

# EECS 168 – Programming I Lab



## LAB 01: INTRO

**Instructor: Arman Ghasemi**

Fall 2024



- TA : Arman Ghasemi
- Email : [arman.ghasemi@ku.edu](mailto:arman.ghasemi@ku.edu)
- Office : 2025
- Office Hour : Wednesday 1:00 PM – 3:00 PM

# Lab Conduct

- Do not use any unauthorized aid, such as chatbots or chegg to obtain answers
- Do not use code provided by another student
- Do not reuse code (by you or anyone) from prior semesters
- If you need help, seek it from:
  - Your lab TA.
  - Dr. Gibbons, email is [jwgibbo@ku.edu](mailto:jwgibbo@ku.edu) / [jwgibbo@gmail.com](mailto:jwgibbo@gmail.com)
- If equipment you don't own (e.g. cycle servers) needs attention or you're having account issues put in a [ticket!](#)

# Objectives

- Get familiar with Python interactive shell
- Learn to make .py files
- Write and run a couple of simple programs
- Learn how to clone and commit code to a GitHub repository
- Submit your work via GitHub

Python is a high-level, interpreted programming language known for its simplicity and readability.

- **Live Shell**
- **.py Files**
- **Running Files**



# Python – Live Shell

- In Python programming, a **live shell** refers to an **interactive shell** that allows you to execute Python code in real-time and see the results immediately.
- **Use Cases for Live Shell:**
  - **Experimenting with code:** Test and debug small snippets of code without the need for a complete script.
  - **Prototyping:** Quickly try out ideas or algorithms to see if they work as expected.
  - **Data Exploration:** In environments like **IPython** or **Jupyter**, you can interact with datasets, visualize data, and manipulate it in real-time.

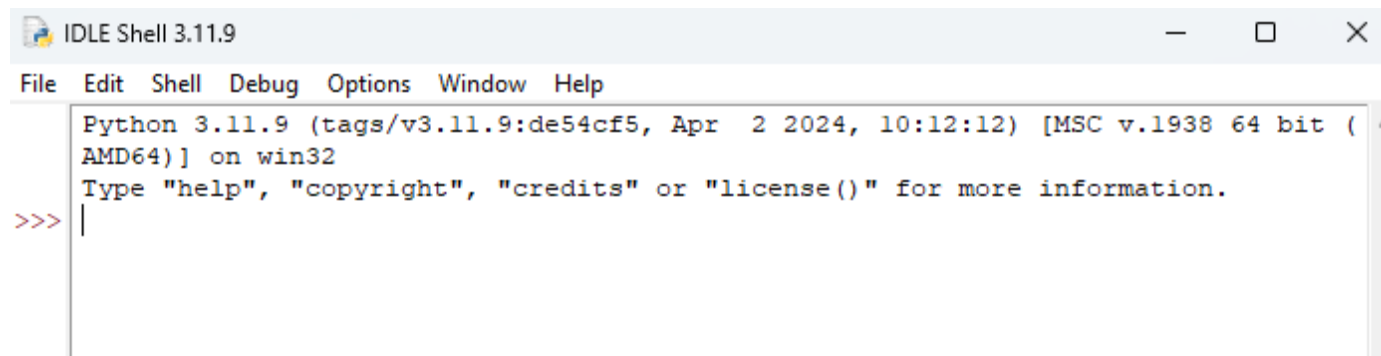
# Python – Live Shell

- To access the python shell on the lab computers, use the command `python3` in the terminal:

```
$>python3
```

- This will launch the interactive shell that you saw in lecture. Good for testing small amounts of code.

**Note:** Just like in lecture, you won't do your lab work here. You'll need to create `.py` files that you can save.

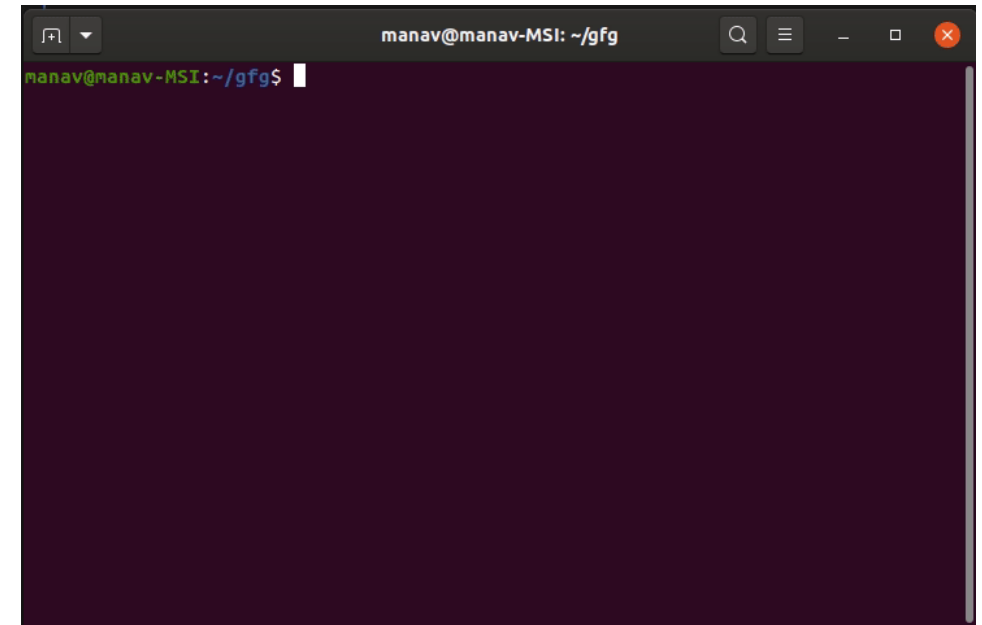


```
IDLE Shell 3.11.9
File Edit Shell Debug Options Window Help
Python 3.11.9 (tags/v3.11.9:de54cf5, Apr  2 2024, 10:12:12) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

# Python – .py file

## Writing code using “nano”

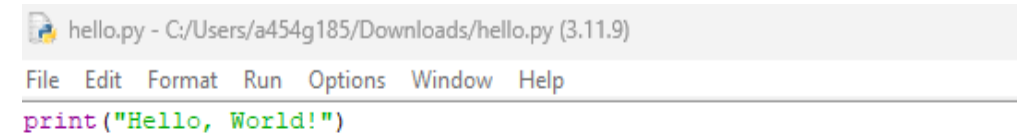
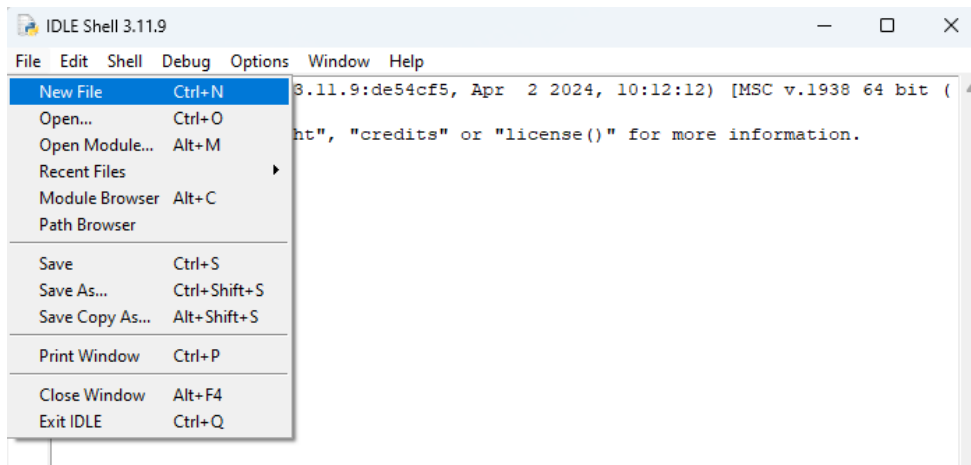
- Make a new file called **hello.py** to do this type the following command:  
**nano hello.py**
- This will open our trusty text editor in your terminal. You can use the arrow keys to move around (no mouse here). Some useful commands:
  - Save: CTRL+O
  - Exit: CTRL+X





# Python – Running your code

- Now we are ready to run our program. The steps to run it are:
- In the IDLE shell (or whichever IDE your using) select the "Run" command



# Python – Running your code

- Check the output of your program and see if it's doing what you expected
- Fix any errors and/or bugs that come up
- Repeat until it does what it's supposed to
- If you want to run your code in terminal you simply type:

```
$>python3 name_of_file.py
```

**Note:** you should be in the file directory

## What is Git?



- Git is a distributed version control system used to track changes in source code and manage software development projects.
- It allows multiple developers to collaborate on a project by maintaining a history of changes, enabling them to work on different parts of the code simultaneously without interfering with each other's progress.
- Git keeps track of changes in files, supports branching and merging of code, and allows users to roll back to previous versions if needed.

## What is GitHub?

- GitHub is a web-based platform that uses Git for version control
- GitHub is a hub for git repositories
- Allows for relatively easy use of git over the internet
- For this class we will be using GitHub classroom



# Exercise 0

- Since we'll be using GitHub to track your progress and submit your work Our first goals are to:

1. Create a GitHub Account
2. Accept the [Lab 01 Assignment](#), which gives us an online repository

W 11:00 - 12:50 PM (1005B) [Link](#)

3. Clone the Lab 01 repository to your local machine so we can work on our lab machine/laptop
- **Optionally**, if you want to install the GitHub Desktop client for your home machine (Windows/Mac Only) you can save your work and upload to GitHub even without a lab machine!

## Exercise 0 - Clone

- Now, you should see a new page with a LOT of different links and buttons. Don't worry. You should see a big [green button labeled "Code"](#) which you can do one of two things with
  1. Copy the URL which you can use to "clone" the repository to your lab machine via terminal.
  2. Click and select the option to open with "GitHub Desktop Client" if you're working on your laptop. This creates a clone of the repository on your local machine and you can begin creating files in it.

## Exercise 0 - Clone

- Copy the URL (same URL from previous steps)
- Create a folder on your local machine that you want to store this weeks lab in
- Right-click on that folder and choose "Open in Terminal"
- Type this command in terminal:

```
git clone https://github.com/the_URL_you_just_copies_here.git
```

# Exercise 1

- Create a python script to print the line “This is my first lab exercise!” to the terminal
- You should create a new folder and a new file called lab01-exercise01.py
- Then upload it to git using the commands
  - `git add <file name>`
  - `git commit -m "< meaningful message>"`
  - `git push`



# Exercise 1

- Save your file(s)
- Add the changed files to git's "staging area" by typing:

```
git add file_name.py
```

- Commit the changes to your local repository by typing:

```
git commit -m "Message explaining what you're saving"
```

# Exercise 1

- We have just added a remote to our repository. Now we will push our changes repository to GitHub.

```
git push <remote> <branch>
```

Example:

```
git push origin master
```

## Exercise 2

- Create a program to display your name, major, and hobbies. Print each statement on its own line. Your output should be similar to the output below. Remember to use `"\t"` in your output String to cause it to indent.

```
My name is John Gibbons.  
I am an EECS major.  
My hobbies are:  
    Coding  
    Board games  
    Walking my dog  
    Cooking  
Goodbye
```

# Homework format

- All programs must have a comment header
- All programs must run
- Attendance is 10% of your grade

```
'''
Author: Your Name Here
KUID: 1234567
Date: Date file was created
Lab: lab##
Last modified: Date file was most recently modified
Purpose: Description of what this file does
'''
```

# Rubric

- [10pts] Attendance
- [20pts] Exercise 0 / Correct submission to GitHub classroom repo
- [30pts] Exercise 1
  - Displays as directed
- [30pts] Exercise 2
  - [15pts] Name, major, and hobbies displayed on separate lines
  - [15pts] Use of tab character
- [10pts] Documentation
  - Name, date, file description at the top of every .py file