficient way to code, more efficient coding habits/version control, the ability to understand others' code better, and the ability to better troubleshoot computer programs having issues on my linux machine.
- Right now I feel like I "coast by" when it comes to the hard software skills required to get things to work for my research. Even installing packages can test my knowledge base, so I'd like to have a better understanding of computation from a less abstract standpoint.
- Learn about Fortran, proficiency in C++, understand MPI and OpenMPI, learn things that are beneficial to my researches

What do I want out of this course?

2021

- I have a lot of experience writing code, but not actually caring about optimization, readability, functionality/use-ability. Those are all things I would like to improve on.
- Learn OpenMPI and Fortran (properly this time), better software development practices
- I want to gain a deeper understanding of all the tools I use already (i.e linux, compilers, makefiles,etc) but also gain skills such as object oriented programming, testing, and git/github.

2020

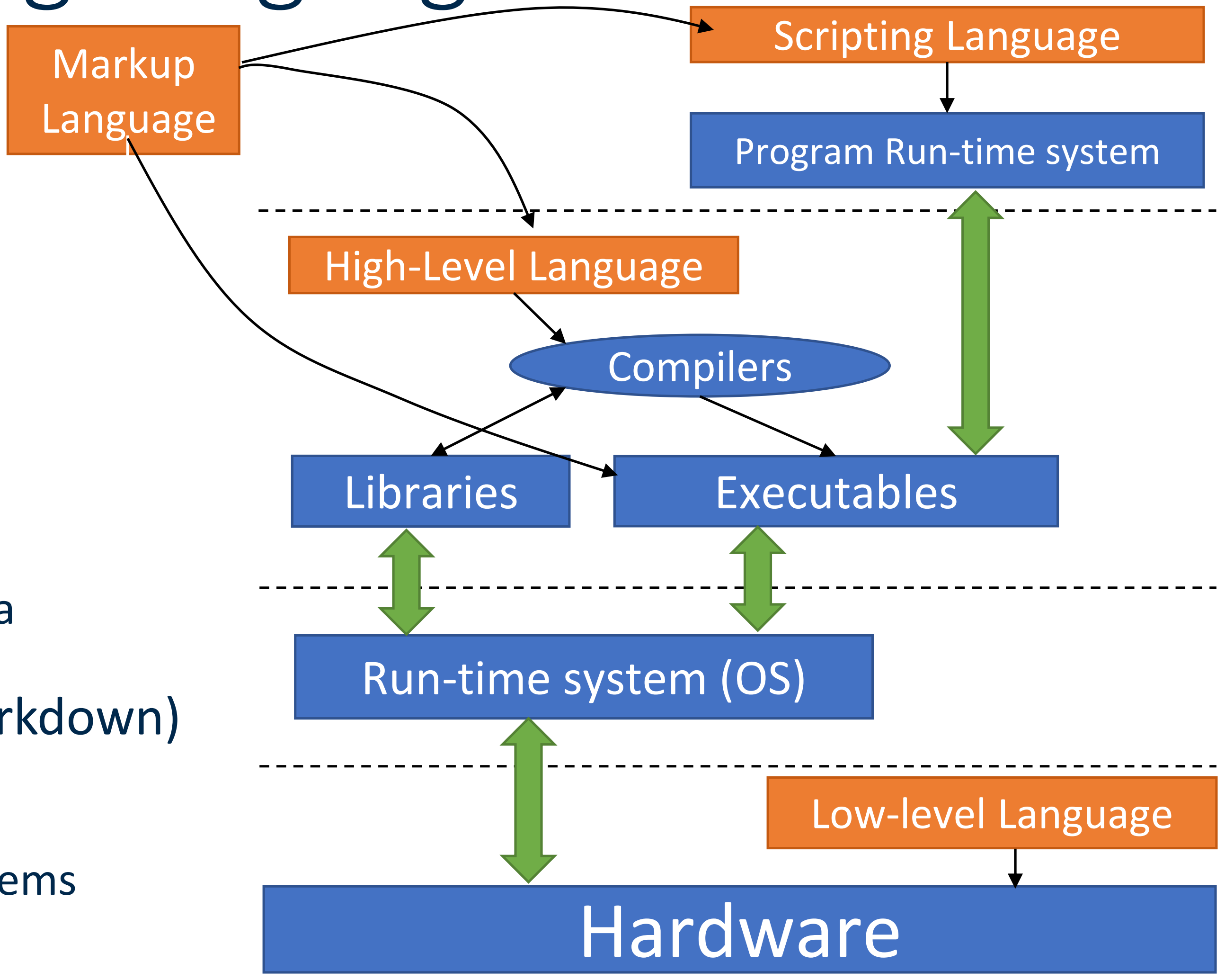
- Use HPC
- Become a better/more efficient programmer
- Become more aware of tools
- Work on large projects

Prepare for a continuation of Hands on!

1. Open a Linux terminal

Types of Programming Languages

- Low Level language (Assembly)
  - Defined by hardware (less portable)
- High Level language (C/C++, Fortran, Java)
  - Defined by run-time system (e.g. Operating System)
  - Portable, depends on compilers
- Scripting language (MATLAB, Python, Bash)
  - Defined by portable run-time system of a program
- Markup language (e.g. XML, YAML, Markdown)
  - Used for annotation
  - Data transfer
  - Input to multiple types of programs/systems



Shell Scripts in Linux

- Scripts are just text files
- First line is special

Bash Hello World

```
#!/bin/bash

echo "Hello World!"
```

Python Hello World

```
#!/usr/bin/python

print("Hello World!")
```

Perl Hello World

```
#!/usr/bin/perl

print("Hello World!\n");
```

Awk Hello World

```
#!/usr/bin/awk -f
BEGIN {
  print "Hello World!"
}
```

Symbol is sometimes called "bang"

Symbol is sometimes called "sha"

Program for interpreting the script

Together it is called "Shebang"

```
#!/usr/bin/python

print("Hello World!")
```

Scripting with Bash

More documentation at Too Long Didn't Program (tldp)

http://tldp.org/LDP/abs/html/index.html

Bash: The Basics

- The contents of your bash file will also work if entered correctly on the command line*

```
#!/bin/bash

# A comment

#print something
echo “Hello World”

#set a variable
myvar=1

#access a variable
echo $myvar
```

```
#!/bin/bash

# For loop
for i in 1 2 3 4; do
  echo $i
done
```

```
#!/bin/bash

# if else
touch newfile
if [ -e  newfile ]; then
  echo “Found newfile”
else
  echo “Did not find newfile”
fi
```

*The exception here is script variables.

Pitfalls

```
#!/bin/bash

# A comment

#print something
echo “Hello World”

#set a variable
myvar=1

#access a variable
echo $myvar
echo “$myvar”
echo ‘$myvar’
```

No whitespace around assignment operator!

Single quote and double quote matter!

Variables defined in script do not persist after script!

Must have whitespace!

```
#!/bin/bash

# if else
touch newfile
if [ -e  newfile ]; then
  echo “Found newfile”
else
  echo “Did not find newfile”
fi
```

Test command “[“ is very confusing

```
$ var=''
$ [ $var = '' ] && echo True
-bash: [: =: unary operator expected
$ [ "$var" = '' ] && echo True
True
$ [[ $var = '' ]] && echo True
True
```

Summary advice on Bash Scripts

When to use

- Simplifying complex commands
- Modifying files and directories
- Simple parsing and modification of file contents
- Working with operating system
- Generating inputs and jobs for parametric studies
- Some software project infrastructure tasks

When not to use

- Arithmetic
  - Especially floating point
- Data processing
  - e.g. parsing CSV files, finding max values, min values, averages
- Graphics (should be obvious)

Homework 1

Introduction to Python

It is the number 1 used programming language!

It is FREE!

It can do most anything other languages can do.

It is comparatively easier to use than high level languages.

There’s a package for just about everything!

Executing python

- Interactively
  - Start a python “shell”: `$ python`
  - To exit the python “shell” enter: Ctrl+d
- Run a script
  - `$ python myscript.py`
- Run a script interactively
  - `$ python -i myscript.py`

Basic Syntax and Operators

Arithmetic Operators

```
print('Hello world')

x = 10.

print(x + 5.)
print(x - 4.)
print(x * 2.)
print(x / 3.)

print(x ** 2.)
# (x^2 in MATLAB)
```

Boolean Operators

```
x = 10.
# comparison
print(x == 15.)
print(x <= 12.)
print(x != 5.)

# boolean logic
print(not True)
print(True and False)
print(True or False)
```

Intrinsic types

- Python uses “dynamic typing”
  - No need to explicitly declare `int`, `float`, `bool`, etc.
  - Types are *implied*

```
x = 10.            #implied float
i = 1              #implied int
l = True           #implied bool
c = “Hello World” #implied string
```

```
#Python 2
2/3 == 0
2./3. == 0.66667

#Python 3
2/3 == 2./3. == 0.66667
2//3 == 0
```

- Declare variables anywhere!

Execution Control Constructs

If-Else

```
# indentation required:
if condition1:
    statements
elif condition2:
    other statements
elif condition3:
    other statements
else:
    alt statements
```

Looping

```
# i = 0, 1, ..., n-1
for i in range(n):
    print(i)

# iterate through a
# sequence
x_list = [1, 2, 4, 8]
for x in x_list:
    print(x)
```

Python Lists

- Effectively the implementation of arrays
  - Can be mixed "types"
- But these arrays are fancy
  - Can function like the classical data structure definitions of stacks, queues, or decks
- Operations on lists are very easy compared to other languages.
- Not very fast…
  - For speed you want numpy.

```
VY0 = [2., 3., 4.]
print(VY0)
print(VY0[0],VY0[-1])
print(len(VY0))

VY0.append(5.)
VY0[2] = "cat"
print(VY0)

VY0.pop(0)
print(VY0)
```

Further Reading

- Numpy and Scipy
  - https://docs.scipy.org/doc/
- Gorelick and Ozsvald, “High Performance Python,” O’Reilly Media, 2014.
  - https://search.lib.umich.edu/catalog/record/018003128
- MOOC: https://www.sololearn.com/Course/Python/
- A great new tool for productivity is the Jupyter Notebook
  - https://jupyter-notebook-beginner-guide.readthedocs.io