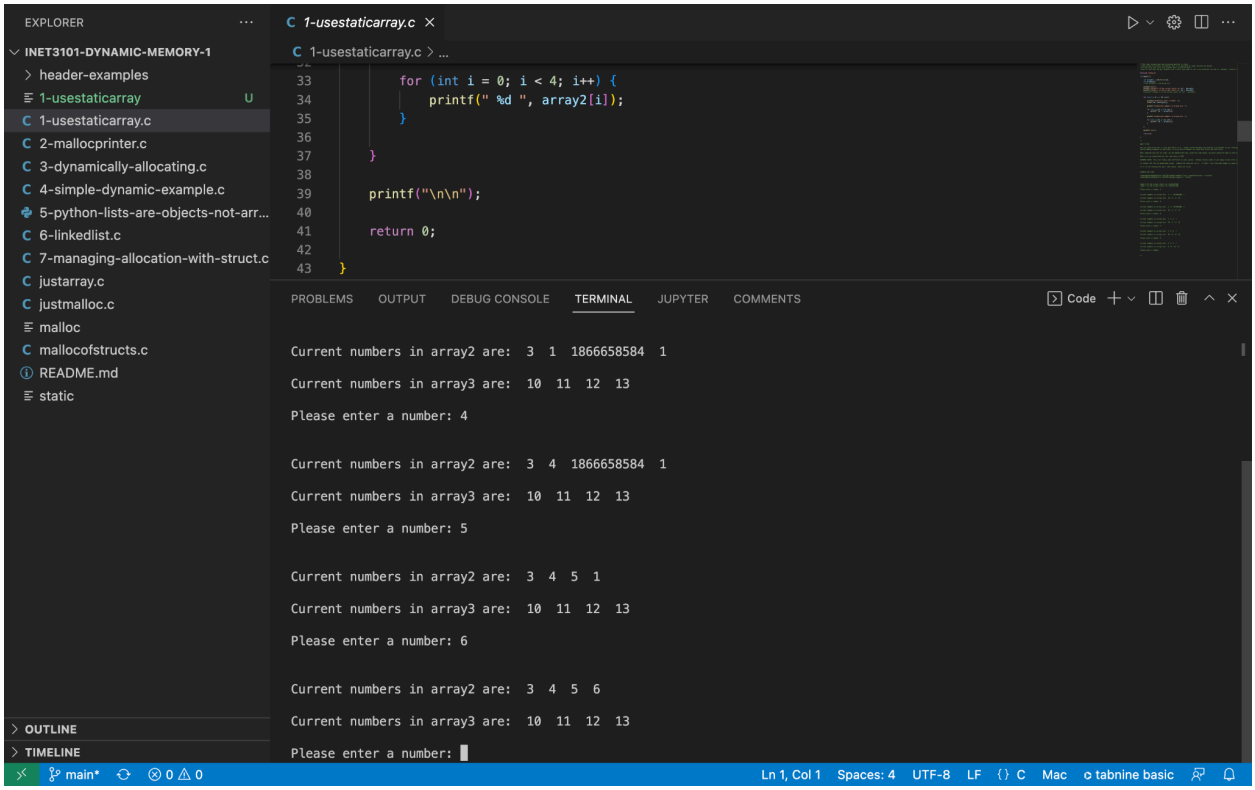
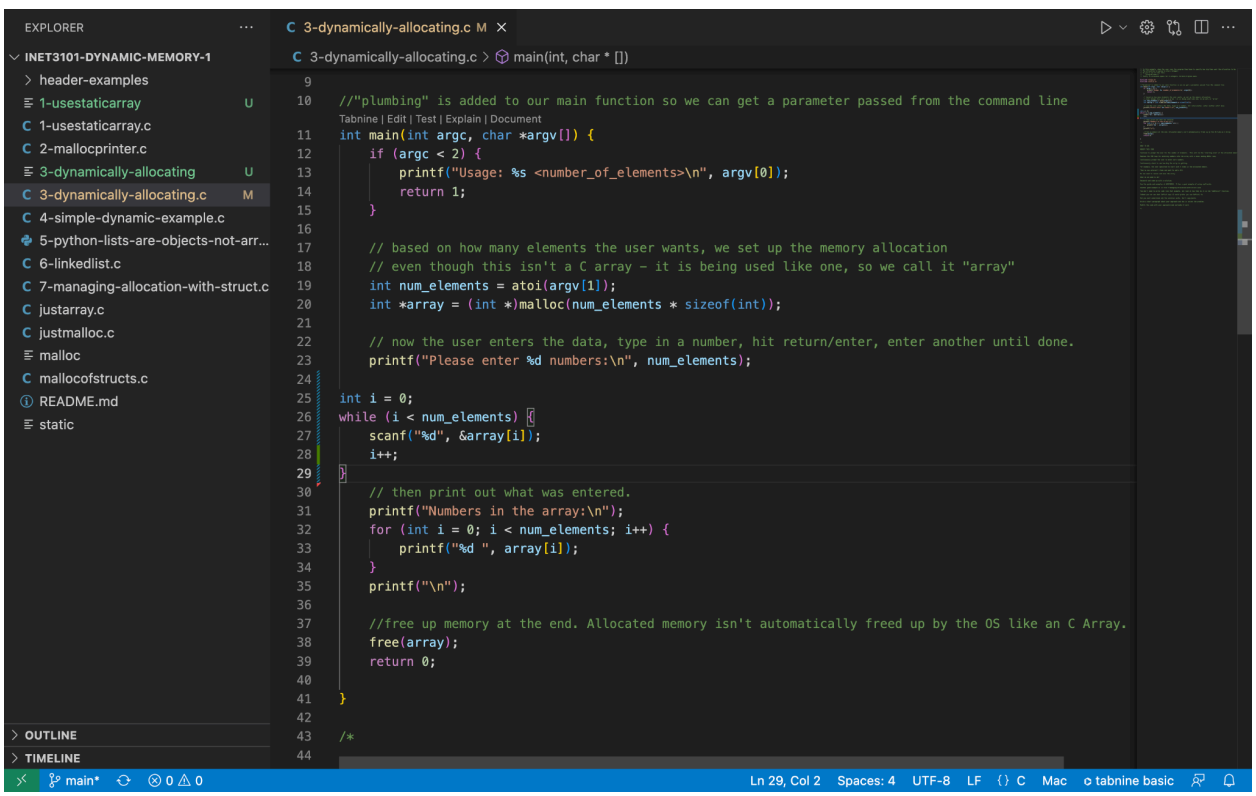
1.

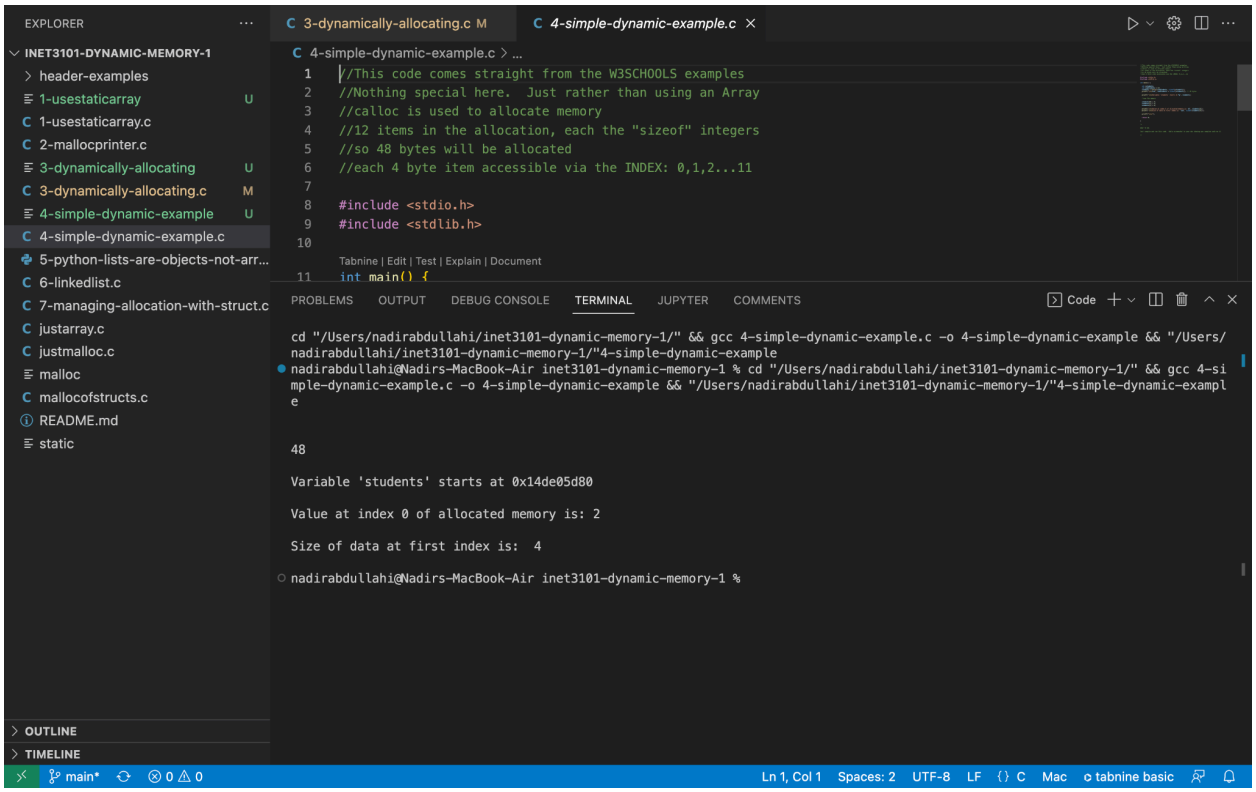

EXPLORER
INET3101-DYNAMIC-MEMORY-1
header-examples
1-usestaticarray
1-usestaticarray.c
2-mallocprinter.c
3-dynamically-allocating.c
4-simple-dynamic-example.c
5-python-lists-are-objects-not-arr...
6-linkedlist.c
7-managing-allocation-with-struct.c
justarray.c
justmalloc.c
malloc
mallocofstructs.c
README.md
static
OUTLINE
TIMELINE
1-usestaticarray.c
for (int i = 0; i < 4; i++) {
printf(" %d ", array2[i]);
}
}
printf("\n\n");
return 0;
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
COMMENTS
Current numbers in array2 are: 3 1 1866658584 1
Current numbers in array3 are: 10 11 12 13
Please enter a number: 4
Current numbers in array2 are: 3 4 1866658584 1
Current numbers in array3 are: 10 11 12 13
Please enter a number: 5
Current numbers in array2 are: 3 4 5 1
Current numbers in array3 are: 10 11 12 13
Please enter a number: 6
Current numbers in array2 are: 3 4 5 6
Current numbers in array3 are: 10 11 12 13
Please enter a number:
main*
Ln 1, Col 1
Spaces: 4
UTF-8
LF
C
Mac
tabnine basic

3.


EXPLORER
INET3101-DYNAMIC-MEMORY-1
header-examples
1-usestaticarray
1-usestaticarray.c
2-mallocprinter.c
3-dynamically-allocating
3-dynamically-allocating.c
4-simple-dynamic-example.c
5-python-lists-are-objects-not-arr...
6-linkedlist.c
7-managing-allocation-with-struct.c
justarray.c
justmalloc.c
malloc
mallocofstructs.c
README.md
static
OUTLINE
TIMELINE
3-dynamically-allocating.c
//"plumbing" is added to our main function so we can get a parameter passed from the command line
Tabnine | Edit | Test | Explain | Document
int main(int argc, char *argv[]) {
if (argc < 2) {
printf("Usage: %s <number_of_elements>\n", argv[0]);
return 1;
}
// based on how many elements the user wants, we set up the memory allocation
// even though this isn't a C array - it is being used like one, so we call it "array"
int num_elements = atoi(argv[1]);
int *array = (int *)malloc(num_elements * sizeof(int));
// now the user enters the data, type in a number, hit return/enter, enter another until done.
printf("Please enter %d numbers:\n", num_elements);
int i = 0;
while (i < num_elements) {
scanf("%d", &array[i]);
i++;
}
// then print out what was entered.
printf("Numbers in the array:\n");
for (int i = 0; i < num_elements; i++) {
printf("%d ", array[i]);
}
printf("\n");
//free up memory at the end. Allocated memory isn't automatically freed up by the OS like an C Array.
free(array);
return 0;
}
/*
main*
Ln 29, Col 2
Spaces: 4
UTF-8
LF
C
Mac
tabnine basic

In this case, it starts at the first index (`i = 0`) and keeps asking for numbers (`scanf("%d", &array[i])`), moving to the next position each time (`i++`).

4.



5. In Object-Oriented Programming (OOP), an object is a fundamental building block that represents a real-world entity or concept. It encapsulates data (attributes) and behaviors (methods) into a single unit, enabling modular and reusable code. Objects are instances of classes, which act as blueprints defining the structure and capabilities of the objects.

6. Linked lists helps us avoid using fixed sized arrays. With traditional arrays, you have to declare the size of it at the beginning, using a linked list you can use as you go. It also ensures you do not need to use memory reallocation because each node is stored separately and linked together.