
Generative Artificial Intelligence for Code Generation (1)

COMP4431 Lab 7

Lab 07 Agenda

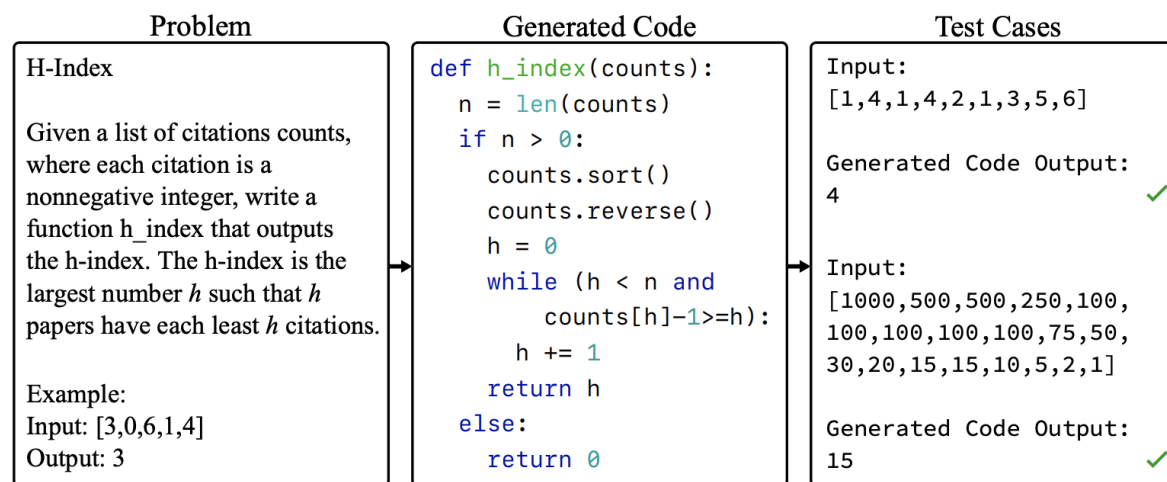
- Introduction to Generative Artificial Intelligence for Code Generation
 - Concepts
 - Leaderboard
 - Access
- Basic Techniques
 - Tab and Chat
 - Models
 - Context Management
 - Prompt Engineering
- Case Study

Concepts

■ Code Generation

- Code Generation is an important field focused on predicting explicit code or program structures from multimodal data sources, including:
 - Incomplete code
 - Programs in other programming languages
 - Natural language descriptions
 - Execution examples
- Code Generation tools can assist the development of automatic programming tools to improve programming productivity.

Ganesh, K. Srivatsa, et al. "A Survey of using Large Language Models for Generating Infrastructure as Code." Proceedings of the 20th International Conference on Natural Language Processing. 2023.



Leaderboard

- Recent development in Code Generation

- Having been trained on a large amount of text for predicting the next word, LLMs have shown remarkable performance in tasks related to code generation and validation.

- Leaderboard

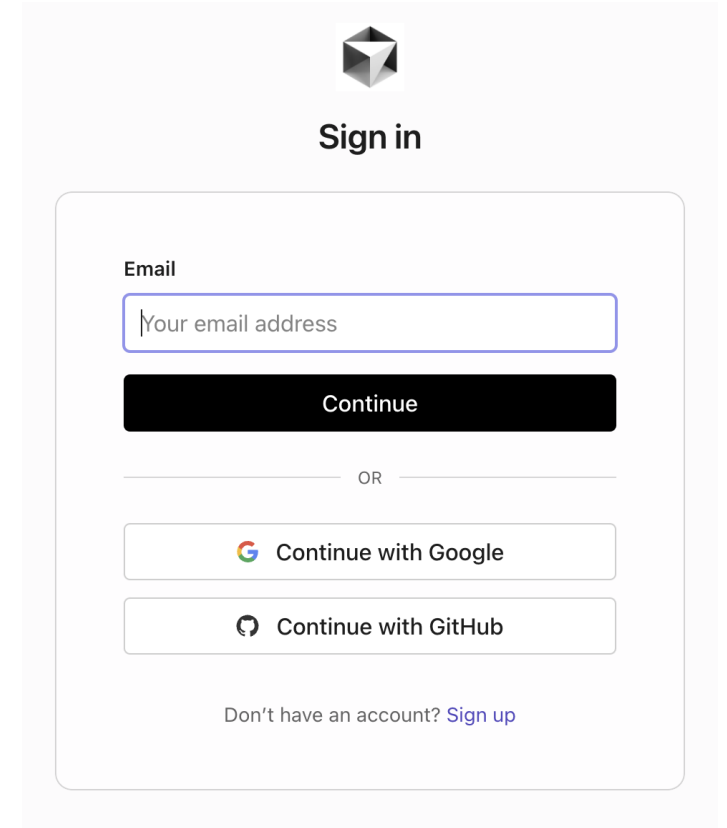
- BigCodeBench, a benchmark that challenges LLMs to invoke multiple function calls as tools from 139 libraries and 7 domains for 1,140 fine-grained tasks.
- Pass@K evaluates the first k generated code's functional accuracy. $K=1$ evaluates if the first generated code can pass all the provided test cases.

#	Model	Pass@1
1	GPT-4o-2024-05-13	56.1
2	DeepSeek-V2-Chat (2024-06-28)	54.1
3	DeepSeek-Coder-V2-Instruct	54
4	GPT-4-Turbo-2024-04-09	53.2
5	Claude-3.5-Sonnet-20240620	52.7
6	GPT-4o-mini-2024-07-18	51.8
7	GPT-4-0613	51.6
8	Claude-3-Opus-20240229	51.5
9	Qwen2.5-72B-Instruct	50.8
10	Gemini-1.5-Pro-API-0514	50.6
10	Hermes-2-Theta-Llama-3-70B	50.6

Access to Generative Artificial Intelligence for Code Generation

■ Step-by-step guide to access to Cursor

- Navigate to the official Cursor AI website <https://www.cursor.com/>
- Create an account
 - Hobby (free)
 - 14 days Pro trial; 50 slow premium model uses; 200 cursor-small uses; 2000 completions uses
 - Pro (\$20.00/month)
 - 500 fast premium model uses per month; Unlimited slow premium model uses; Unlimited cursor-small uses; Unlimited completions completions; 10 Claude Opus uses per month
- Download the appropriate version for your operating system
- Run the installer and follow the on-screen instructions
- Launch Cursor AI and complete any additional setup prompts



Sign in

Email

Your email address

Continue

OR

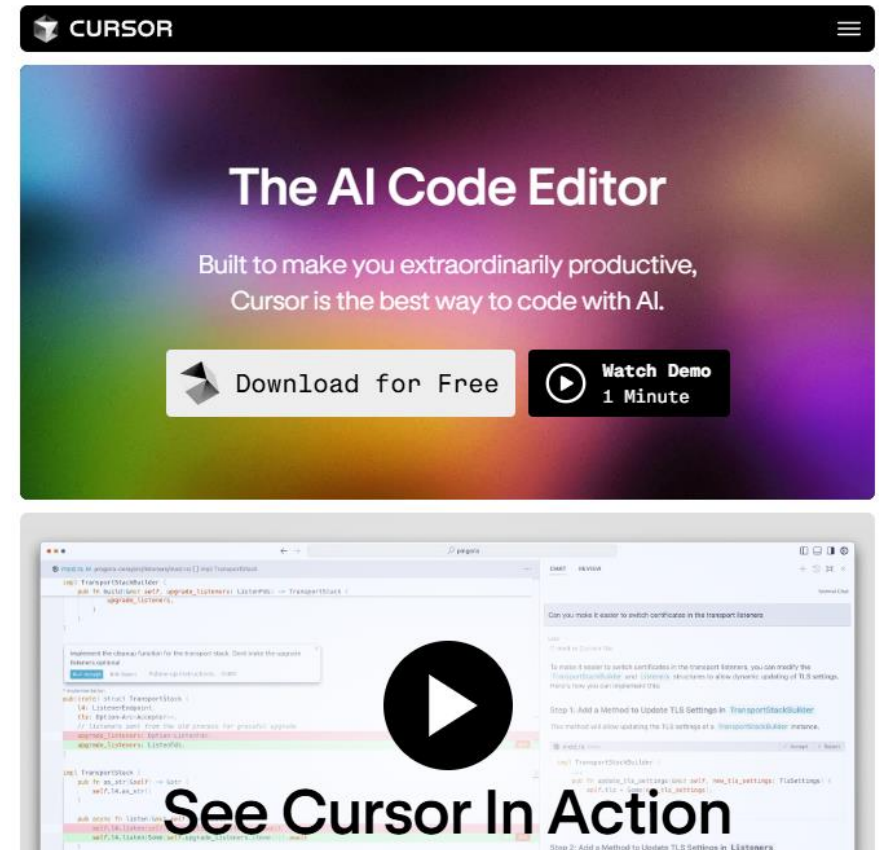
Continue with Google

Continue with GitHub

Don't have an account? [Sign up](#)

Access to Generative Artificial Intelligence for Code Generation

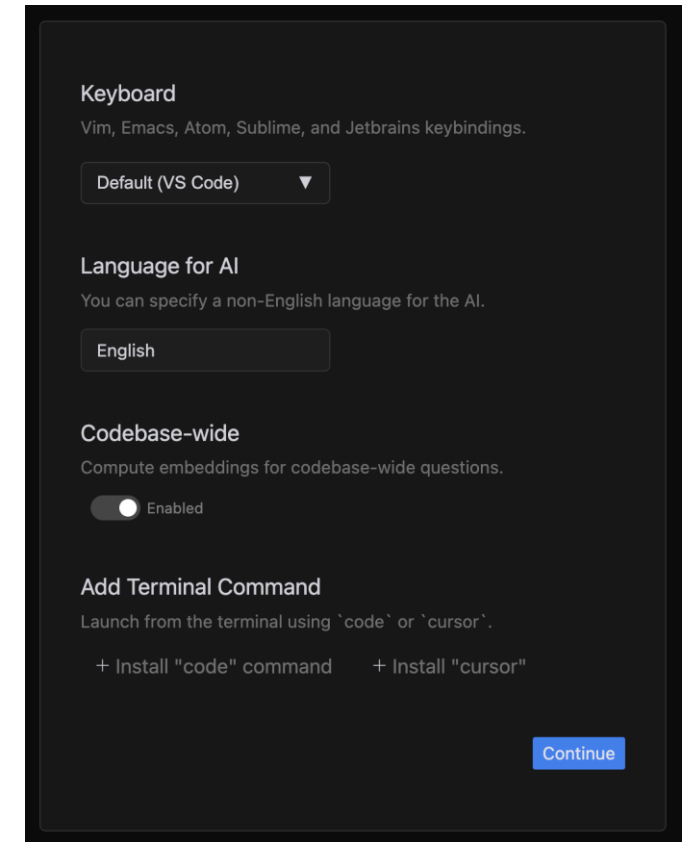
- Step-by-step guide to access to Cursor
 - Navigate to the official Cursor AI website
<https://www.cursor.com/>
 - Create an account
 - Hobby (free)
 - 14 days Pro trial; 50 slow premium model uses; 200 cursor-small uses; 2000 completions uses
 - Pro (\$20.00/month)
 - 500 fast premium model uses per month; Unlimited slow premium model uses; Unlimited cursor-small uses; Unlimited completions completions; 10 Claude Opus uses per month
 - Download the appropriate version for your operating system
 - Run the installer and follow the on-screen instructions
 - Launch Cursor AI and complete any additional setup prompts



Access to Generative Artificial Intelligence for Code Generation

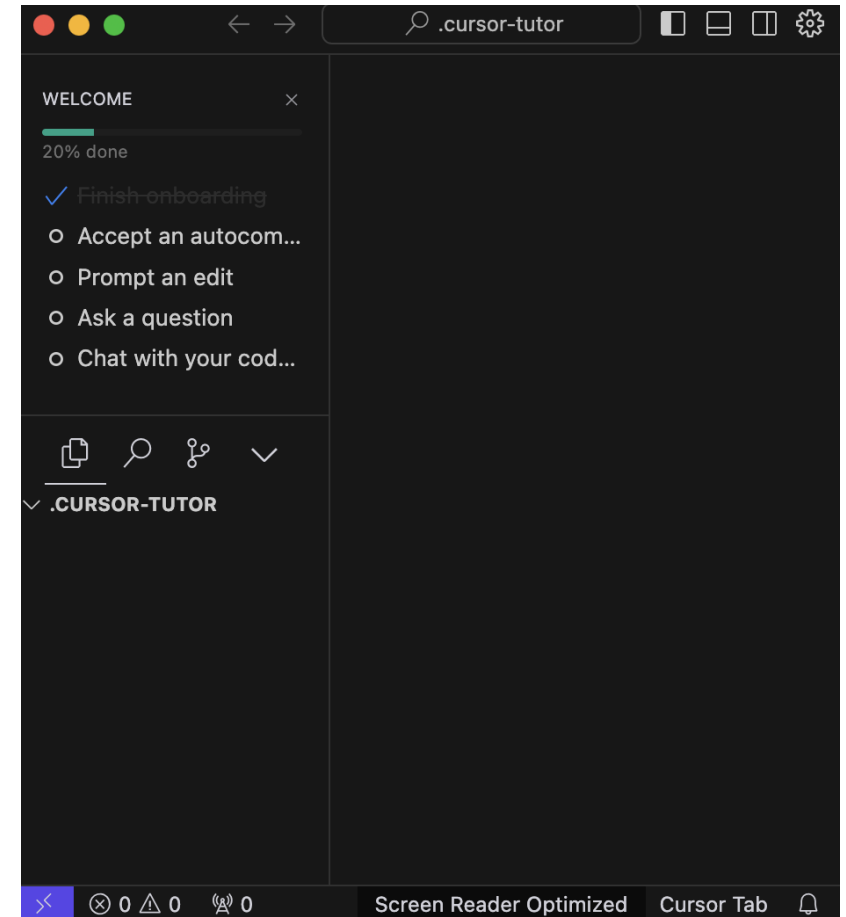
■ Step-by-step guide to access to Cursor

- Navigate to the official Cursor AI website <https://www.cursor.com/>
- Create an account
 - Hobby (free)
 - 14 days Pro trial; 50 slow premium model uses; 200 cursor-small uses; 2000 completions uses
 - Pro (\$20.00/month)
 - 500 fast premium model uses per month; Unlimited slow premium model uses; Unlimited cursor-small uses; Unlimited completions completions; 10 Claude Opus uses per month
- Download the appropriate version for your operating system
- Run the installer and follow the on-screen instructions
- Launch Cursor AI and complete any additional setup prompts



Access to Generative Artificial Intelligence for Code Generation

- Step-by-step guide to access to Cursor
 - Navigate to the official Cursor AI website <https://www.cursor.com/>
 - Create an account
 - Hobby (free)
 - 14 days Pro trial; 50 slow premium model uses; 200 cursor-small uses; 2000 completions uses
 - Pro (\$20.00/month)
 - 500 fast premium model uses per month; Unlimited slow premium model uses; Unlimited cursor-small uses; Unlimited completions completions; 10 Claude Opus uses per month
 - Download the appropriate version for your operating system
 - Run the installer and follow the on-screen instructions
 - Launch Cursor AI and complete any additional setup prompts



Lab 07 Agenda

- Introduction to Generative Artificial Intelligence for Code Generation
 - Concepts
 - Leaderboard
 - Access
- Basic Techniques
 - Tab and Chat
 - Models
 - Context Management
 - Prompt Engineering
- Case Study

Tab and Chat

■ Cursor Tab

- Cursor Tab can suggest edits around your cursor position, including:
 - Insertions of additional code
 - Suggestions based on your recent changes
 - Recommendations based on linter errors

■ Inline Generation

- Cursor Chat allows you to ask questions or solve problems within your codebase using advanced language models, all within your editor. When no code is selected and you press Ctrl/Cmd K, Cursor will generate new code based on your input in the prompt bar.

■ Inline Edits

- For in-place edits, simply select the code you want to modify and enter your instructions in the prompt bar.

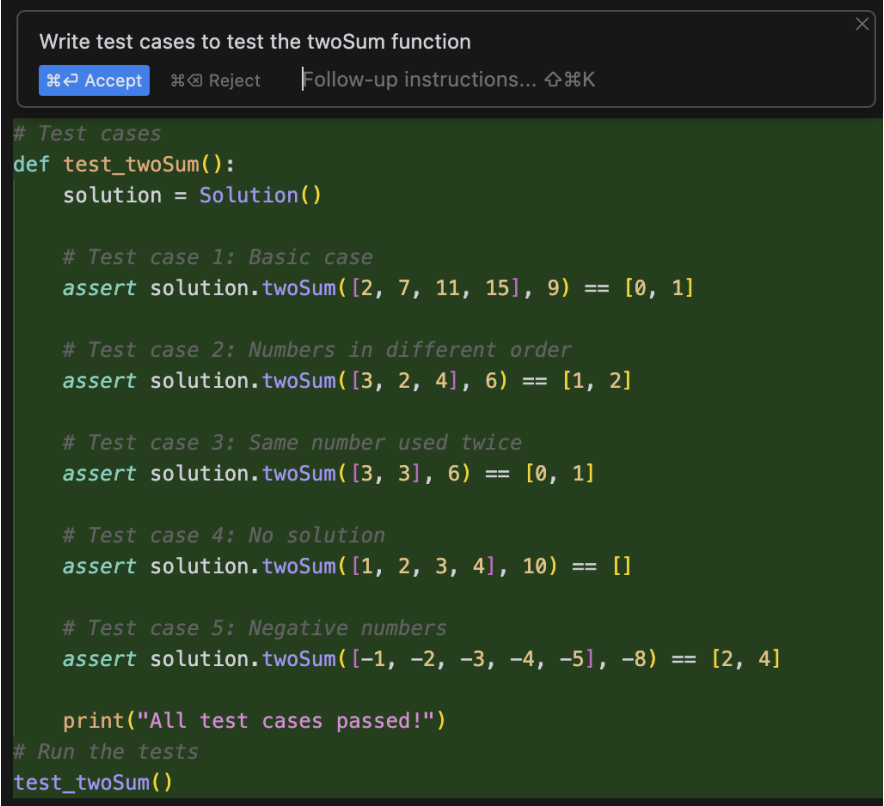
■ Cursor terminal

- In the built-in terminal, pressing Ctrl/Cmd K opens a prompt bar at the bottom of the terminal.

```
two_sum.py
1  # Given an array of integers nums and an integer target, return
   # indices of the two numbers such that they add up to target.
2
3  # You may assume that each input would have exactly one solution,
   # and you may not use the same element twice.
4
5  # You can return the answer in any order.
6
7  class Solution(object):
8      def twoSum(self, nums, target):
9          """
10             :type nums: List[int]
11             :type target: int
12             :rtype: List[int]
13             """
14             for i in range(len(nums)):
15                 for j in range(i + 1, len(nums)):
16                     if nums[i] + nums[j] == target:
17                         return [i, j]
18             return []
```

Tab and Chat

- **Cursor Tab**
 - Cursor Tab can suggest edits around your cursor position.
- **Inline Generation**
 - Cursor Chat allows you to ask questions or solve problems within your codebase using advanced language models, all within your editor. When no code is selected and you press Ctrl/Cmd K, Cursor will generate new codes based on your input in the prompt bar.
 - Write test cases to test the twoSum function.
- **Inline Edits**
 - For in-place edits, simply select the code you want to modify and enter your instructions in the prompt bar.
- **Cursor terminal**
 - In the built-in terminal, pressing Ctrl/Cmd K opens a prompt bar at the bottom of the terminal.



```
Write test cases to test the twoSum function
Accept Reject Follow-up instructions... K

# Test cases
def test_twoSum():
    solution = Solution()

    # Test case 1: Basic case
    assert solution.twoSum([2, 7, 11, 15], 9) == [0, 1]

    # Test case 2: Numbers in different order
    assert solution.twoSum([3, 2, 4], 6) == [1, 2]

    # Test case 3: Same number used twice
    assert solution.twoSum([3, 3], 6) == [0, 1]

    # Test case 4: No solution
    assert solution.twoSum([1, 2, 3, 4], 10) == []

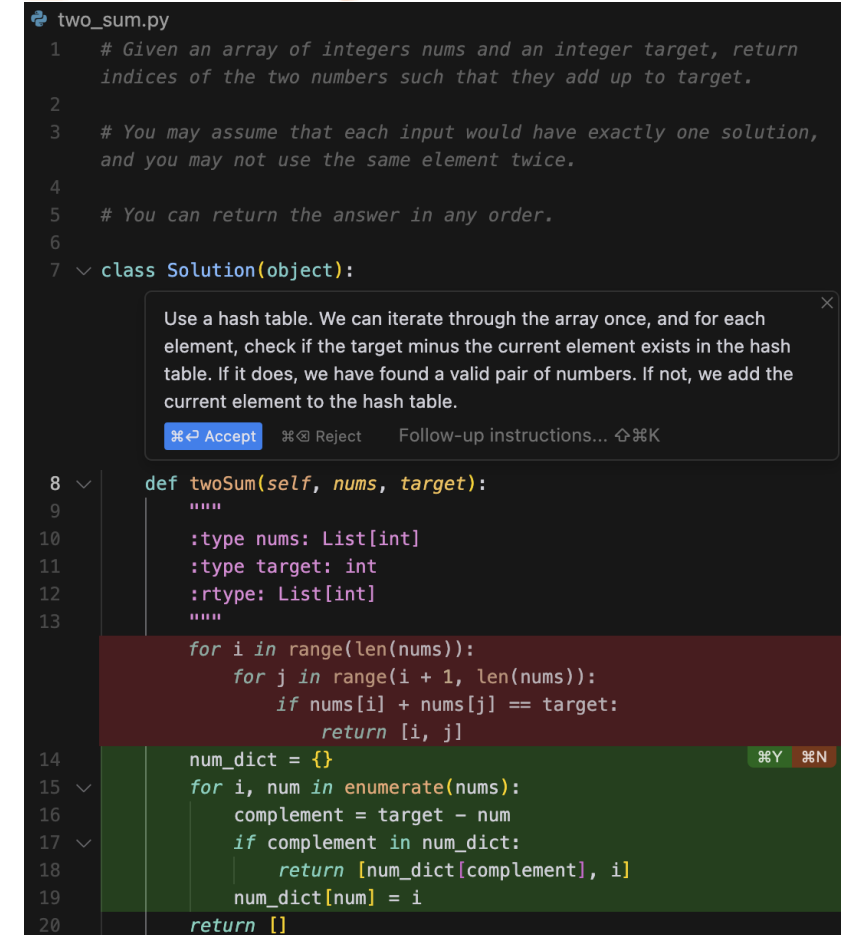
    # Test case 5: Negative numbers
    assert solution.twoSum([-1, -2, -3, -4, -5], -8) == [2, 4]

    print("All test cases passed!")

# Run the tests
test_twoSum()
```

Tab and Chat

- **Cursor Tab**
 - Cursor Tab can suggest edits around your cursor position.
- **Inline Generation**
 - Cursor Chat allows you to ask questions or solve problems within your codebase using advanced language models, all within your editor. When no code is selected and you press Ctrl/Cmd K, Cursor will generate new codes based on your input in the prompt bar.
- **Inline Edits**
 - For in-place edits, simply select the code you want to modify and enter your instructions in the prompt bar.
 - Use a hash table. We can iterate through the array once, and for each element, check if the target minus the current element exists in the hash table. If it does, we have found a valid pair of numbers. If not, we add the current element to the hash table.
- **Cursor terminal**
 - In the built-in terminal, pressing Ctrl/⌘ K opens a prompt bar at the bottom of the terminal.



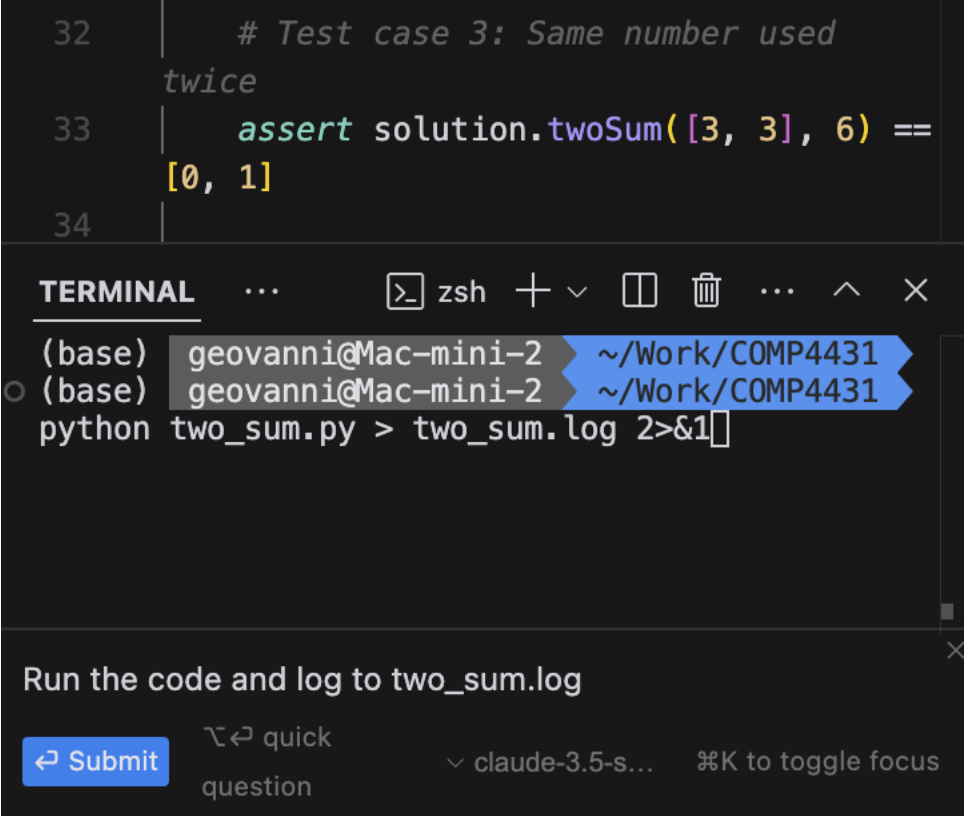
```
two_sum.py
1  # Given an array of integers nums and an integer target, return
  indices of the two numbers such that they add up to target.
2
3  # You may assume that each input would have exactly one solution,
  and you may not use the same element twice.
4
5  # You can return the answer in any order.
6
7  class Solution(object):
8
9      def twoSum(self, nums, target):
10         """
11         :type nums: List[int]
12         :type target: int
13         :rtype: List[int]
14         """
15         num_dict = {}
16         for i, num in enumerate(nums):
17             complement = target - num
18             if complement in num_dict:
19                 return [num_dict[complement], i]
20             num_dict[num] = i
21         return []
```

Use a hash table. We can iterate through the array once, and for each element, check if the target minus the current element exists in the hash table. If it does, we have found a valid pair of numbers. If not, we add the current element to the hash table.

Accept Reject Follow-up instructions... ⌘K

Tab and Chat

- **Cursor Tab**
 - Cursor Tab can suggest edits around your cursor position.
- **Inline Generation**
 - Cursor Chat allows you to ask questions or solve problems within your codebase using advanced language models, all within your editor. When no code is selected and you press Ctrl/Cmd K, Cursor will generate new code based on your input in the prompt bar.
- **Inline Edits**
 - For in-place edits, simply select the code you want to modify and enter your instructions in the prompt bar.
- **Cursor terminal**
 - In the built-in terminal, pressing Ctrl/⌘ K opens a prompt bar at the bottom of the terminal. You can:
 - Accept the command by pressing Esc
 - Execute the command immediately using Ctrl/⌘ + Enter
 - Run the code and log to two_sum.log



```
32 | # Test case 3: Same number used
    | twice
33 | assert solution.twoSum([3, 3], 6) ==
    | [0, 1]
34 |
```

TERMINAL ... `>_ zsh` `+` `∨` `□` `🗑` `...` `^` `×`

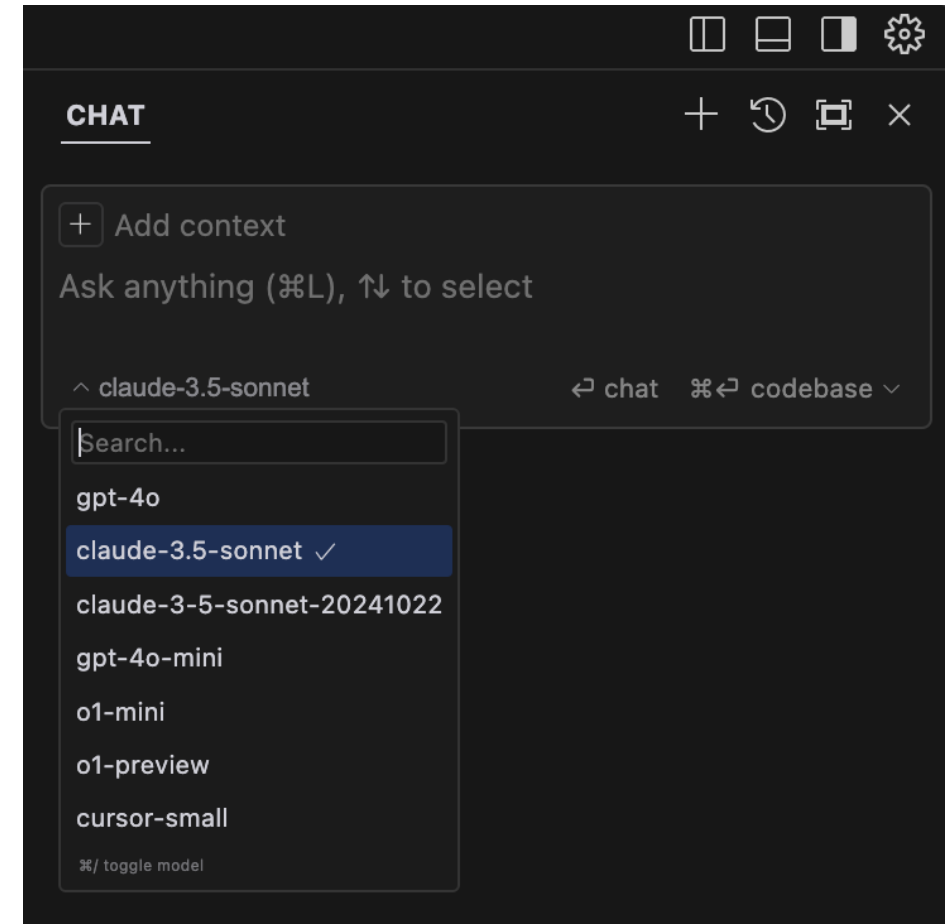
```
(base) geovanni@Mac-mini-2 ~/Work/COMP4431
(base) geovanni@Mac-mini-2 ~/Work/COMP4431
python two_sum.py > two_sum.log 2>&1
```

Run the code and log to two_sum.log

`↩ Submit` `⌘↩ quick question` `∨ claude-3.5-s...` `⌘K to toggle focus`

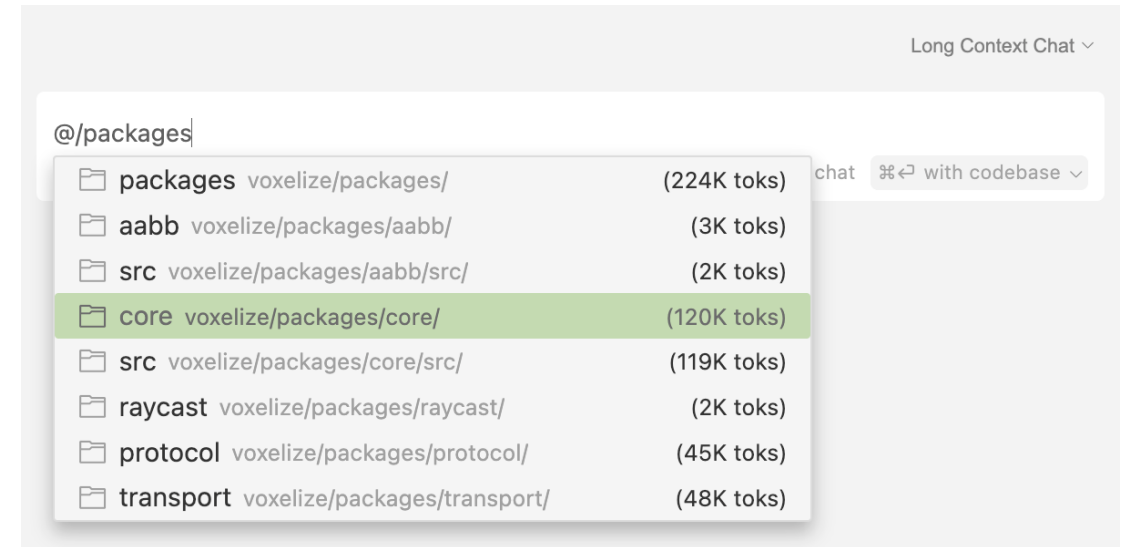
Models

- Underneath the AI input box, you will see a dropdown that allows you to select the model you want to use.
- By default, Cursor has these models ready to use:
 - gpt-4o
 - gpt-4
 - claude 3.5 Sonnet
 - cursor-small
- In long context chat, the model selection is limited to the models that support long context:
 - gpt-4o-128k
 - gemini-1.5-flash-500k
 - claude-3-haiku-200k
 - claude-3-sonnet-200k
 - claude-3-5-sonnet-200k
- You can also set your own Azure API key so that you will be using Azure OpenAI models at your own cost.
 - <https://docs.cursor.com/advanced/api-keys>



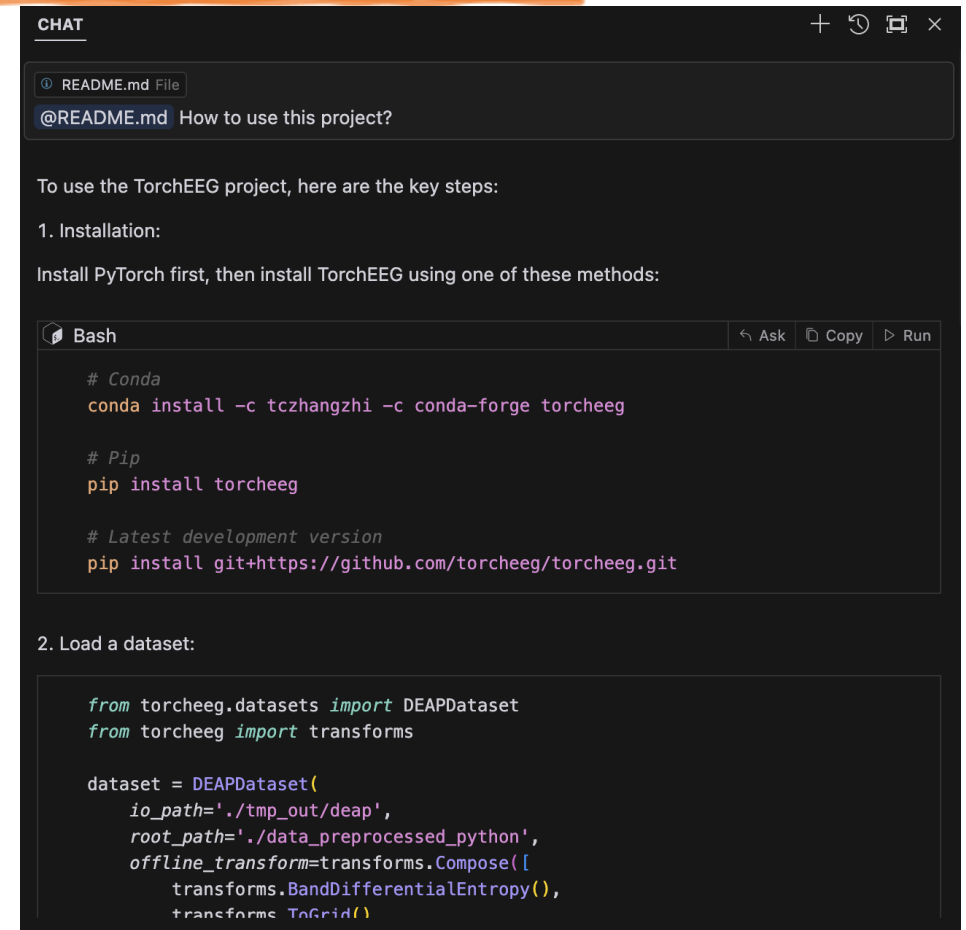
Context Management

- For language models to give good answers, they need to know specific things that are relevant to your codebase — context.
 - Cursor has several **built-in features to provide context in chat**, such as:
 - automatically including context across your entire codebase
 - searching the web
 - indexing documentation, and user-specified references to code blocks
 - You can **use @ symbols by typing @**. A popup menu will appear with a list of suggestions, and it will automatically filter to only show the most relevant suggestions based on your input.
 - @Files
 - @Folders
 - @Code
 - @Docs
 - @Web



Context Management

- Cursor has several built-in features to provide context. You can use @ symbols by typing @.
 - @Files
 - you can reference entire files by using @Files.
 - @Folders
 - You can reference entire folders by using @Folders.
 - @Code
 - You can select the code you want to reference, and click on Add to Chat (Ctrl/⌘ Shift L)



The screenshot shows the Cursor AI chat window. At the top, there's a header 'CHAT' with window controls. Below it, a file explorer shows 'README.md File' selected. The chat input field contains '@README.md How to use this project?'. The chat response provides instructions for using the TorchEEG project, including installation steps and code snippets for loading a dataset.

```
CHAT
+ ↺ 🖨 ✕

① README.md File
@README.md How to use this project?

To use the TorchEEG project, here are the key steps:

1. Installation:

Install PyTorch first, then install TorchEEG using one of these methods:

Bash Ask Copy Run

# Conda
conda install -c tczhangzhi -c conda-forge torcheeg

# Pip
pip install torcheeg

# Latest development version
pip install git+https://github.com/torcheeg/torcheeg.git

2. Load a dataset:

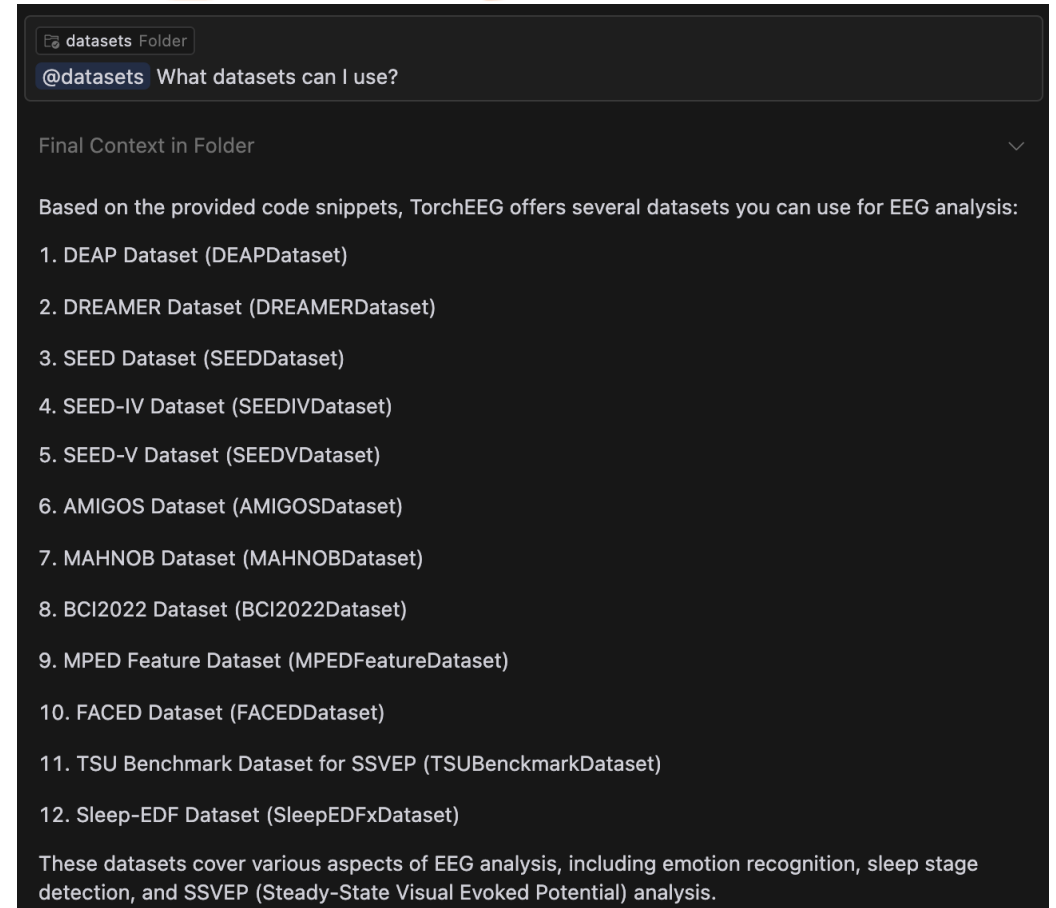
from torcheeg.datasets import DEAPDataset
from torcheeg import transforms

dataset = DEAPDataset(
    io_path='./tmp_out/deap',
    root_path='./data_preprocessed_python',
    offline_transform=transforms.Compose([
        transforms.BandDifferentialEntropy(),
        transforms.ToGrid()
```

Please refer to <https://github.com/torcheeg/torcheeg> for the demo project

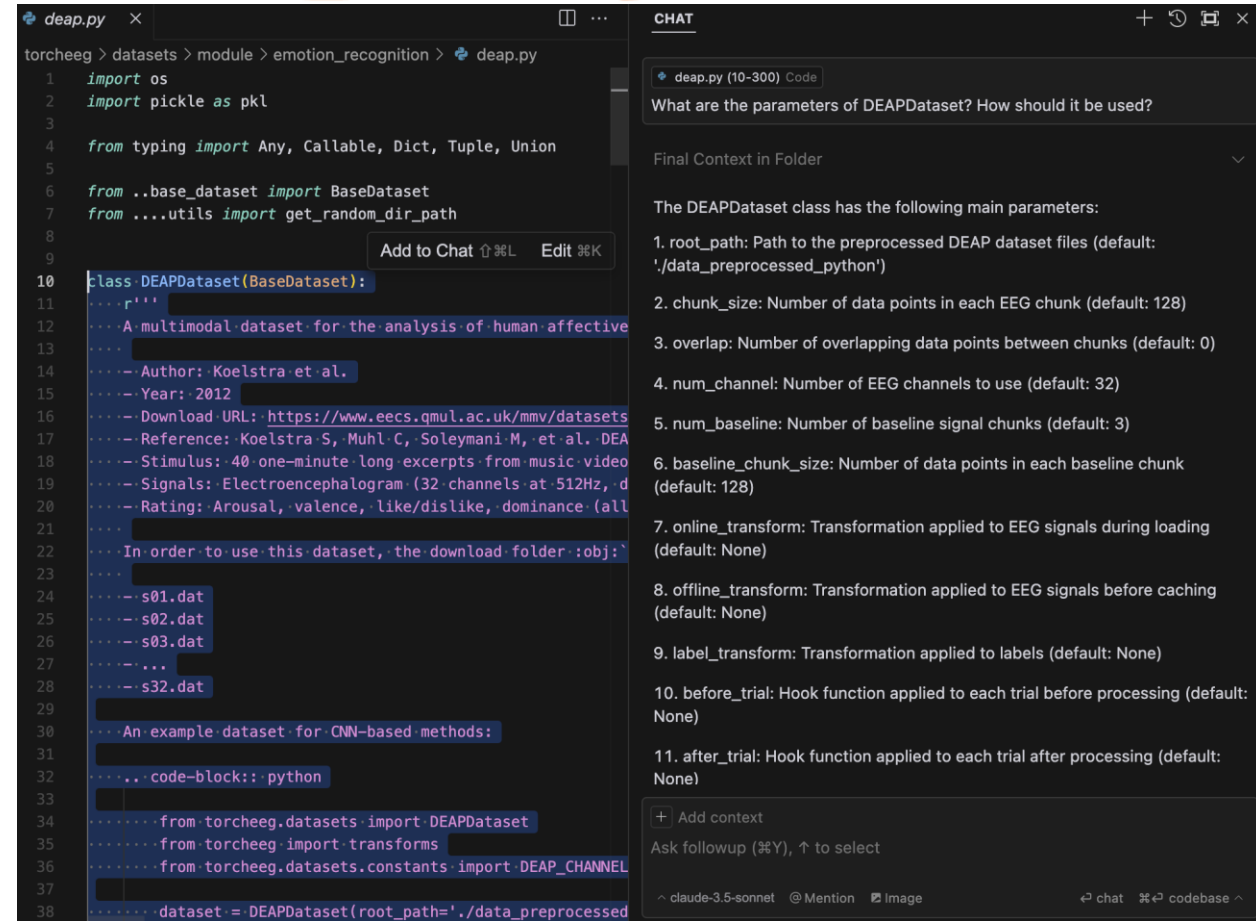
Context Management

- Cursor has several built-in features to provide context. You can use @ symbols by typing @.
 - @Files
 - you can reference entire files by using @Files.
 - @Folders
 - You can reference entire folders by using @Folders.
 - @Code
 - You can select the code you want to reference, and click on Add to Chat (Ctrl/⌘ Shift L)



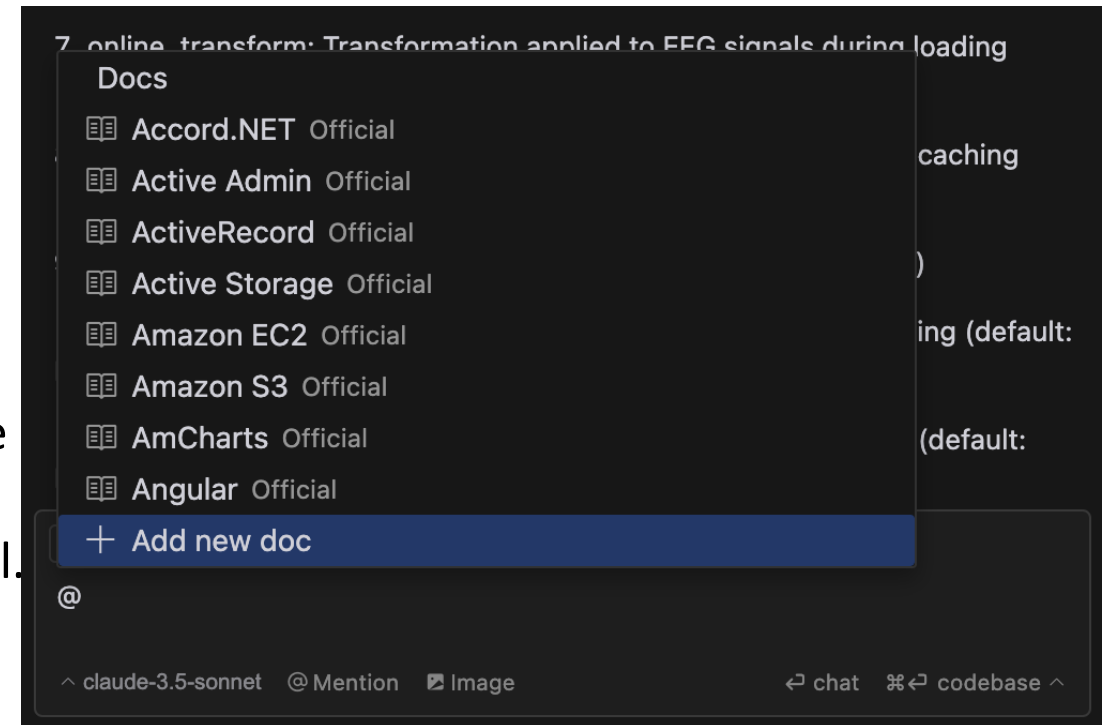
Context Management

- Cursor has several built-in features to provide context. You can use @ symbols by typing @.
 - @Files
 - you can reference entire files by using @Files.
 - @Folders
 - You can reference entire folders by using @ Folders.
 - @Code
 - You can select the code you want to reference, and click on Add to Chat (Ctrl/⌘ Shift L)



Context Management

- Cursor has several built-in features to provide context. You can use @symbols by typing @.
 - Cursor comes with a set of third-party docs crawled, indexed, and ready to be used as context.
- If you want to crawl and index custom docs that are not already provided, you can do so by @Docs > Add new doc.
- The modal will appear after you've pasted in the URL of your desired doc.
- You can access them by using the @Docs symbol.



Context Management

- Cursor has several built-in features to provide context. You can use @symbols by typing @.
 - Cursor comes with a set of third-party docs crawled, indexed, and ready to be used as context.
 - If you want to crawl and index custom docs that are not already provided, you can do so by @Docs > Add new doc.
 - The modal will appear after you've pasted in the URL of your desired doc.
 - You can access them by using the @Docs symbol.

CHAT
State Visual Evoked Potential) analysis.

Add new doc

@ NAME PREFIX

TorchEEG https://torcheeg.readthedocs.io/en/er

ENTRYPOINT

https://torcheeg.readthedocs.io/en/latest/

Confirm

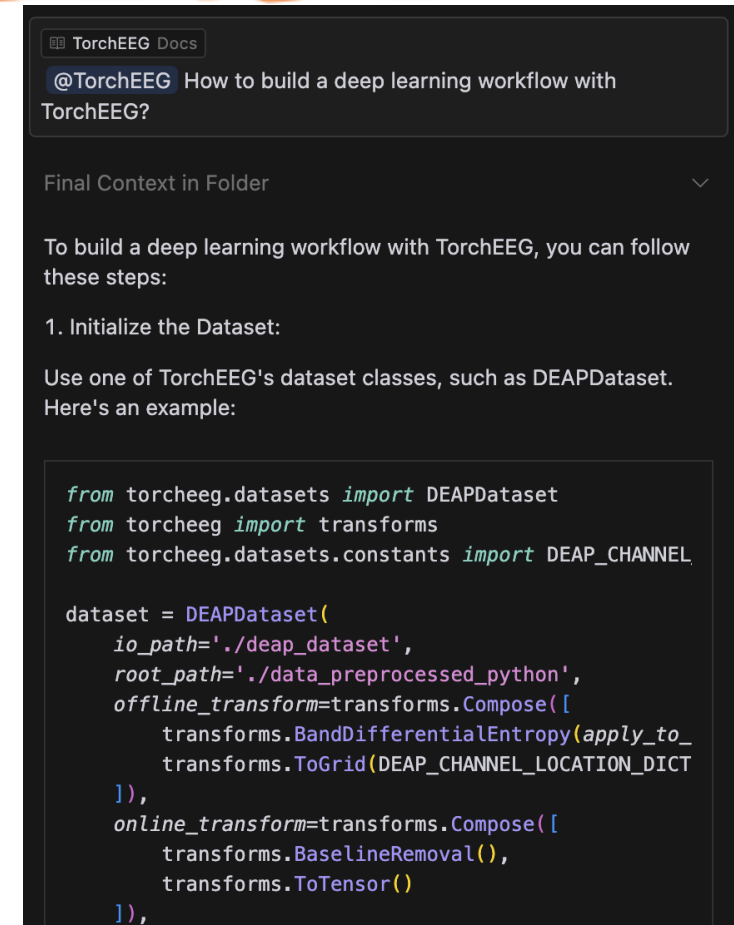
options, which you can configure when initializing the dataset object. The datasets are designed to work seamlessly with the TorchEEG framework, providing easy access to EEG data for your deep learning models.

deap.py (10-300) Code

What are the parameters of DEAPDataset? How should it be used?

Context Management

- Cursor has several built-in features to provide context. You can use @symbols by typing @.
 - Cursor comes with a set of third-party docs crawled, indexed, and ready to be used as context.
 - If you want to crawl and index custom docs that are not already provided, you can do so by @Docs > Add new doc.
 - The modal will appear after you've pasted in the URL of your desired doc.
 - You can access them by using the @Docs symbol.



TorchEEG Docs

@TorchEEG How to build a deep learning workflow with TorchEEG?

Final Context in Folder

To build a deep learning workflow with TorchEEG, you can follow these steps:

1. Initialize the Dataset:

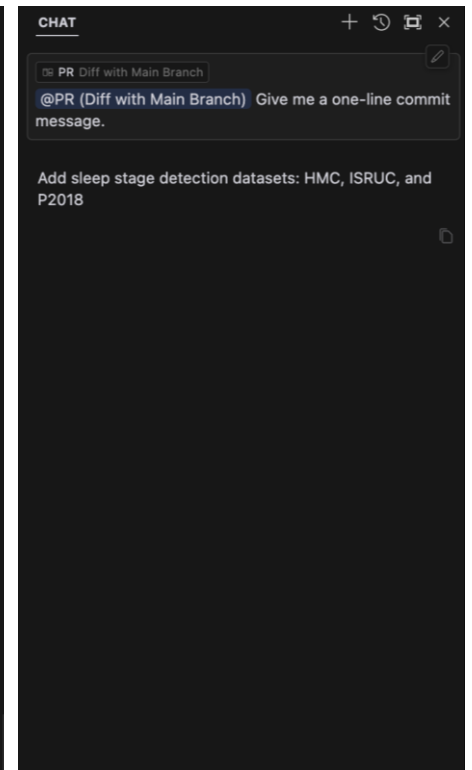
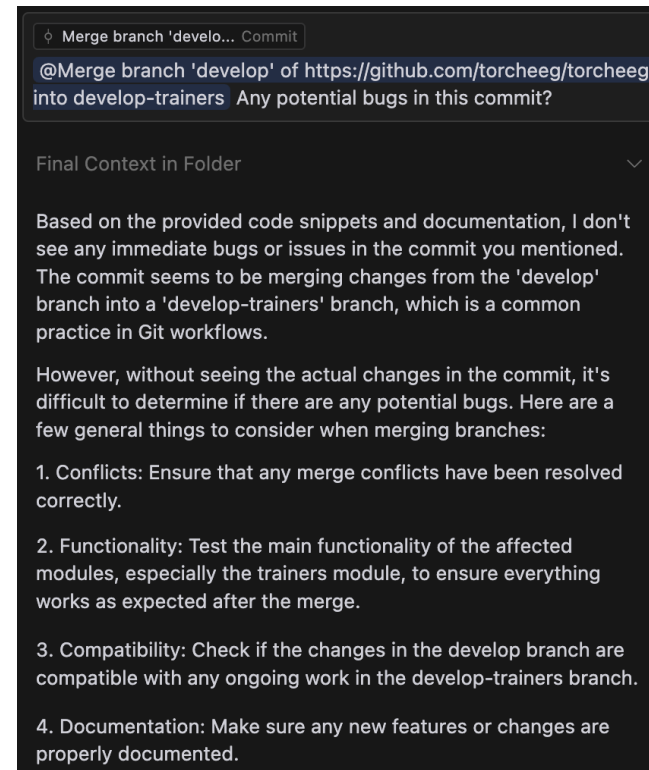
Use one of TorchEEG's dataset classes, such as DEAPDataset. Here's an example:

```
from torcheeg.datasets import DEAPDataset
from torcheeg import transforms
from torcheeg.datasets.constants import DEAP_CHANNEL

dataset = DEAPDataset(
    io_path='./deap_dataset',
    root_path='./data_preprocessed_python',
    offline_transform=transforms.Compose([
        transforms.BandDifferentialEntropy(apply_to=
        transforms.ToGrid(DEAP_CHANNEL_LOCATION_DICT
    ]),
    online_transform=transforms.Compose([
        transforms.BaselineRemoval(),
        transforms.ToTensor()
    ]),
```

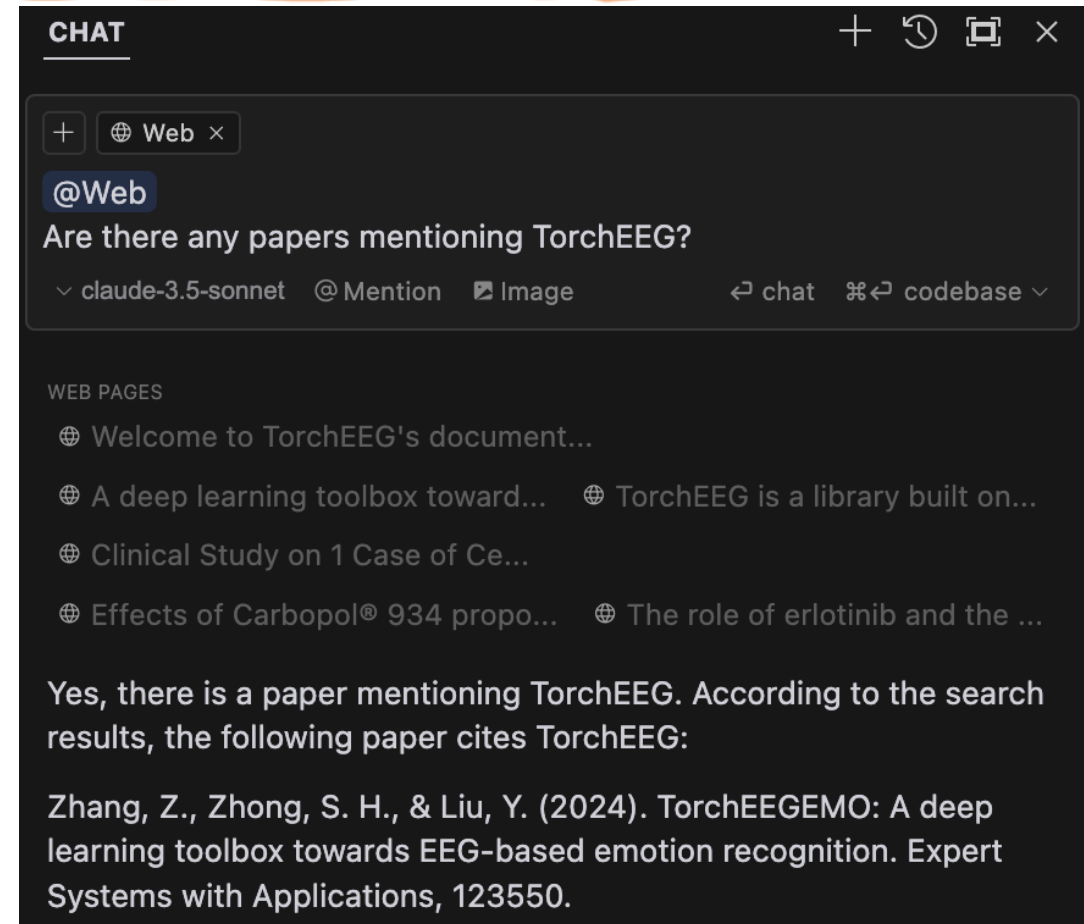
Context Management

- Cursor has several built-in features to provide context. You can use @symbols by typing @.
 - Cursor comes with a set of third-party docs crawled, indexed, and ready to be used as context.
 - If you want to crawl and index custom docs that are not already provided, you can do so by @Docs > Add new doc.
 - The modal will appear after you've pasted in the URL of your desired doc.
 - You can access them by using the @Docs symbol.



Context Management

- Cursor has several built in features to provide context. You can use @ symbols by typing @.
 - With @Web, Cursor constructs a search query based on the query and the context you've provided, and searches the web to find relevant information as additional context.
- This is particularly useful for finding the most up-to-date information.



Prompt Engineering

- Start general, then get specific.
 - When writing a prompt for LLM coder, first give LLM coder a broad description of the goal or scenario. Then list any specific requirements.
 - Write a function that tells me if a number is prime
 - The function should take an integer and return true if the integer is prime
 - The function should error if the input is not a positive integer
- Use examples to help LLM coder understand what you want.
 - You can provide example input data, example outputs, and example implementations.
 - Write a function that finds all dates in a string and returns them in an array.
 - Example:
 - findDates("I have a dentist appointment on 11/14/2023 and book club on 12-1-23")
 - Returns: ["11/14/2023", "12-1-23"]
- Break the process down into smaller tasks, and ask LLM coder to accomplish them one by one.

Write a function that tells me if a number is prime.

The function should take an integer and return true if the integer is prime.
The function should error if the input is not a positive integer.

Certainly! I'll write a Python function that determines if a number is prime and meets the requirements you specified. Here's the implementation:

```
def is_prime(n):  
    """  
    Check if a number is prime.  
  
    Args:  
    n (int): The number to check.  
  
    Returns:  
    bool: True if the number is prime, False otherwise.  
  
    Raises:  
    ValueError: If the input is not a positive integer.  
    """  
    if not isinstance(n, int) or n <= 0:  
        raise ValueError("Input must be a positive integer.")  
  
    if n == 1:  
        return False # 1 is not considered prime  
  
    if n == 2:
```


Prompt Engineering

- Start general, then get specific.
 - When writing a prompt for LLM coder, first give LLM coder a broad description of the goal or scenario. Then list any specific requirements.
 - Write a function that tells me if a number is prime
 - The function should take an integer and return true if the integer is prime
 - The function should error if the input is not a positive integer
- Use examples to help LLM coder understand what you want.
 - You can provide example input data, example outputs, and example implementations.
 - Write a function that finds all dates in a string and returns them in an array.
 - Example:
 - findDates("I have a dentist appointment on 11/14/2023 and book club on 12-1-23")
 - Returns: ["11/14/2023", "12-1-23"]
- Break the process down into smaller tasks, and ask LLM coder to accomplish them one by one.

Write a function that finds all dates in a string and returns them in an array.

Example:

findDates("I have a dentist appointment on 11/14/2023 and book club on 12-1-23")

Returns: ["11/14/2023", "12-1-23"]

Here's a Python function that finds all dates in a string and returns them in an array:

```
import re

def findDates(text):
    # Regular expression pattern to match various date formats
    date_pattern = r'\b(\d{1,2}[-/]\d{1,2}[-/]\d{2,4}|\d{4}'

    # Find all matches in the text
    dates = re.findall(date_pattern, text)

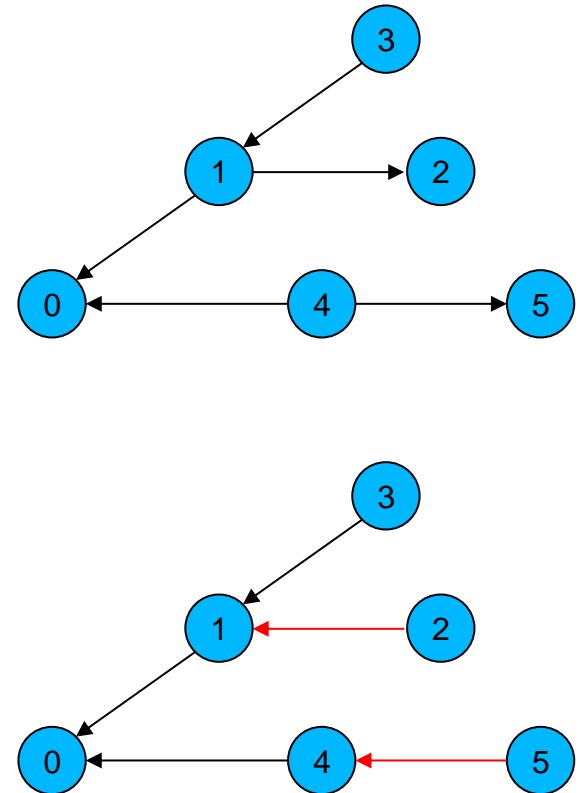
    return dates
```

Lab 07 Agenda

- Introduction to Generative Artificial Intelligence for Code Generation
 - Concepts
 - Leaderboard
 - Access
- Basic Techniques
 - Tab and Chat
 - Models
 - Context Management
 - Prompt Engineering
- Case Study

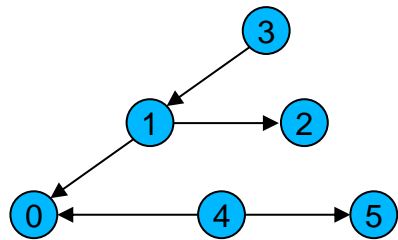
Case Study

- The Information Center has set up a network to allow **data uploads between servers**. This network can be represented by a set of permitted **directed transmissions**, where a permitted directed transmission from server a_i to server b_i can be formulated as $[a_i, b_i]$. It's given that $0 \leq a_i, b_i < n, a_i \neq b_i$. There are n ($n \geq 2$) servers in the network with server 0 being the central server. **The number of directed transmissions is less than n .**
- The Information Center plans to update this network. The objective is to ensure the central server (0) can receive data uploads from every other server in the network. Your task is to **reorient some of these permitted transmissions** so that the **central server can receive data from other servers**. Return the minimum number of transmission changes required.
- If **it's not feasible** to enable the central server to receive data from other servers by adjusting the directions of existing permitted transmissions, then **return -1**.
- Example:
 - Input: transmissions = $[[1, 0], [3, 1], [1, 2], [4, 0], [4, 5]]$, num_servers=6
 - Expected output: 2
 - Transmission changes are demonstrated in red lines to ensure the central server can receive data from all the other servers.

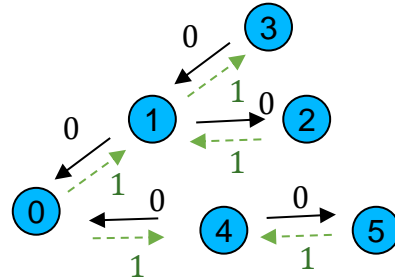


Case Study

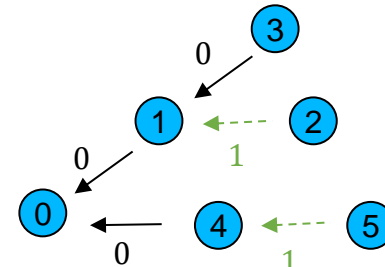
- Break the process down into smaller tasks, and ask LLM coder to accomplish them one by one.
 - Initialize a directed graph
 - Add nodes for each server
 - Add directed edges based on transmissions. Assign a weight of 0 to all edges, indicating indicating no change is needed, for the edge[0] to receive from edge[1]. Assign reverse edges with a weight of 1, indicating a change is needed, to allow the edge[0] to receive from the edge[1]
 - Determine the shortest paths from other servers to node 0
 - Collect edges in the shortest paths and their associated weights (indicating change cost)



original graph



weighted graph



Dijkstra's algorithm from other nodes to node 0

←--- need revise to get data message
←--- already can get data message

Case Study

- Create a template to define function inputs and outputs. Follow Google Python Style Guide for input/output descriptions (<https://google.github.io/styleguide/pyguide.html>)
 - Args: List each parameter by name.
 - Returns: Describe the semantics of the return value, including any type information that the type annotation does not provide.
 - Raises: List all exceptions that are relevant to the interface followed by a description.
- Use Python type hints for input/output data types:
 - Type annotations provide hints to tools but are not runtime enforced.
 - Use typing module for complex type annotation.s (<https://docs.python.org/3/library/typing.html>)
- Never forget to write case studies.
transmissions = [[1, 0], [3, 1], [1, 2], [4,0], [4,5]]
num_servers = 6
answer = 2
result = solve(transmissions, num_servers)
assert result == answer, f"Test case 1: expected {answer}, got {result}"

```
case_study.py
15 import networkx as nx
16 from typing import List
17
18 def solve(transmissions: List[List[int]], num_servers: int) -> int:
19     """
20     Your solution to the problem goes in this function.
21
22     Args:
23         transmissions (List[List[int]]): The permitted transmission between servers,
24         e.g., [[1, 0], [3, 1], [1, 2], [4,0], [4,5]]
25         num_servers (int): The number of servers in the network, e.g., 6
26
27     Returns:
28         int: the minimum number of transmission changes required, e.g., 2
29     """
30     return -2
31
32 # test case 1
33 transmissions = [[1, 0], [3, 1], [1, 2], [4, 0], [4, 5]]
34 num_servers = 6
35 answer = 2
36 result = solve(transmissions, num_servers)
37 assert result == answer, f"Test case 1: expected {answer}, got {result}"
38 print('Passed test case 1...')
39
40 # test case 2
41 transmissions = [[1, 0], [3, 1], [2, 3]]
42 num_servers = 4
43 answer = 0
44 result = solve(transmissions, num_servers)
45 assert result == answer, f"Test case 2: expected {answer}, got {result}"
46 print('Passed test case 2...')
47
48 # test case 3
49 transmissions = [[1, 0], [0, 2], [3, 4]]
50 num_servers = 5
51 answer = -1
52 result = solve(transmissions, num_servers)
53 assert result == answer, f"Test case 3: expected {answer}, got {result}"
54 print('Passed test case 3...')
```

Case Study

■ Inline Edits on the template

- Initialize a directed graph
- Add nodes for each server
- Add directed edges based on transmissions, Assign a weight of 0 to all edges, indicating indicating no change is needed, for the edge[0] to receive from edge[1]. Assign reverse edges with a weight of 1, indicating a change is needed, to allow the edge[0] to receive from the edge[1].
- Determine the shortest paths from other servers to node 0
- Collect edges in the shortest paths and their associated weights (indicating change cost)

■ Inline Edits on the generated codes

- Use a set to record the edges that need to be flipped to avoid an edge being recorded multiple times

```
the 'edge[1]' Determine the shortest paths from other servers to node 0 Collect edges in the shortest paths and their associated weights (indicating change cost)
Accept Reject Follow-up instructions... K
6
2
"""
return -2
# Initialize directed graph
G = nx.DiGraph()

# Add nodes for each server
for i in range(num_servers):
    G.add_node(i)

# Add edges with weights
for edge in transmissions:
    # Original direction (weight 0 - no change needed)
    G.add_edge(edge[0], edge[1], weight=0)
    # Reverse direction (weight 1 - change needed)
    G.add_edge(edge[1], edge[0], weight=1)

# Find shortest paths from all nodes to node 0
total_changes = 0
unreachable = False

for node in range(1, num_servers):
    try:
        path = nx.shortest_path(G, node, 0, weight='weight')
        # Sum up weights along the path
        path_cost = sum(G[path[i]][path[i+1]]['weight'] for i in
range(len(path)-1))
        total_changes += path_cost
    except nx.NetworkXNoPath:
        unreachable = True
        break

return -1 if unreachable else total_changes
```

Case Study

- Inline Edits on the template
 - Initialize a directed graph
 - Add nodes for each server
 - Add directed edges based on transmissions. Assign a weight of 0 to all edges, indicating indicating no change is needed, for the edge[0] to receive from edge[1]. Assign reverse edges with a weight of 1, indicating a change is needed, to allow the edge[0] to receive from the edge[1].
 - Determine the shortest paths from other servers to node 0
 - Collect edges in the shortest paths and their associated weights (indicating change cost)
- Inline Edits on the generated codes
 - Use a set to record the edges that need to be flipped to avoid an edge being recorded multiple times

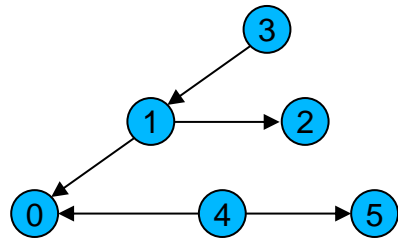
```
Use a set to record the edges that need to be flipped to avoid an edge being recorded multiple times
⌕↔ Accept ⌕⌂ Reject Follow-up instructions... ⌕⌕K

# Find shortest paths from all nodes to node 0
total_changes = 0
edges_to_flip = set()
unreachable = False

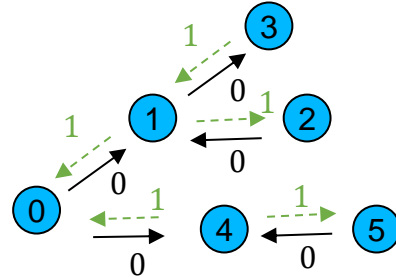
for node in range(1, num_servers):
    try:
        path = nx.shortest_path(G, node, 0, weight='weight')
        # Sum up weights along the path
        path_cost = sum(G[path[i]][path[i+1]]['weight'] for i in range(len(path)-1))
        total_changes += path_cost
        # Record edges that need to be flipped
        for i in range(len(path)-1):
            if G[path[i]][path[i+1]]['weight'] == 1:
                edges_to_flip.add((path[i], path[i+1]))
    except nx.NetworkXNoPath:
        unreachable = True
        break

return -1 if unreachable else total_changes
return -1 if unreachable else len(edges_to_flip)
```

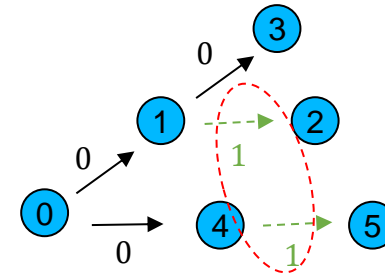
Case Study



original graph



weighted graph



Dijkstra's algorithm from
node 0 to others

←--- need revise to get data message
←--- already can get data message

- [Extensive Reading] The algorithm can be further optimized into a faster version:
 - Initialize a directed graph
 - Add nodes for each server
 - Assign a weight of 1 to all edges, indicating a change is needed, to allow the edge[0] to receive from the edge[1]
 - Add reverse edges with a weight of 0, indicating no change is needed, for the `edge[0]` to receive from edge[1]
 - Determine the shortest paths from node 0 to other servers
 - Collect edges in the shortest paths and their associated weights (indicating change cost)

Question Time!

Or See you next week!