

# **CPSC 1101**

# **Introduction to Computing**

School of Engineering & Computing  
Dept. of Computer Science & Engineering  
Dr. Sidike Paheding

# Class Itinerary

- Introductions
- Review Syllabus
- Cover [fairfield.blackboard.com](https://fairfield.blackboard.com)
- Setup Python compiler
- Setup text editor
- Chapter 1 reading assignment

# Course Syllabus



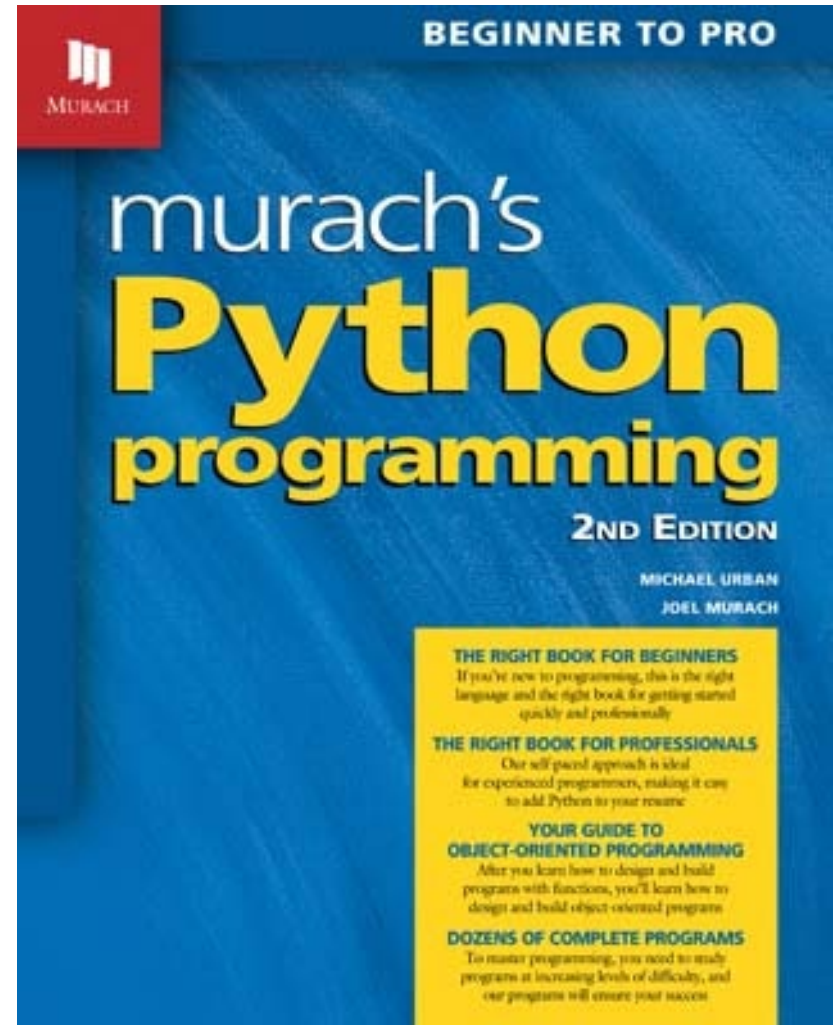
Fairfield University

# Course Info

- **TIME & ROOM:**  
CPSC 1101 – 02: 11:00 am – 12:15 pm, T & F, BNW 166  
CPSC 1101 – 03: 2:00 pm – 3:15 pm, M & R, BNW 131
- **OFFICE HOURS:**  
3:30 am – 4:30 pm Monday and Tuesday or by appointment
- **COURSE CONTENT:**  
Fairfield University Blackboard - <https://fairfield.blackboard.com/>
- **INSTRUCTOR:**  
**Dr. Sidike Paheding**  
spaheding@fairfield.edu  
Office: BNW 236B
- Teaching Assistant (TA):  
James Haimoff [james.haimoff@student.fairfield.edu](mailto:james.haimoff@student.fairfield.edu)

# Textbook

- “Murach’s Python Programming (2<sup>nd</sup> Edition)”, Michael Urban and Joel Murach, Mike Murach & Associates, 2021, ISBN 978-1943872749 **(required)**
- “Think Python: How to Think Like a Computer Scientist (2<sup>nd</sup> Edition)”, Allen Downey, O’Reilly Media, 2016, ISBN-13: 978-1491939369 **(optional)**
- Online Resource:  
<https://www.learnpython.org/>



# COURSE OUTCOMES

- Demonstrate the ability to solve complex computational problem through the process of planning, design, implementation and testing a software application. (1, 2, 6)<sup>1</sup>
- Demonstrate an understanding of the breadth of Computer Science and its fundamental disciplines as applied through web architecture & web security (3, 4, 5)<sup>1</sup>
- Be fluent in the use of programming fundamentals: control flow, statement blocks, variable assignments, conditional statements, loops, function calls, & modules in Python. (1, 2, 6)<sup>1</sup>
- Demonstrate a basic understanding of the concepts of object-oriented programming as used in Python through both existing and user defined classes. (1, 2)<sup>1</sup>
- Demonstrate the proper application of the best practices in computer science: introduction to development programming through IDE & web services. (3, 5)<sup>1</sup>

<sup>1</sup><http://www.abet.org/wp-content/uploads/2018/02/C001-18-19-CAC-Criteria-Version-2.0-updated-02-12-18.pdf>

# COURSE STRUCTURE

- **Lecture** – Classes will be conducted in traditional lecture format.
- **In-class programming assignments** – The professor will test if your code is producing correct results.
- **Homework** – Programming assignments that will build on the week's lecture content.
- **Readings** – Course textbook pages, relevant articles and additional supporting content will be assigned for students to read.
- **Discussions** – opportunities to share questions about key concepts, homework assignments, and more.

# Exams and Quizzes

- There will be a midterm and final exam in this course.
- The exams will cover lectures, homework assignment, and readings. Please note in the grading section that these are weighted more heavily than homework assignments.
- Quizzes will be also given during the semesters.



# FINAL PROJECT

- You will synthesize your computer science knowledge and Python programming skills to create a larger-scale project demonstrating what you've learned.

The project grading will be based on:

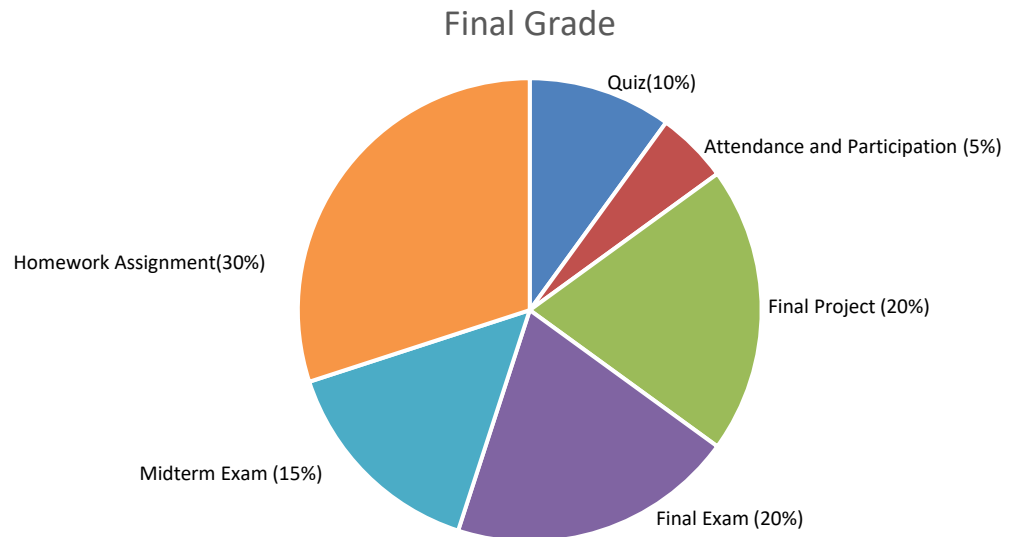
- ✓ Functionality
- ✓ Application of CS fundamentals
- ✓ Relevance
- ✓ Complexity

# GRADE SCALE

Grade Name	Grade Range %
A	93% and 100%
A-	90% and less than 93%
B+	87% and less than 90%
B	83% and less than 87%
B-	80% and less than 83%
C+	77% and less than 80%
C	73% and less than 77%
C-	70% and less than 73%
D	60% and less than 70%
F	0% and less than 60%

# GRADING

- Homework Assignment (30%)
- Final Project (20%)
- Quiz (10%)
- Midterm Exam (15%)
- Final Exam (20%)
- Attendance and Participation (5%)
- *There are reading assignments for each chapter (see syllabus).*



# Class POLICY

- ATTENDANCE POLICY
  - As a student at Fairfield University, you are expected to attend every scheduled class session.
  - Please inform the instructor in advance, preferably by e-mail, if you must be absent from a class.
  - while in class **DO NOT** use cell phones, Internet browsers, chats, or any computer software and tools for personal matters
- MISSED CLASSES
  - Students are responsible for obtaining material distributed in class in days when he/she was absent.
  - There are no make-up exams unless the student missed exam due to a pre-arranged, documented excused absence. **Any uncoordinated, unexcused missed exam will result in a score of 0 for that exam.**
- LATE ASSIGNMENTS
  - Late submission will be assessed a penalty of 20% with no more than two days allowed.

# Others

- See syllabus about
  - [ACADEMIC INTEGRITY](#)
    - Unless stated otherwise by your instructor, **individual work** is expected. Anything you turn in with your name on it must be your own work, that is, written or coded by you and not copied from anyone or anywhere else.
  - [ACADEMIC DISHONESTY](#)
  - [ACCESSIBILITY](#)
  - [COURSE WITHDRAWAL](#)
  - [COMMITMENT TO DIVERSITY AND INCLUSION](#)
- [AI USAGE STATEMENT](#)
  - **Prohibit AI usage:** The use of generative artificial intelligence tools for assignments in this course, including tools like ChatGPT, Perplexity, Gemini and other AI writing or coding assistants, is prohibited. The use of generative artificial intelligence tools for the completion of, or to support the completion of, an examination, quiz, assignment, lab report, or any other form of academic assessment, will be considered an academic offense in this course.

# Blackboard

Course Management System



Fairfield University

# **How to command computers to perform a task?**

# What Is Programming?

- Many people use computers for everyday tasks such as electronic banking or writing a term paper.
- Computers can carry out a wide range of tasks because they execute different *programs*, each of which directs the computer to work on a specific task.
- The computer itself is a machine that stores data (numbers, words, pictures), interacts with devices (the monitor, the sound system, the printer), and executes programs.



# What Is Programming?

A *computer program* tells a computer, in minute detail, the sequence of steps that are needed to fulfill a task.

**Hardware:** The physical computer and peripheral devices are collectively called the *hardware*.

**Software:** The programs the computer executes are called the *software*.

**Software** (i.e., the instructions you write to command computers to perform **actions** and make **decisions**) controls computers (often referred to as **hardware**).

# What Is Programming?

## Programming

is the act of designing and  
implementing computer programs.

# Programming Languages

Programmers write instructions in various programming languages, some directly **understandable** by computers and others requiring intermediate *translation* steps.

3 general  
types of  
programming  
languages:

- Machine Languages
- Assembly Languages
- High-level Languages

# Machine Languages

- Any computer can directly understand *only* its own **machine language**, defined by its hardware design.
- Machine languages generally consist of numbers (ultimately reduced to 1s and 0s). Such languages are cumbersome for humans.
- Example: here is a section of an early machine language payroll program that adds overtime pay to base pay and store the result in gross pay:

```
+1300042774  
+1400593419  
+1200274027
```

# Assembly Languages

- Programming in machine language—the numbers that computers could directly understand—was simply too slow and tedious for most programmers.
- Instead, they began using English-like abbreviations to represent elementary operations.
- These abbreviations formed the basis of [assembly languages](#).
- *Translator programs* called [assemblers](#) were developed to convert assembly-language programs to machine language.
- Example: A section of payroll program written in assembly language shows adding overtime pay to base pay and store the result in gross pay:

load	basepay
add	overpay
store	grosspay

# High-Level Languages

- Although assembly-language code is clearer to humans, it's incomprehensible to computers until translated to machine language.
- To speed the programming process even further, **high-level languages** were developed in which *single statements could be written to accomplish substantial tasks*.
- High-level languages allow you to write instructions that look almost like everyday English and contain commonly used mathematical expressions.
- Translator programs called **compilers** convert high-level language programs into machine language.

Example: a payroll program written in a high-level language might contain a single statement such as:

```
grosspay = basepay + overpay
```

# High/Low -Level Languages

- *High-level languages*
  - independent of the processor type and hardware. They will work equally well:
    - on an Intel Pentium and a processor
    - in a cell phone
  - close to our human language
  - examples include: C++, Java, Pascal, Python
- *Low-level language*
  - the machine code for a specific architecture and hardware of a particular type of computer
  - closer to the native language of a computer (binary)
  - example: assembly language, machine Code

# Three types of language Categories

- **Scripting Language:** Write the script to perform certain tasks.
  - subcategory of programming languages that set instructions to another program
  - need to be interpreted, which scans the code line by line, instead of compiled.
  - most widely used to create a website.
  - e.g., Javascript, PHP, Perl, Python, VBScript
- **Other Programming languages:** A set of instructions that tells a computer to perform a task.
  - Need to be converted into machine code because a computer can only understand such as binary language (0 and 1).
  - We write the instructions in human-readable form and then convert this into machine-level language. This conversion is done by the compiler which scans the complete code in one go.
  - e.g., Java, C, C++, C#
- **Markup Languages:** prepare a structure for the data so it does not include any kind of algorithm. Like HTML which tells the browser how to structure data for a specific page, layout, etc., so that the web browsers can read HTML documents and display them correctly.
  - Example languages: HTML, CSS(Cascading Style Sheets)



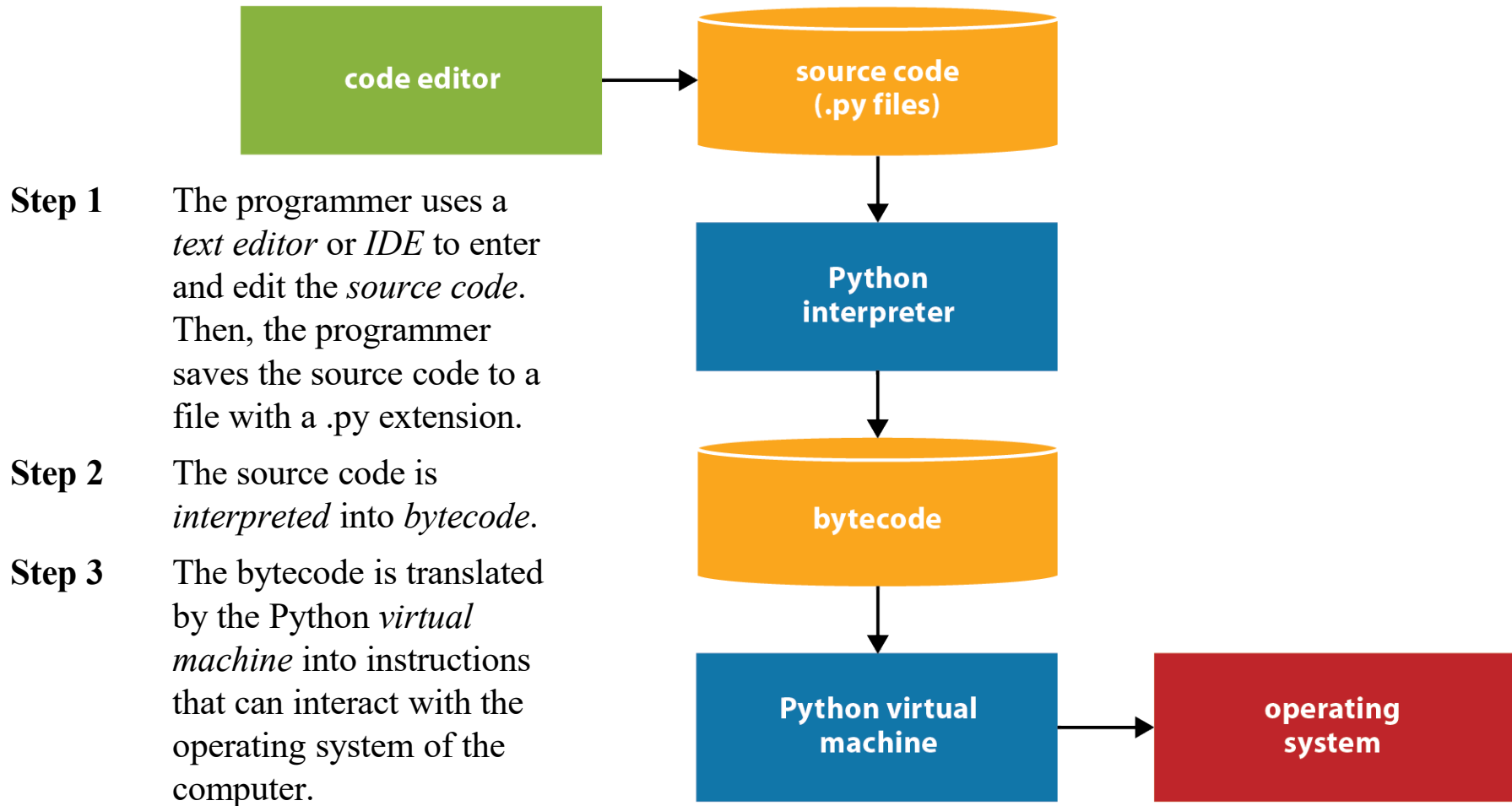
# Scripting Languages v.s. other Programming Languages

- Programming languages are **compiled**, while scripting languages are mostly **interpreted**.
  - A compiled language is converted into machine code so that the processor can execute
  - An interpreted language, the implementations execute instructions directly without compiling
  - The compiled programs run faster than interpreted programs.

Scripting languages	Programming languages
(Mostly) interpreted	Compiled
Slower at runtime	Faster at runtime
Less code-intensive	More code-intensive
Creates apps as part of a stack	Creates standalone apps

[Table credit](#)

# What is Python





- Scripting language
- designed by Guido van Rossum. Released in 1991
- an open-source and high-level programming language
- Used for
  - **Web development:** its efficiency, speed, and rich open-source libraries make Python excellent for web development. Libraries like *Django* and *Flask* were written in Python.
  - **Software development:** Create light-weighted software programs and applications running on your devices. Many Python packages, such as *NumPy*, *Tkinter*, *SciPy*, etc.
  - **Machine learning and artificial intelligence:** The extensive Python library makes many AI-based apps easy to develop and manage. Libraries such as *TensorFlow*, *Scikit-learn*, etc.
  - **Data Science:** Extensive libraries in Data Science. e.g., *Pandas*, *Numpy*, *PyBrain*, *Seaborn*, *Matplotlib*, *Keras*, and *PyTorch* are used in data science to visualize and analyze large volumes of data.
  - **GUIs:** Build desktop GUIs utilizing an efficient framework, such as *PyGUI*, *PyGtk*, *PyQt4*, *PyQt5*, etc.
  - **Game development:** Python libraries like *PySoy*. Games developed using Python are Disney's Toontown Online, Battlefield 2, etc.

[source](#)

# Programming terms

- Variables
- Constant
- Input / Output i.e. i/o
- Conditional statement
- Loops / repetition
- Functions / methods

# Debugging

- Debugging is the process of finding and fixing bugs in a program.
- What are bugs, Why do we debug
- Syntax vs. runtime vs. logical errors
- Exceptions
  - Exceptions are a specific type of runtime error that can be handled by the program.

# Comments

- Comments start with #, and they are ignored by the compiler.
- Inline comments are coded after statements to describe what they do.
- Block comment: """

...

...

"""

"""

Guidelines for using comments

- Use comments to describe portions of code that are hard to understand, but don't overdo them.
- Use comments to comment out (or disable) statements that you don't want to test.
- If you change the code that's described by comments, change the comments too.

"""

# Comments

```
# This is a tutorial program that illustrates the use of the while
# and if statements

# initialize variables
counter = 0
score_total = 0
test_score = 0

# get scores
while test_score != 999:
    test_score = int(input("Enter test score: "))
    if test_score >= 0 and test_score <= 100:
        score_total += test_score
        counter += 1

# calculate average score
#average_score = score_total / counter
#average_score = round(average_score)
average_score = round(
    score_total / counter)

# display the result
print("=====")
print("Total Score: " + str(score_total)
      + "\nAverage Score: " + str(average_score))
```

Block comments

Inline comments

Commented out statements

# add score to total

# add 1 to counter

# implicit continuation

# same results as commented out statements

# implicit continuation

# Why Python

- Python has a simple syntax.
- Python supports the development of a wide range of programs, including games, web applications, and system administration.
- Python is used by many successful companies, including Google, IBM, Disney, and EA Games.
- Python is open source.



# Why Python in this course?



[source](#)

# Why Python in this course?

- Top 10 programming languages
- In-demand skill in market

## Most in-demand programming languages of 2022

*Based on LinkedIn job postings in the USA & Europe*

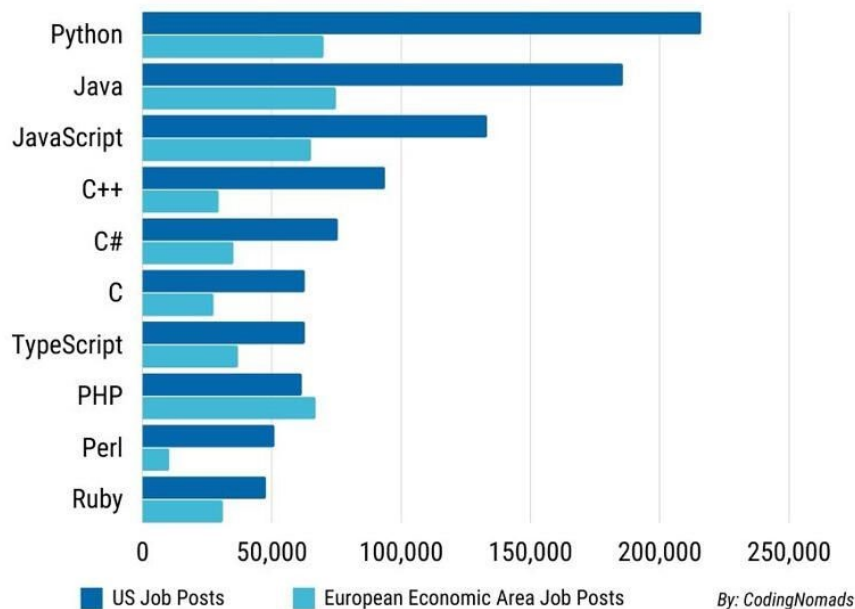


Image: CodingNomads

[source](#)

# Programming Platforms

VS Code, Spyder, Jupyter Notebook



Fairfield University

# Anaconda



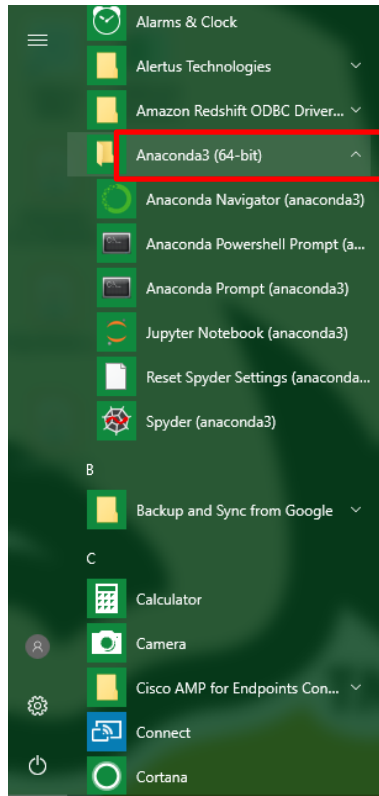
- A modern open source analytics platform powered by Python
- Leading open data science platform
  - makes the open source tools of data science
- Easy to collaborate across the entire data science team



<https://www.anaconda.com/>

# Anaconda

- Launch an Anaconda Navigator.






## Windows

From the Start menu, click the Anaconda Navigator desktop app.


# Anaconda Navigator


 ANACONDA NAVIGATOR


[Sign in to Anaconda Cloud](#)


[Home](#)  
[Environments](#)  
[Projects \(beta\)](#)  
[Learning](#)  
[Community](#)  
  
[Documentation](#)  
[Developer Blog](#)  
[Feedback](#)  
  
  


Applications on root Channels Refresh


  
jupyter  
notebook  
5.0.0  
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.  
[Launch](#)

  
qtconsole  
4.3.0  
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.  
[Launch](#)

  
spyder  
3.1.4  
Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features  
[Launch](#)

  
glueviz  
0.10.4  
Multidimensional data visualization across files. Explore relationships within and among related datasets.  
[Install](#)

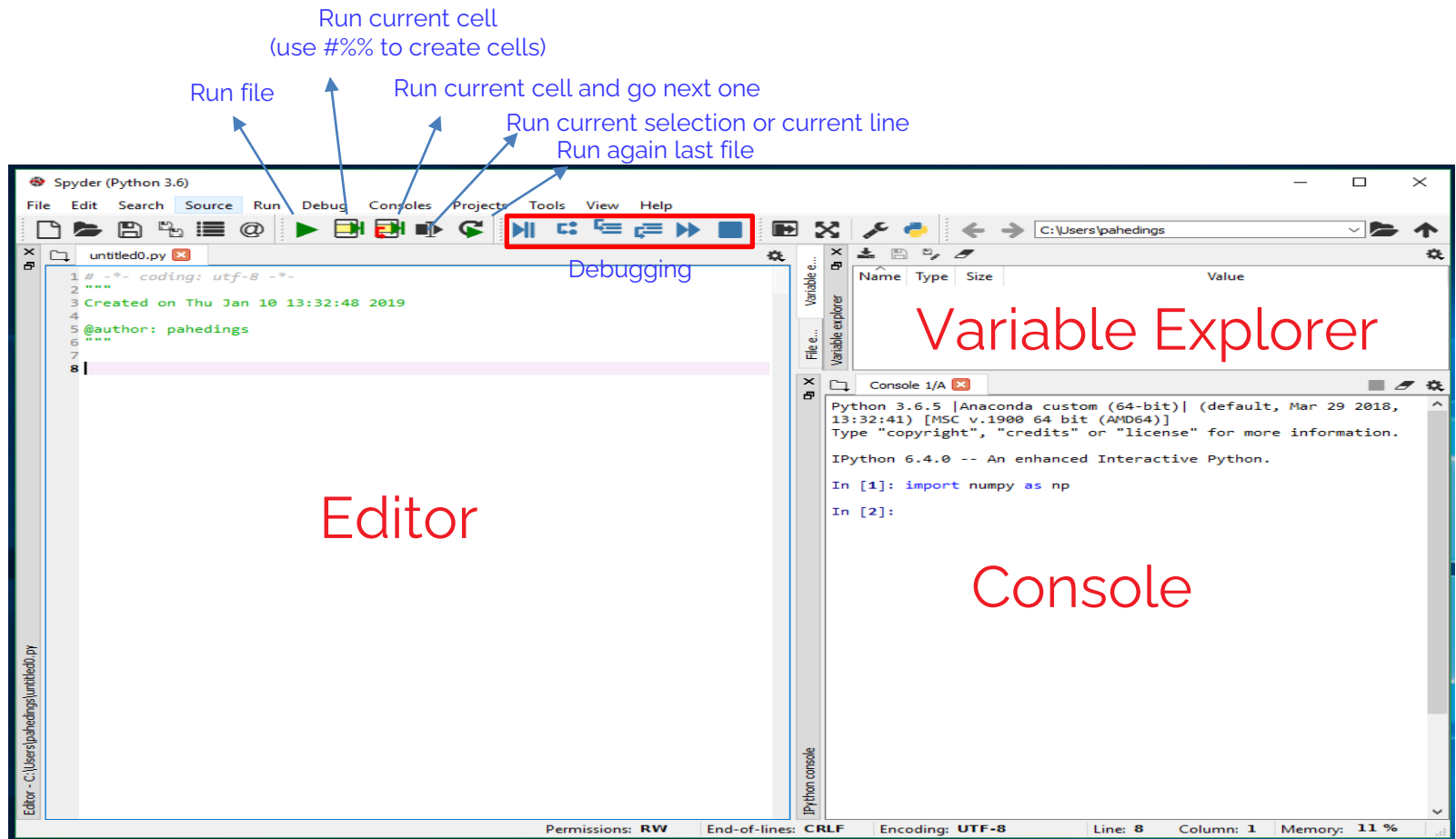
  
orange3  
3.4.1  
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.  
[Install](#)

  
rstudio  
1.0.136  
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.  
[Install](#)

**Work with Python/R packages and environments without needing to type `conda` commands in a terminal window.**

# Spyder

- The scientific Python IDE (**I**ntegrated **D**evelopment **E**nvironment)
- Offers editing, analysis, debugging, and profiling functionality



# Jupyter Notebook

- Open source web application
- Live code, equations, visualizations and narrative text.
- Support Python, R, Julia, and Scala.

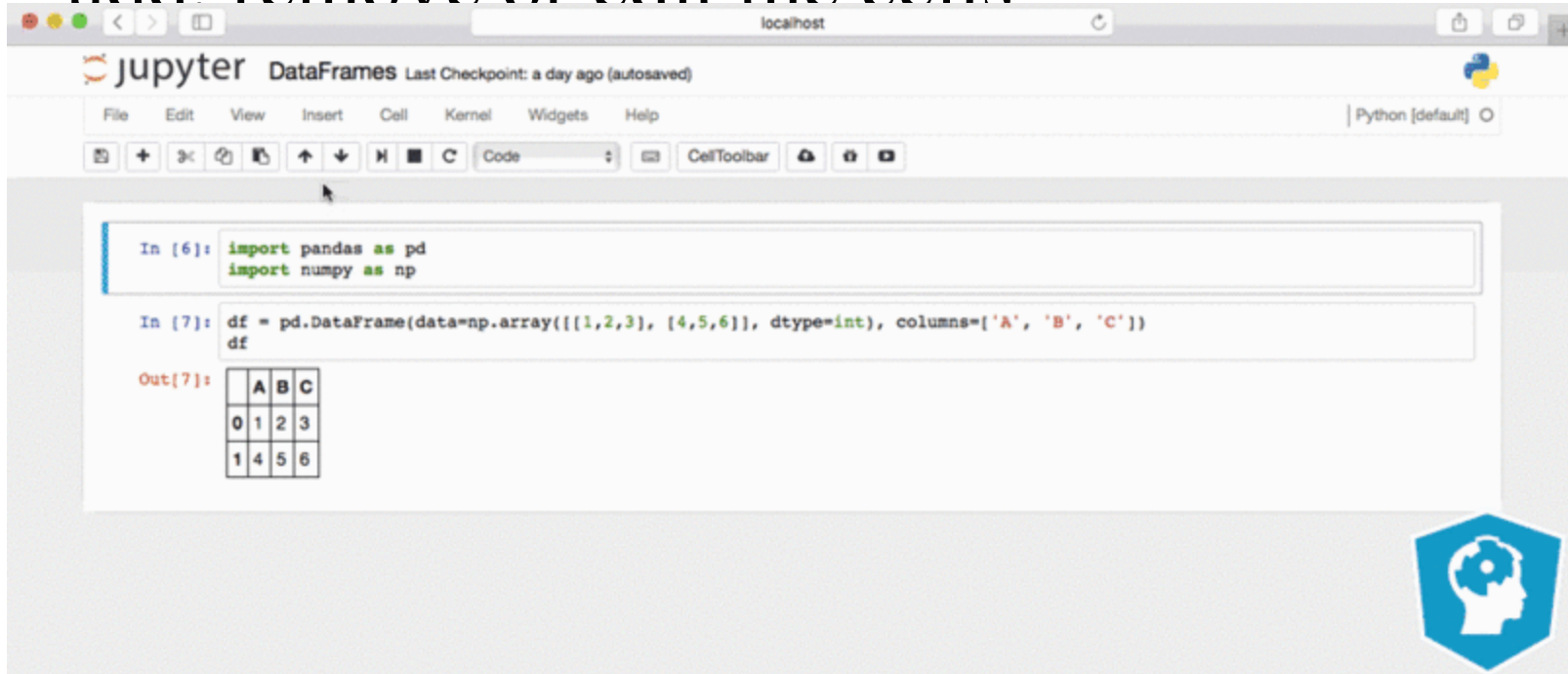


<https://jupyter.org/>



# Jupyter Notebook – edit cell

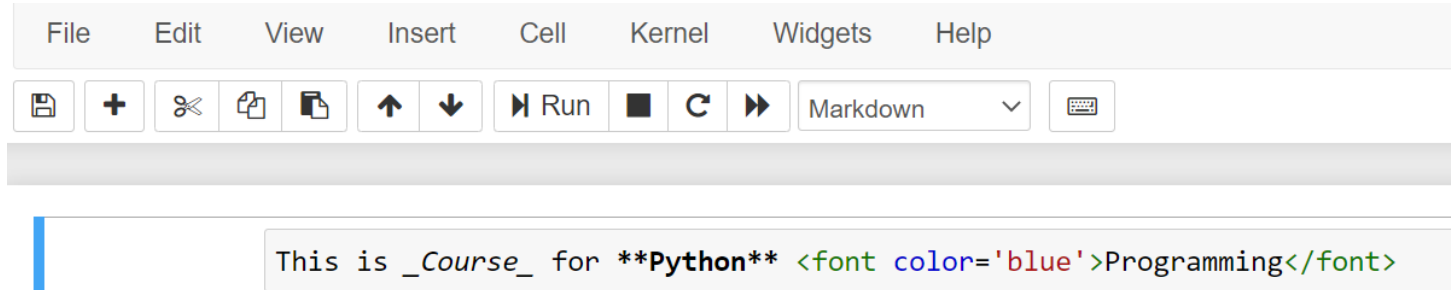
- add, remove or edit the cells



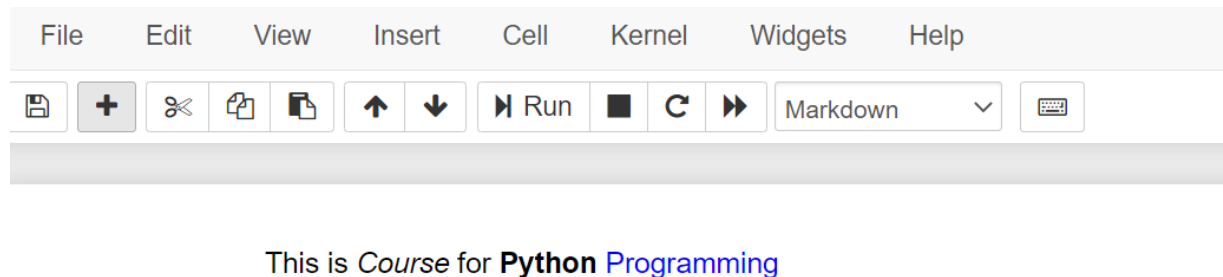
<https://www.datacamp.com/tutorial/tutorial-jupyter-notebook>

# Jupyter Notebook - Markdown

- Jupyter Notebook supports Markdown, which is a markup language that is a superset of HTML.

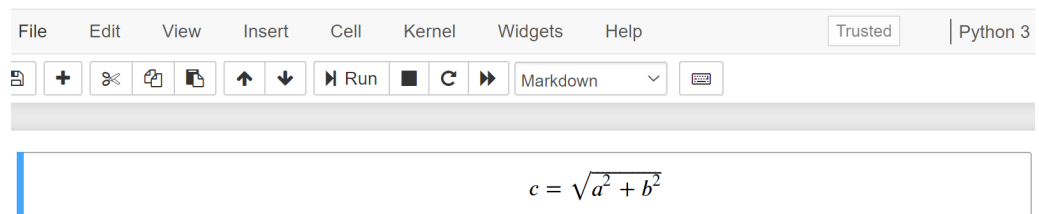
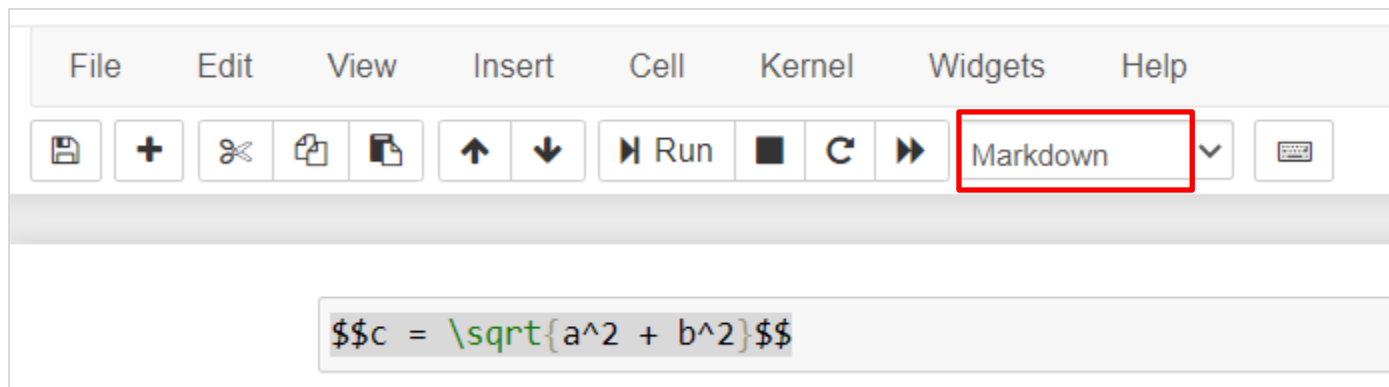


- When you run the cell, the output should look like this:



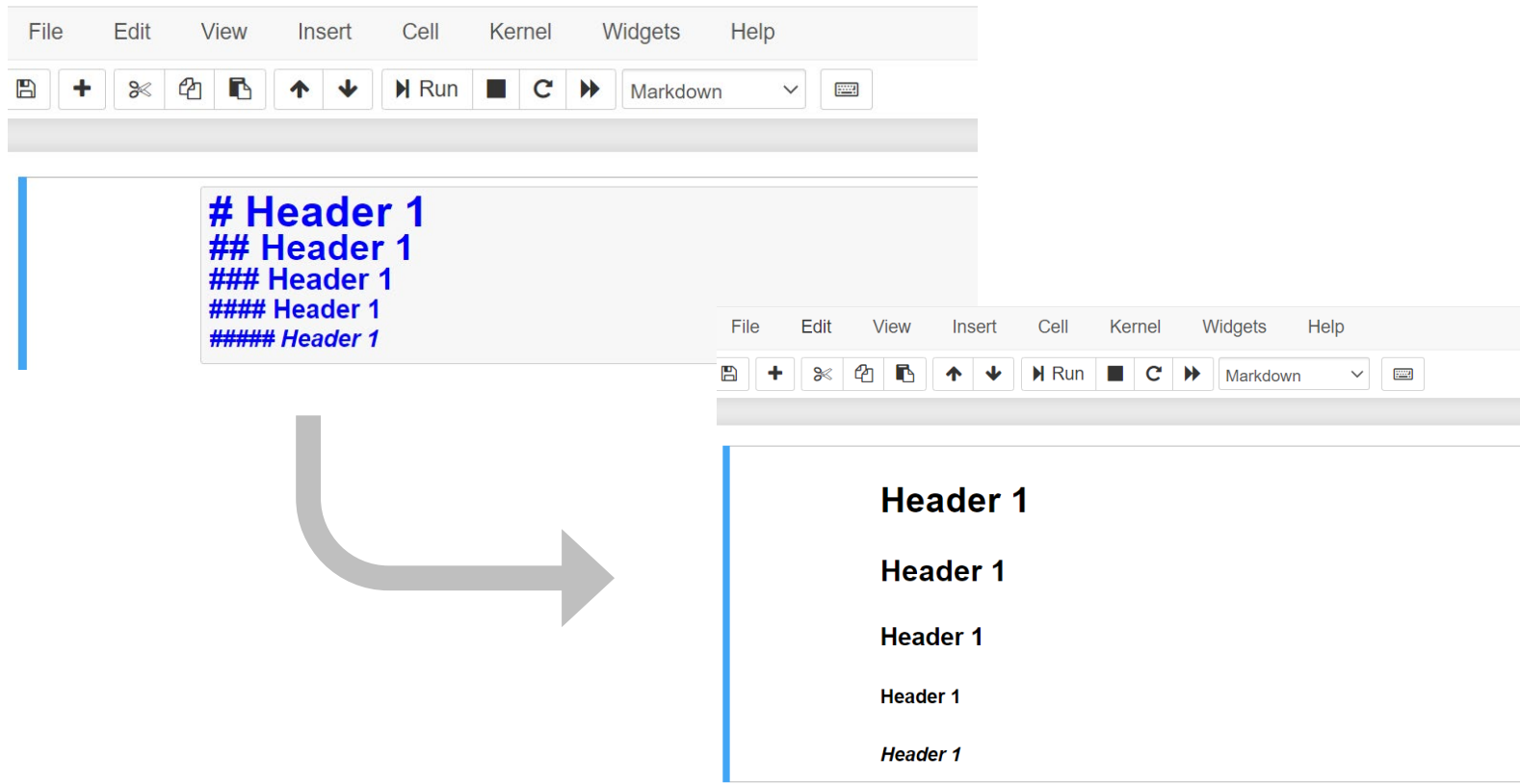
# Jupyter Notebook - LaTeX

- if you want to insert LaTeX in your code cells, you just have to put your LaTeX math inside  $\$xxx\$,$  for instance:



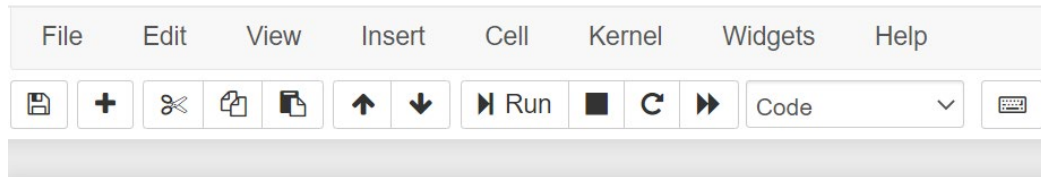
# Jupyter Notebook - Headers

- Use the humble pound sign. The more pound signs you use, the smaller the header.

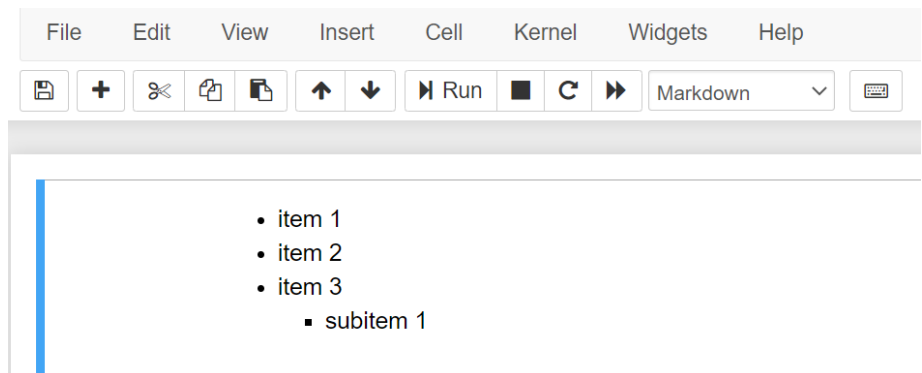
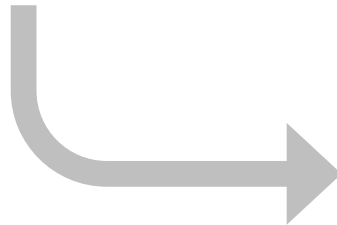


# Jupyter Notebook – Creating list

- You can create a list (bullet points) by using dashes, plus signs, or asterisks. Here is an example:

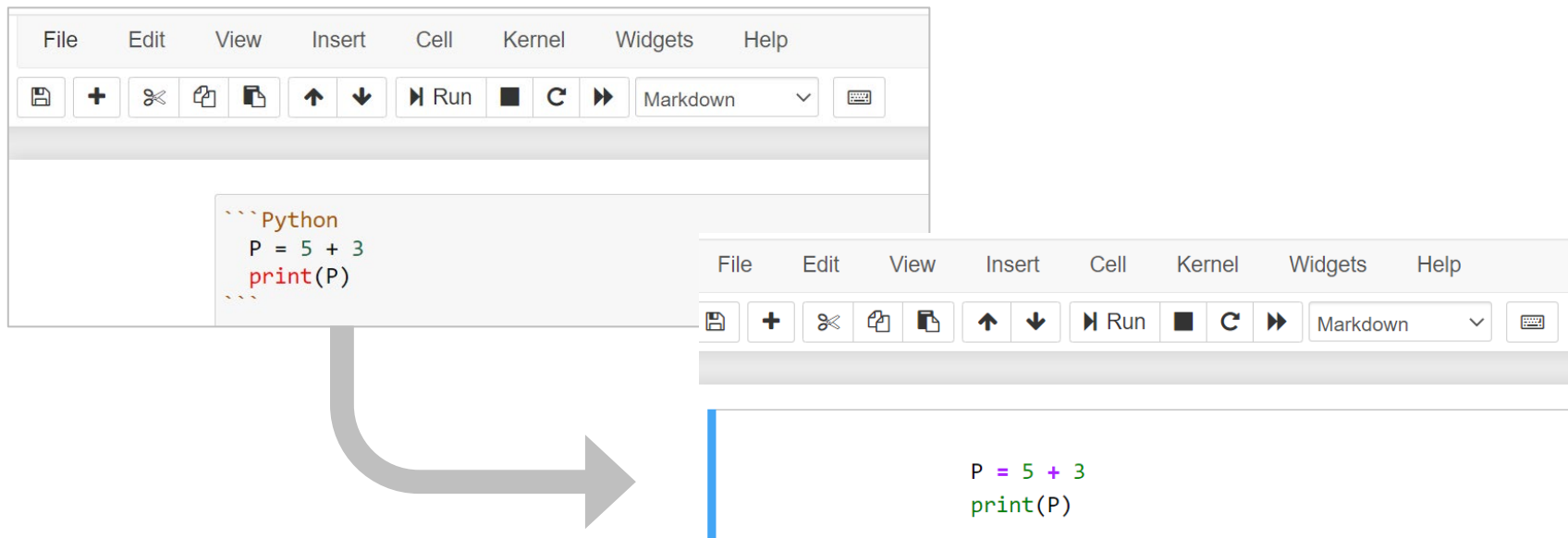


```
* item 1  
- item 2  
+ item 3  
  * subitem 1
```



# Jupyter Notebook – inline code highlighting

- For inline code highlighting, you can surround the code with backticks (`). If you want to insert a block of code, you can use triple backticks (```) and also specify the programming language:



# Google Colab

- Great platform for ML and data science
- Write and execute Python in your browser
- Code, text, results interactions.
- Zero configuration required
- Free access to GPUs
- Easy sharing
- Doc and examples:

<https://colab.research.google.com/notebooks/intro.ipynb>

- Tutorial video: [Get started with Google Colaboratory](#)

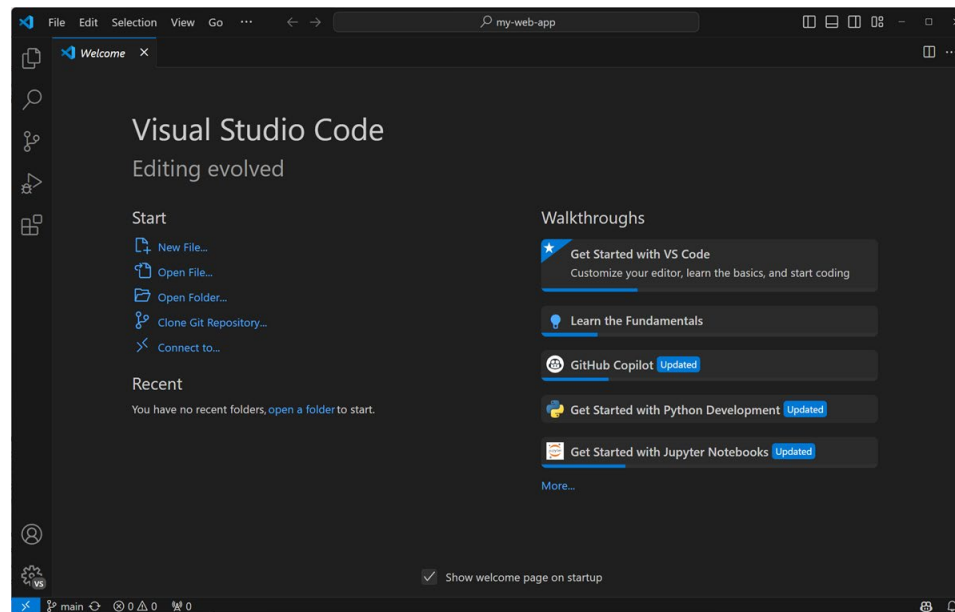


Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

# Visual Studio Code

- A lightweight source code editor which runs on your desktop and is available for Windows, macOS and Linux.



- [Visual Studio Code Tips and Tricks](#)
- [Introductory Videos](#)



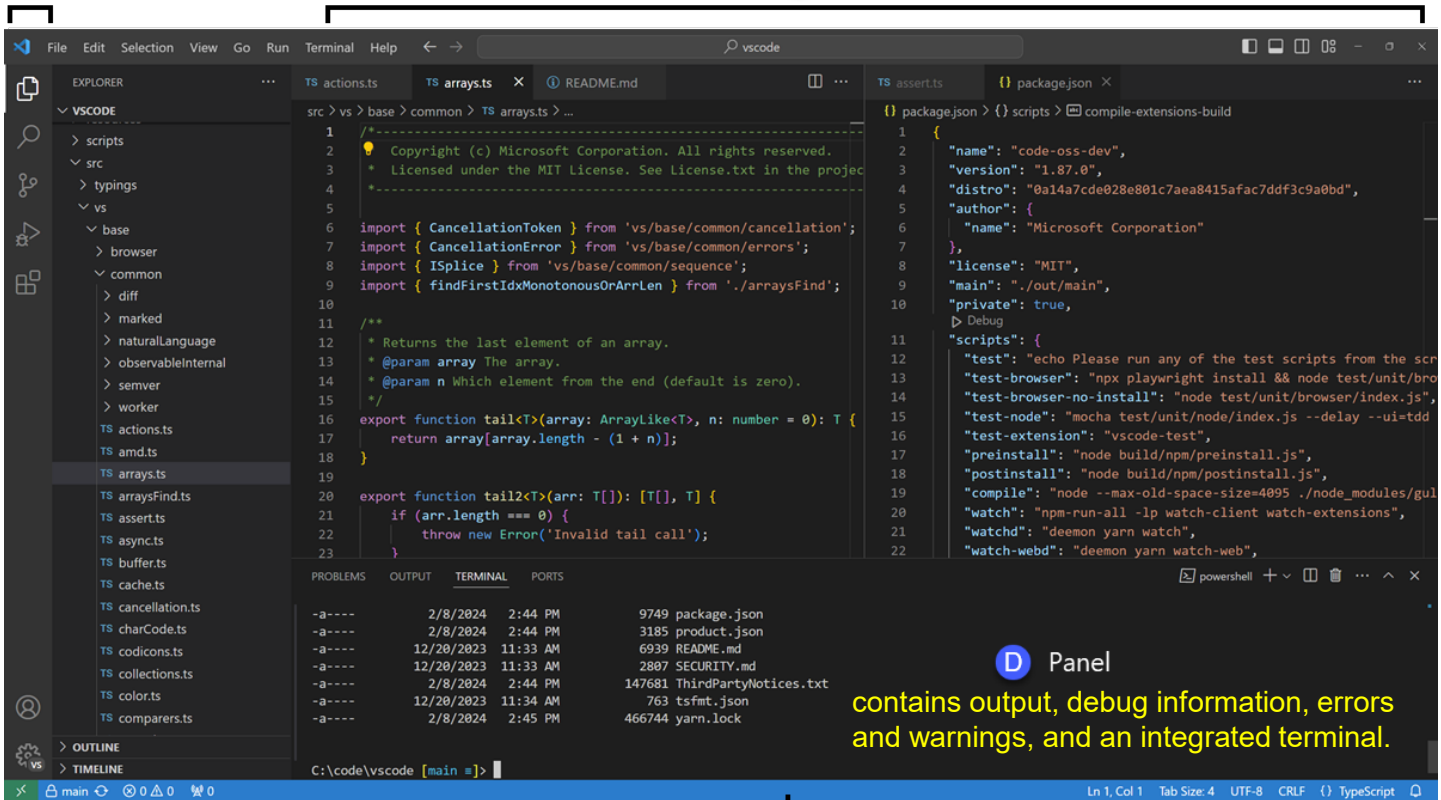
# User Interface

Lets you switch between views and gives you additional context-specific indicators

The main area to edit your files. You can open as many editors as you like side by side vertically and horizontally.

**A** Activity Bar

**C** Editor Groups



**B** Primary Side Bar

**E** Status Bar

Contains different views like the Explorer to assist you while working on your project.

Information about the opened project and the files you edit.

# VS Code

- Getting Started with Python in VS Code :  
<https://code.visualstudio.com/docs/python/python-tutorial>
- Python in Visual Studio Code:  
<https://code.visualstudio.com/docs/languages/python>
- Visual Studio Code on Windows (Python text editor)  
<https://code.visualstudio.com>
- Python extension for Visual Studio code  
<https://marketplace.visualstudio.com/items?itemName=ms-python.python>

# VS Code

- Visual Studio Code on macOS:  
[https://code.visualstudio.com/docs/setup/mac#\\_launching-from-the-command-line](https://code.visualstudio.com/docs/setup/mac#_launching-from-the-command-line)
- NOTE: if you are using MacOS version, you might run into a PATH-related issue (i.e. "brew" not in the path). This link shows a couple of options to fix this issue:
- <https://stackoverflow.com/questions/65487249/getting-a-warning-when-installing-homebrew-on-macos-big-sur-m1-chip>

# **Thank you for participating in CPSC 1101 - Intro to Computing.**

Are there any questions?

[spaheding@fairfield.edu](mailto:spaheding@fairfield.edu)