DSA3 Lab session 1

Administrivia

- Lab time Fridays 14:00-18:00
- Lab focus will not be helping with assignments, but you can come and ask for help
- We’ll use 1-2 hours of lab time to review the week’s topics, do some exercises, talk about the new assignment

Coding assignments

- 10 assignments total, done in Python

- To pass the course:
  - All assignments completed
  - At least 60% completed on time (by assigned deadline)
  - Completed means passing all of the unit tests, not just writing some code that does some things

- There will also be one project, bigger than DSA3 assignments, not as big as the DSA2 project, details later

Coding assignments (cont.)

- You’re encouraged to work together in groups of two
  - You may not repeat partners over multiple assignments
  - ASK before you join someone’s repo, we don’t want any more Kdrama

- Working on an assignment alone is also fine

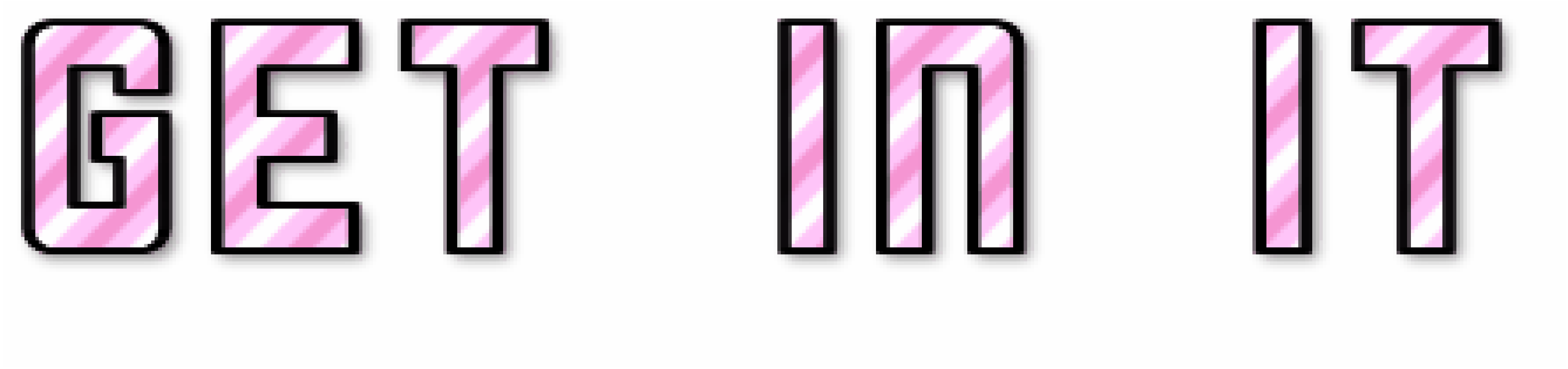
- You’re also allowed to pair up for the project
  - You can pair with anyone, including prior partners from assignments

Coding assignments (cont.)

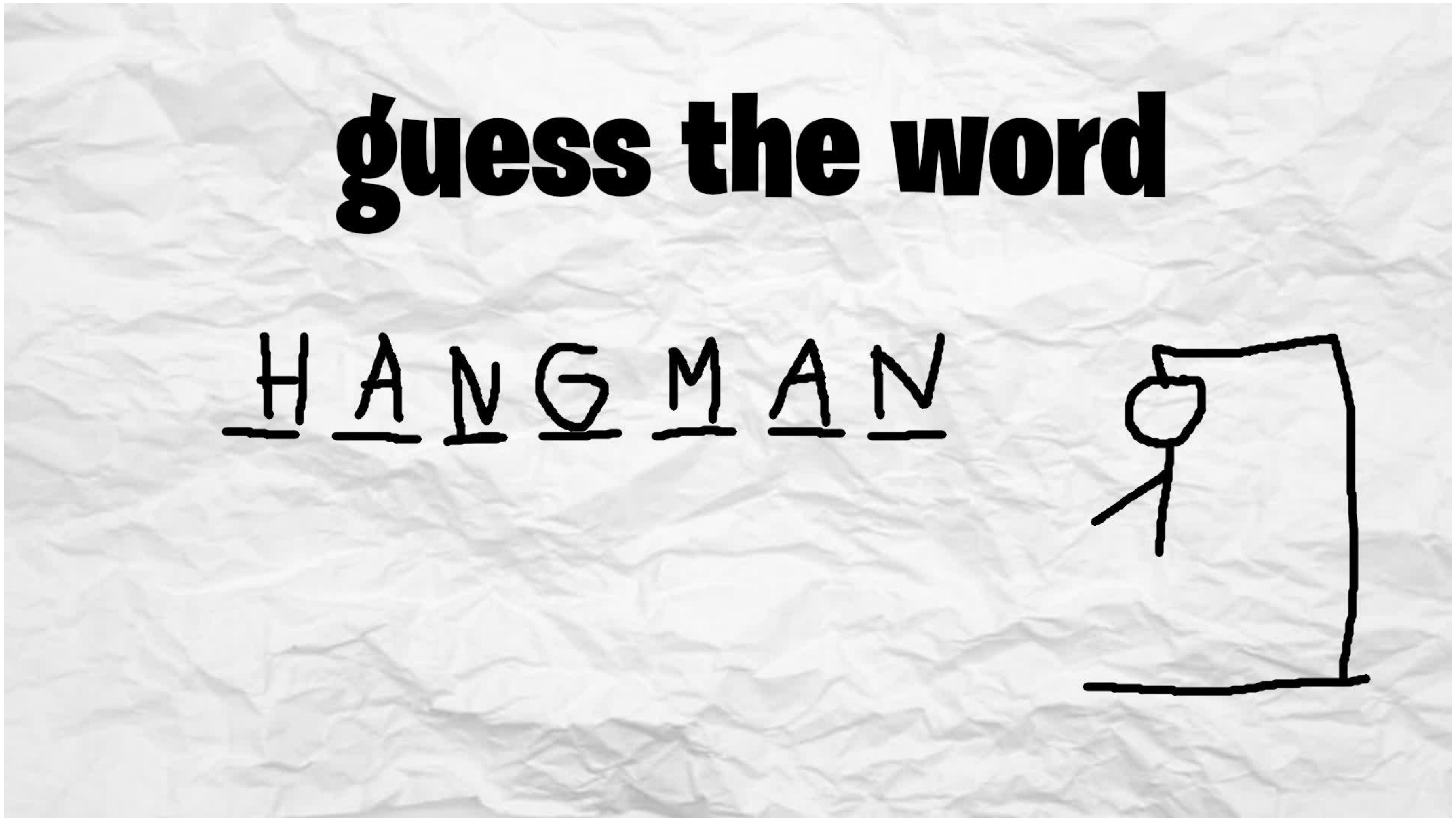
- Assignments released Fridays just before tutorial starts
- Assignments due the next NEXT Monday 14:00 just before class time
- This means you have 10 days per assignment
- This also means assignments will overlap

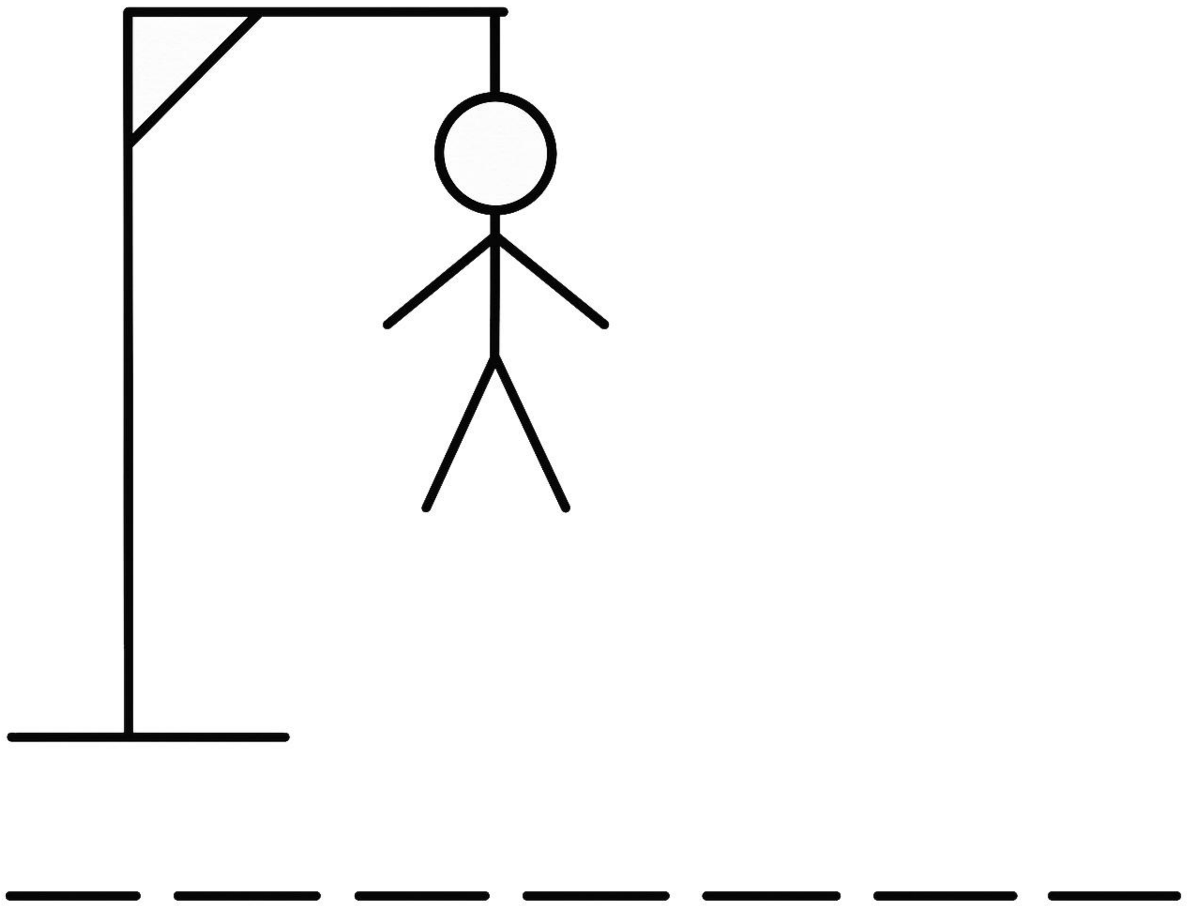
Last bit of administrivia

- If you are not yet in the DSACL3 Github team



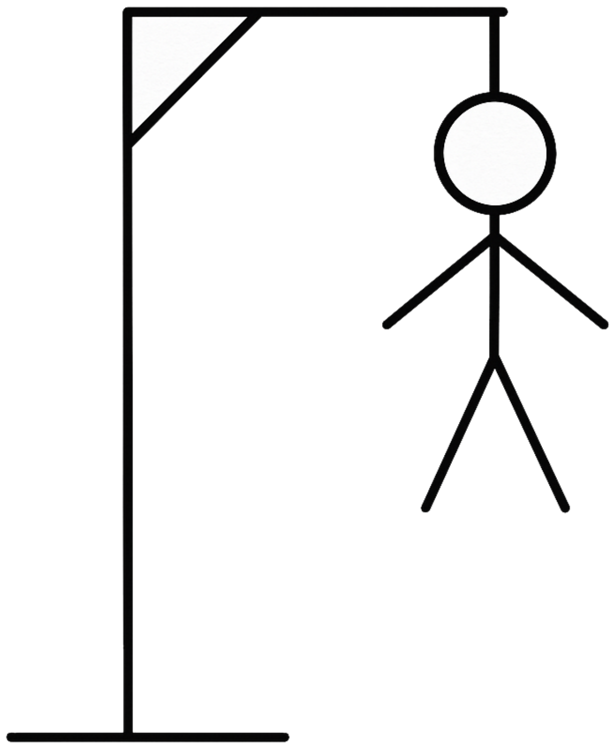
Let’s talk about assignment 1





What letter would you guess?

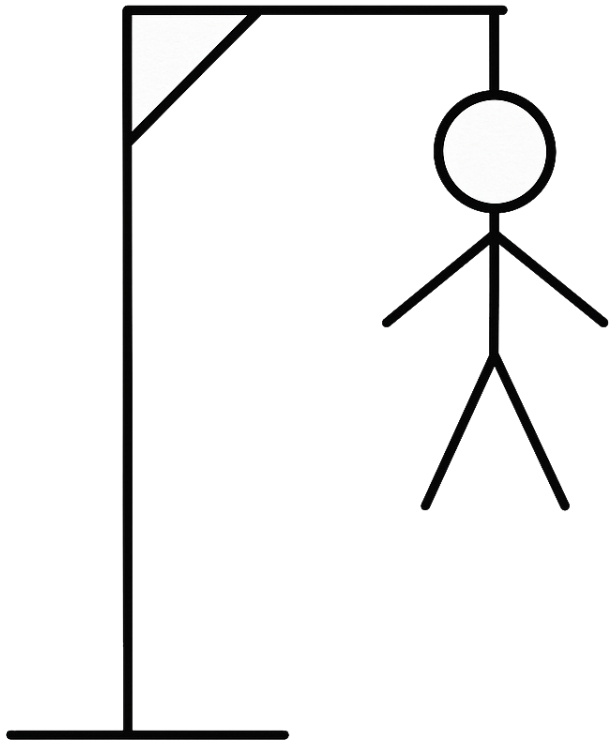
Why?



J U M P I ___ ___

What letter would you guess?

Why?



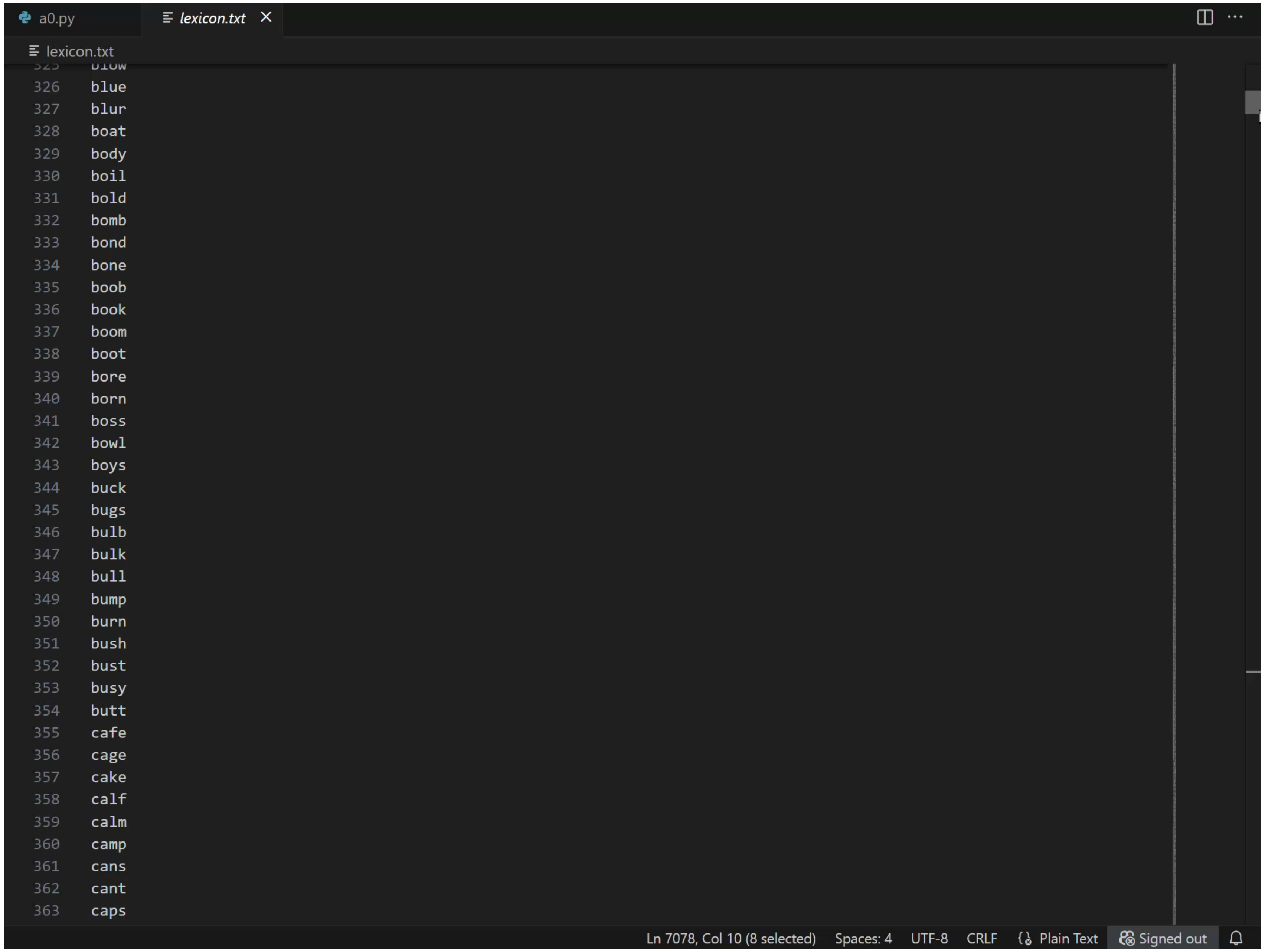
G U N N I _ _

What letter would you guess?

Why?

Assignment 1

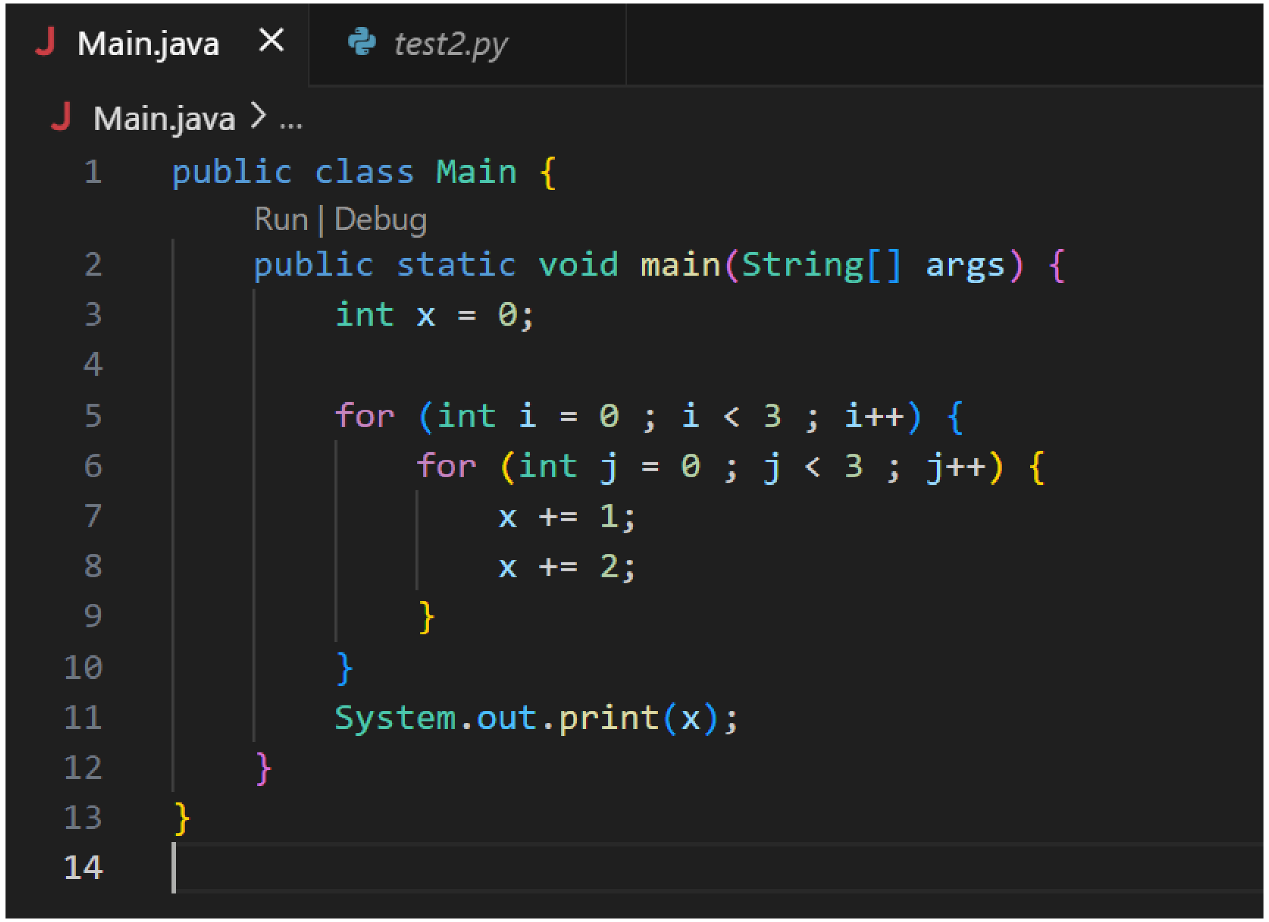
- We want to make guesses based on probability
- The probability distribution across letters changes depending on context
- But how do we establish a probability distribution?



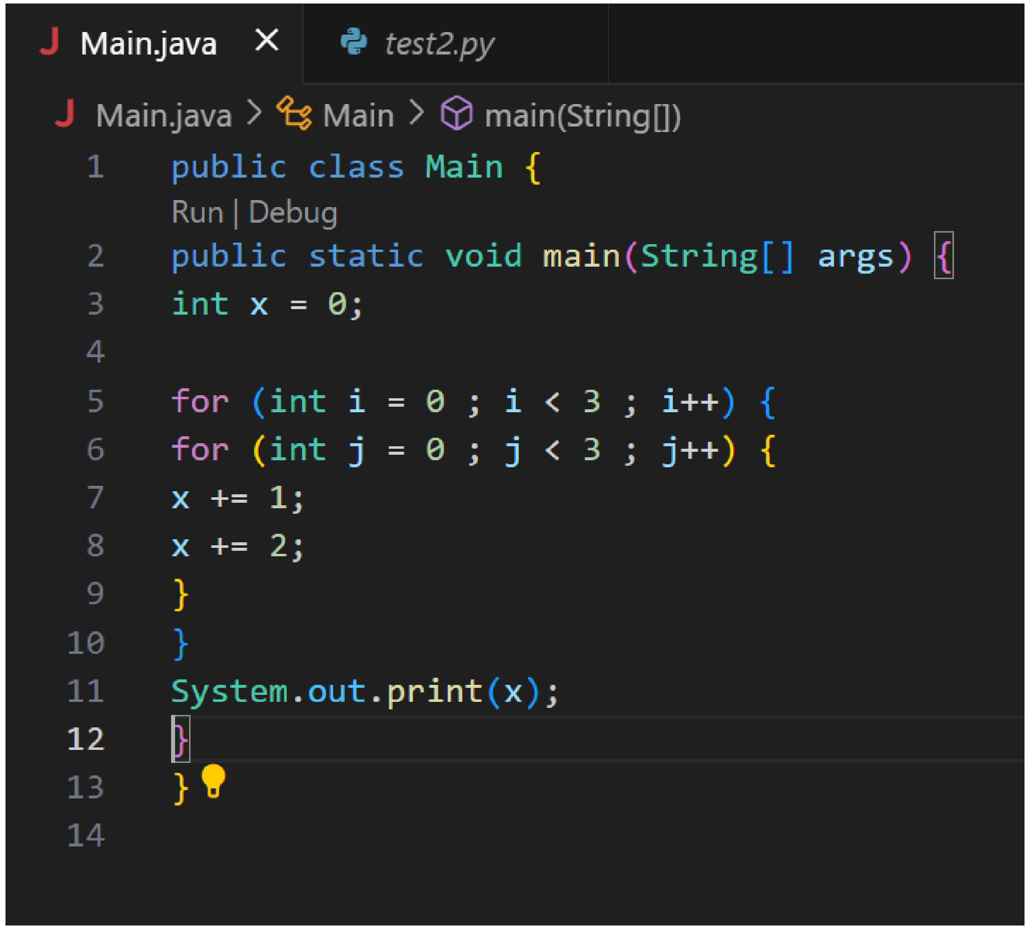
```
blue
blur
boat
body
boil
bold
bomb
bond
bone
boob
book
boom
boot
bore
born
boss
bowl
boys
buck
bugs
bulb
bulk
bull
bump
burn
bush
bust
busy
butt
cafe
cage
cake
calf
calm
camp
cans
cant
caps
```

A quick review of Python for proficient Java coders like yourselves

These two do exactly the same thing…

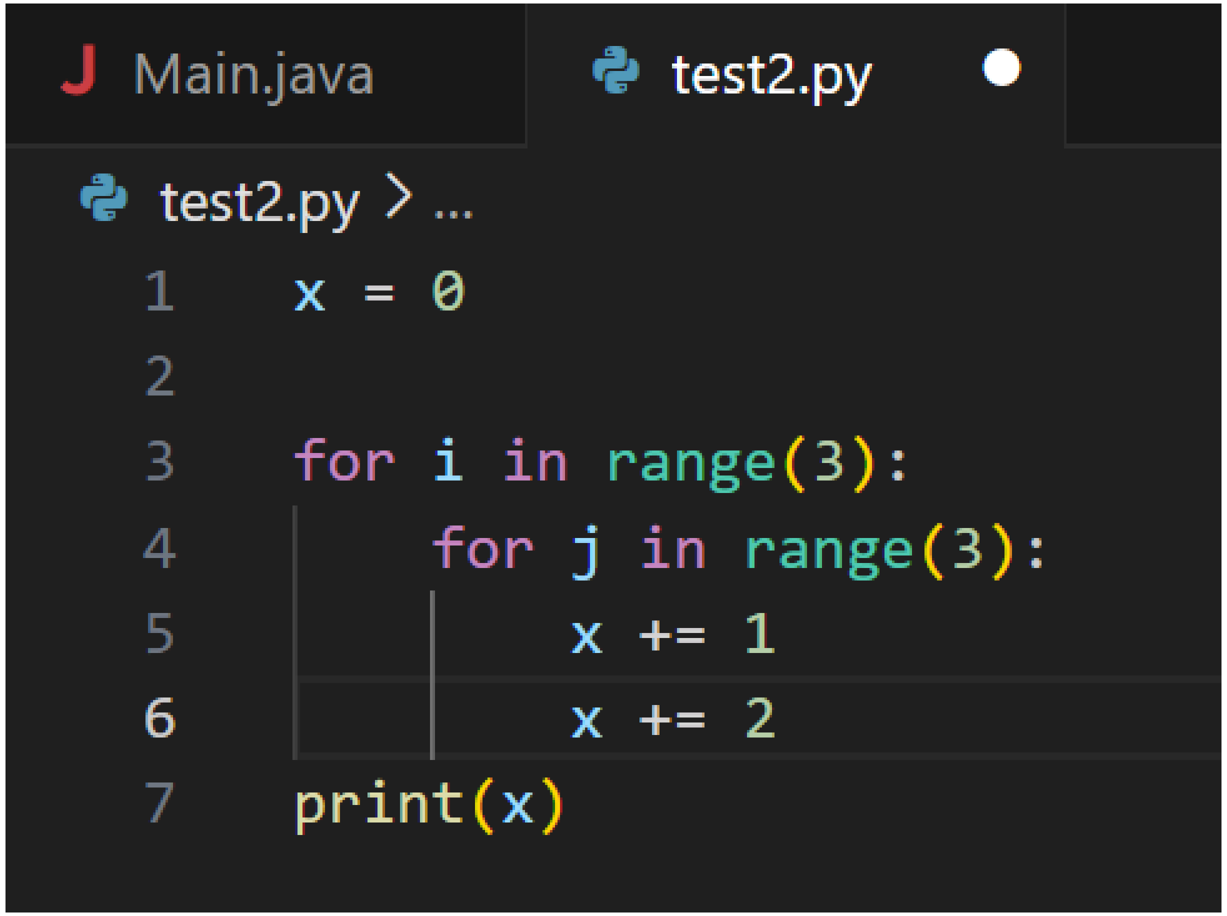


```java
public class Main {
    public static void main(String[] args) {
        int x = 0;

        for (int i = 0 ; i < 3 ; i++) {
            for (int j = 0 ; j < 3 ; j++) {
                x += 1;
                x += 2;
            }
        }
        System.out.print(x);
    }
}

```



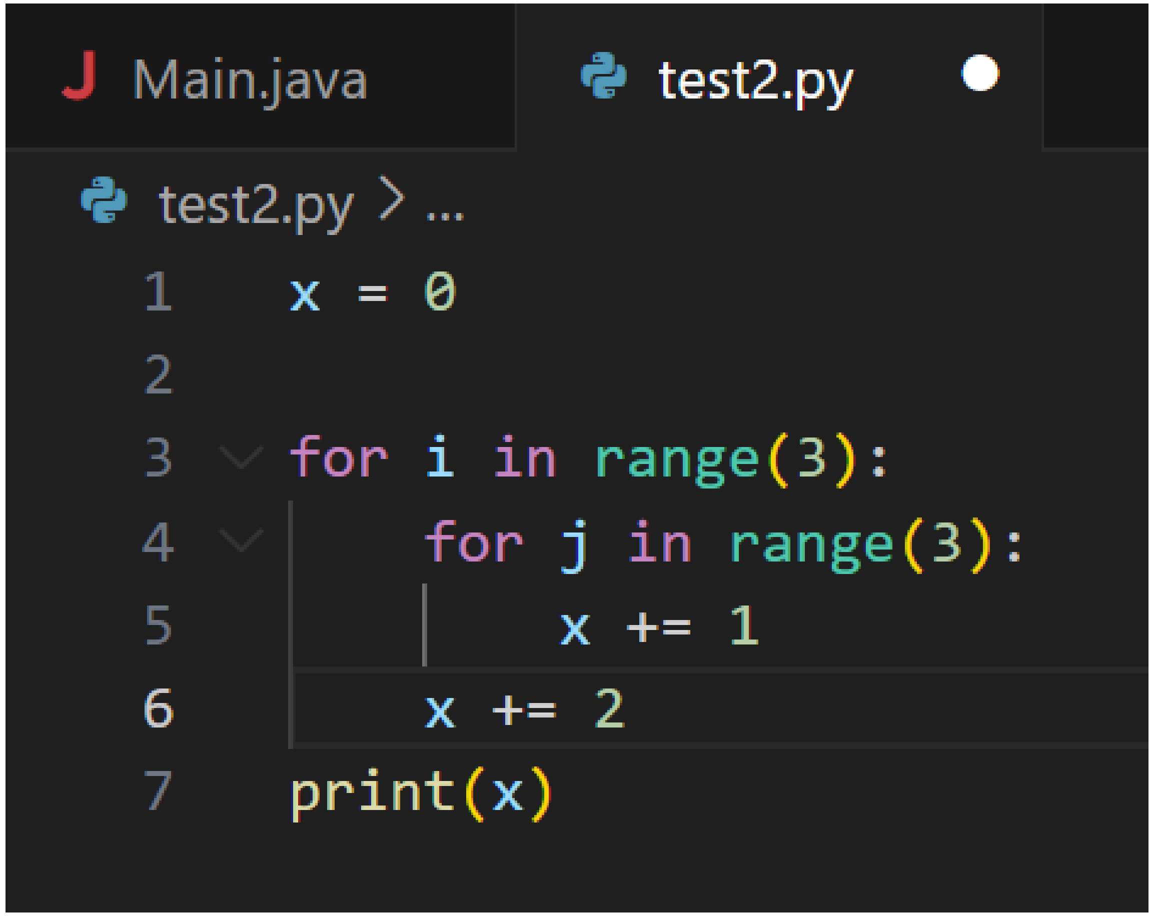
```java
public class Main {
public static void main(String[] args) {
int x = 0;

for (int i = 0 ; i < 3 ; i++) {
for (int j = 0 ; j < 3 ; j++) {
x += 1;
x += 2;
}
}
System.out.print(x);
}
}

```

These two do not…


Main.java test2.py

test2.py > …

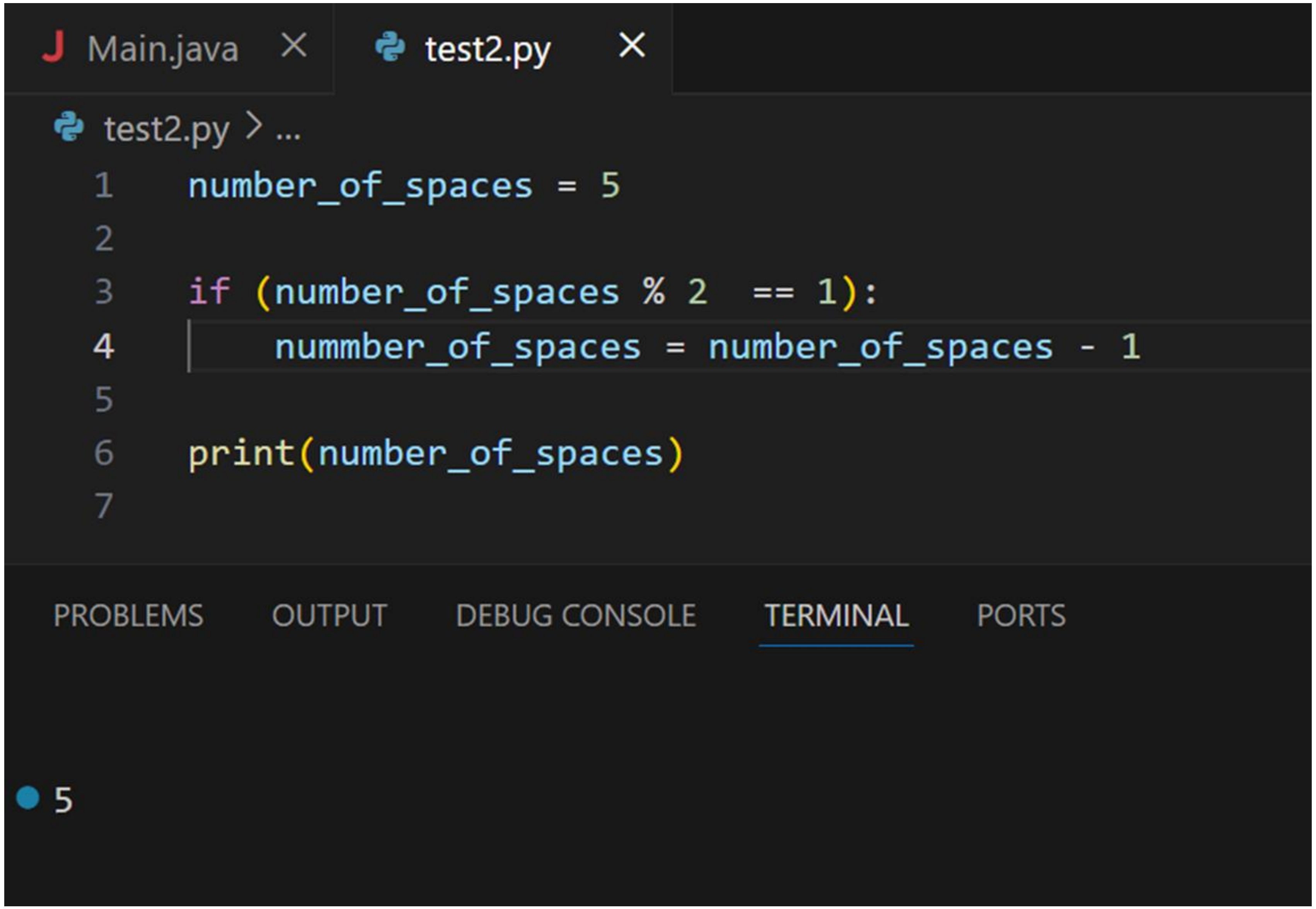
```
x = 0

for i in range(3):
    for j in range(3):
        x += 1
        x += 2
print(x)
```


Main.java test2.py

test2.py > …

```
x = 0

for i in range(3):
    for j in range(3):
        x += 1
    x += 2
print(x)
```

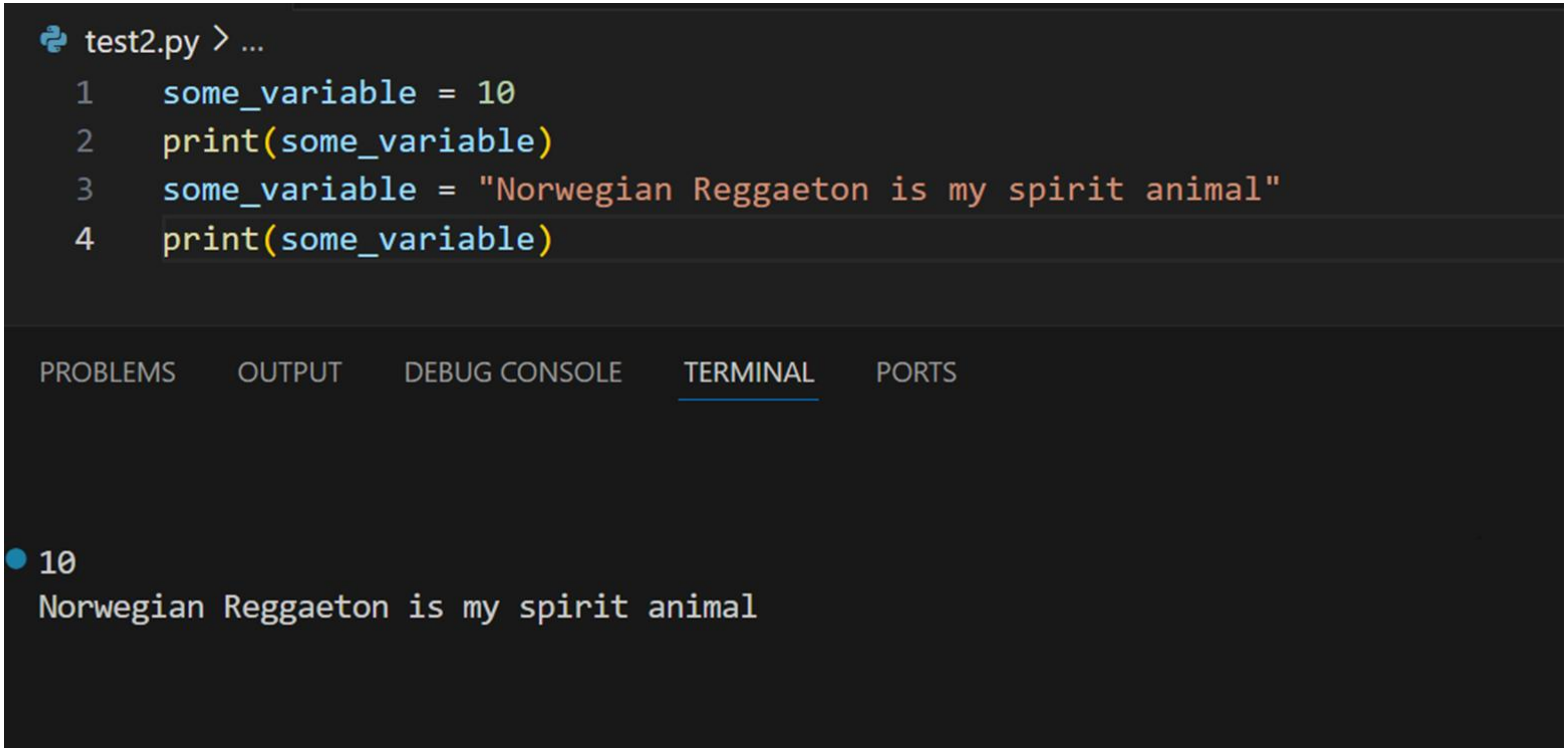
What's happening here?



```python
number_of_spaces = 5

if (number_of_spaces % 2  == 1):
    nummber_of_spaces = number_of_spaces - 1

print(number_of_spaces)

```

```
5
```

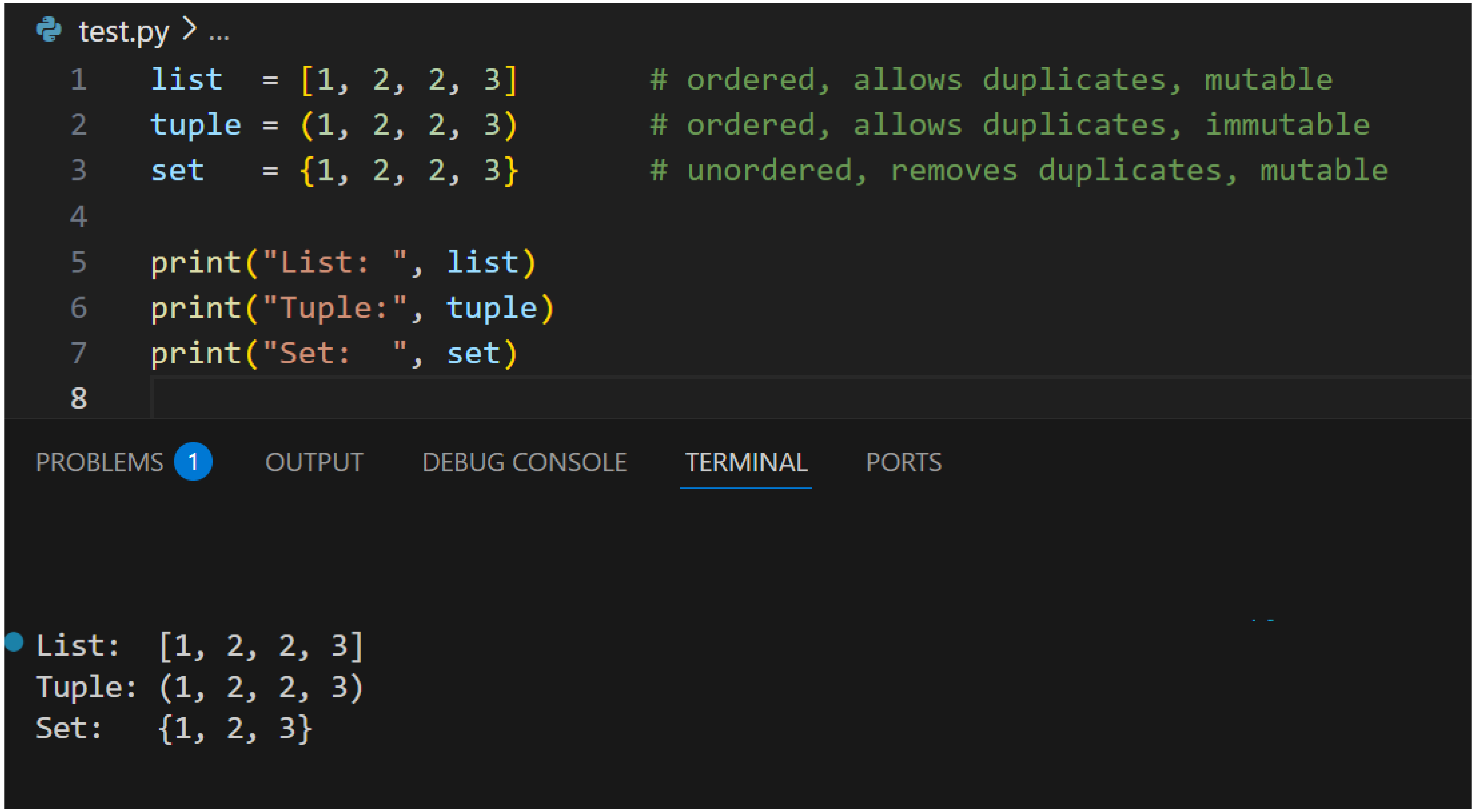
Dynamically typed variables



```python
some_variable = 10
print(some_variable)
some_variable = "Norwegian Reggaeton is my spirit animal"
print(some_variable)
```

```
10
Norwegian Reggaeton is my spirit animal
```

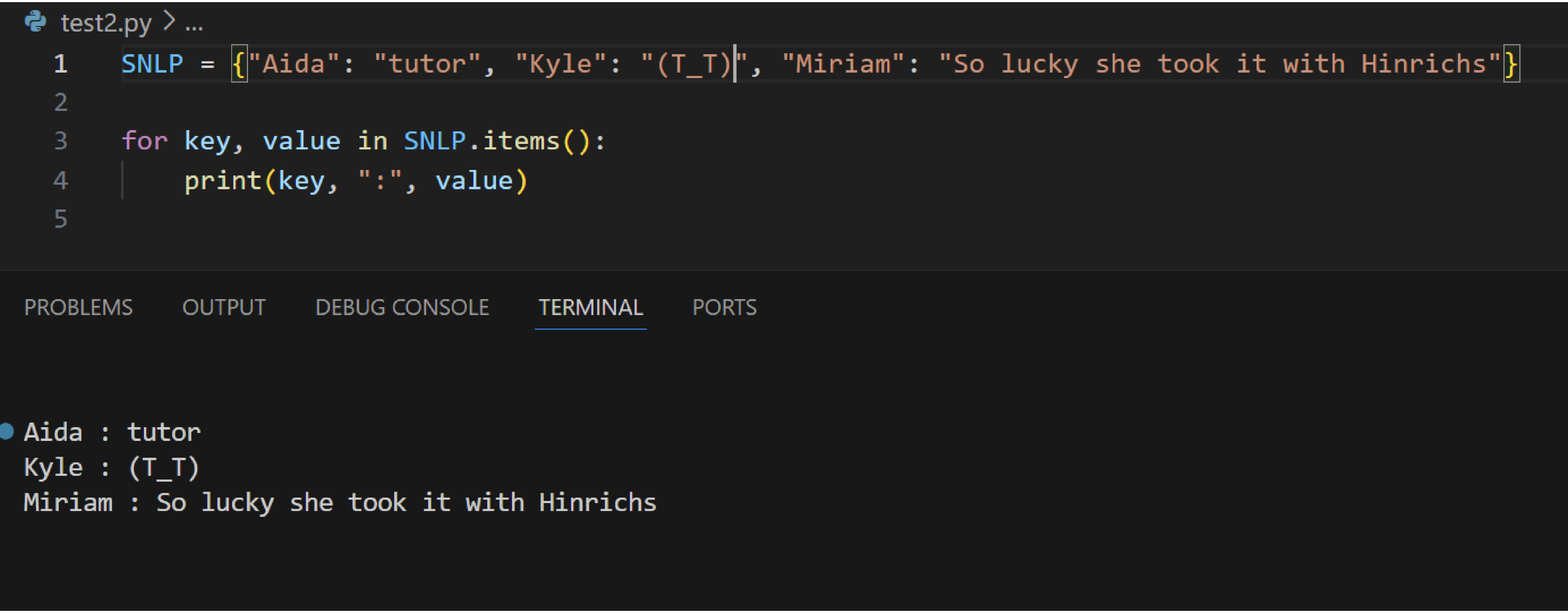
Lists, tuples, and sets



```python
list  = [1, 2, 2, 3]       # ordered, allows duplicates, mutable
tuple = (1, 2, 2, 3)       # ordered, allows duplicates, immutable
set   = {1, 2, 2, 3}       # unordered, removes duplicates, mutable

print("List: ", list)
print("Tuple:", tuple)
print("Set:  ", set)

```

```
List:  [1, 2, 2, 3]
Tuple: (1, 2, 2, 3)
Set:   {1, 2, 3}
```

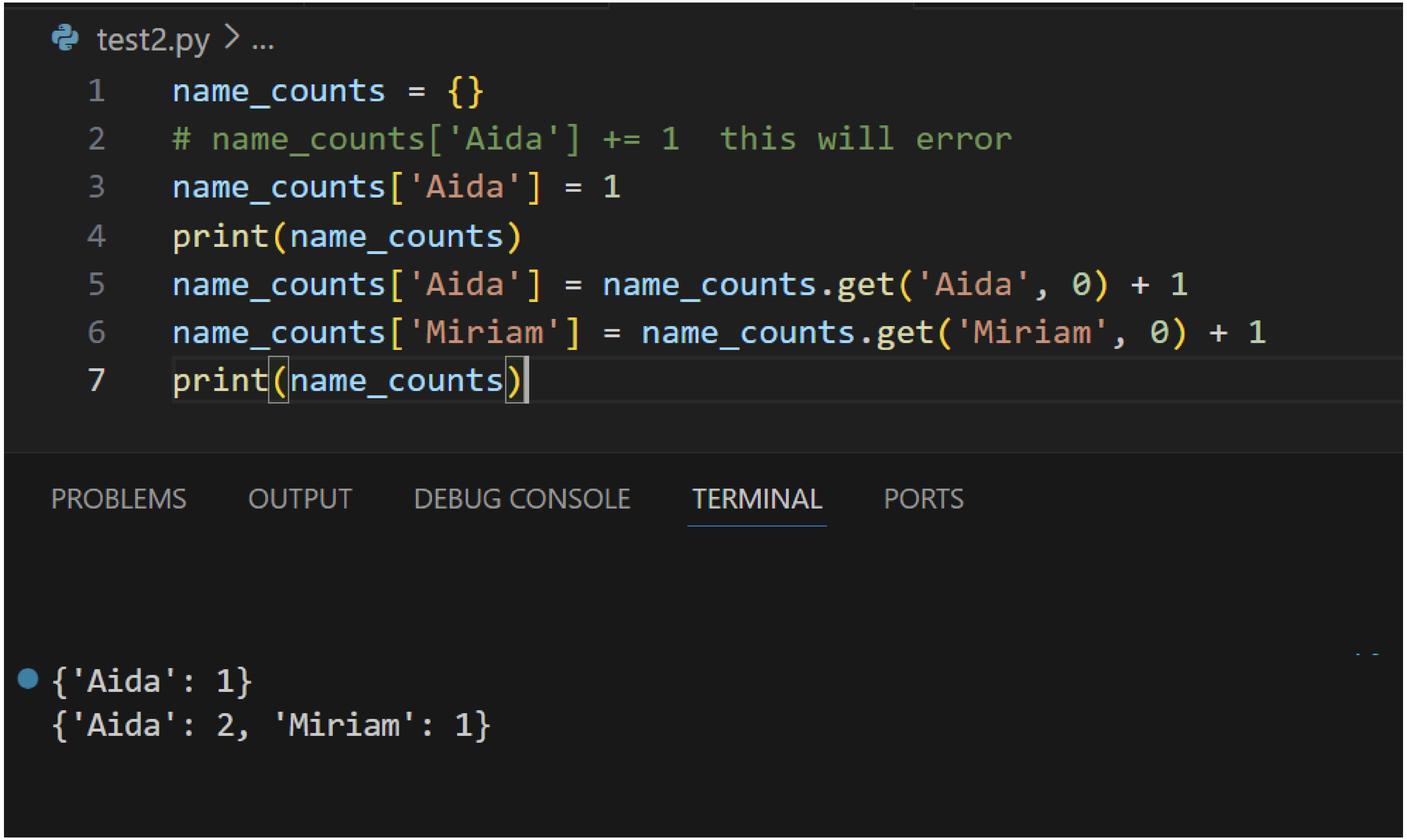
And one more...



```python
SNLP = {"Aida": "tutor", "Kyle": "(T_T)", "Miriam": "So lucky she took it with Hinrichs"}

for key, value in SNLP.items():
    print(key, ":", value)

```

```
Aida : tutor
Kyle : (T_T)
Miriam : So lucky she took it with Hinrichs
```

And one more…



```python
name_counts = {}
# name_counts['Aida'] += 1  this will error
name_counts['Aida'] = 1
print(name_counts)
name_counts['Aida'] = name_counts.get('Aida', 0) + 1
name_counts['Miriam'] = name_counts.get('Miriam', 0) + 1
print(name_counts)
```

```
{'Aida': 1}
{'Aida': 2, 'Miriam': 1}
```

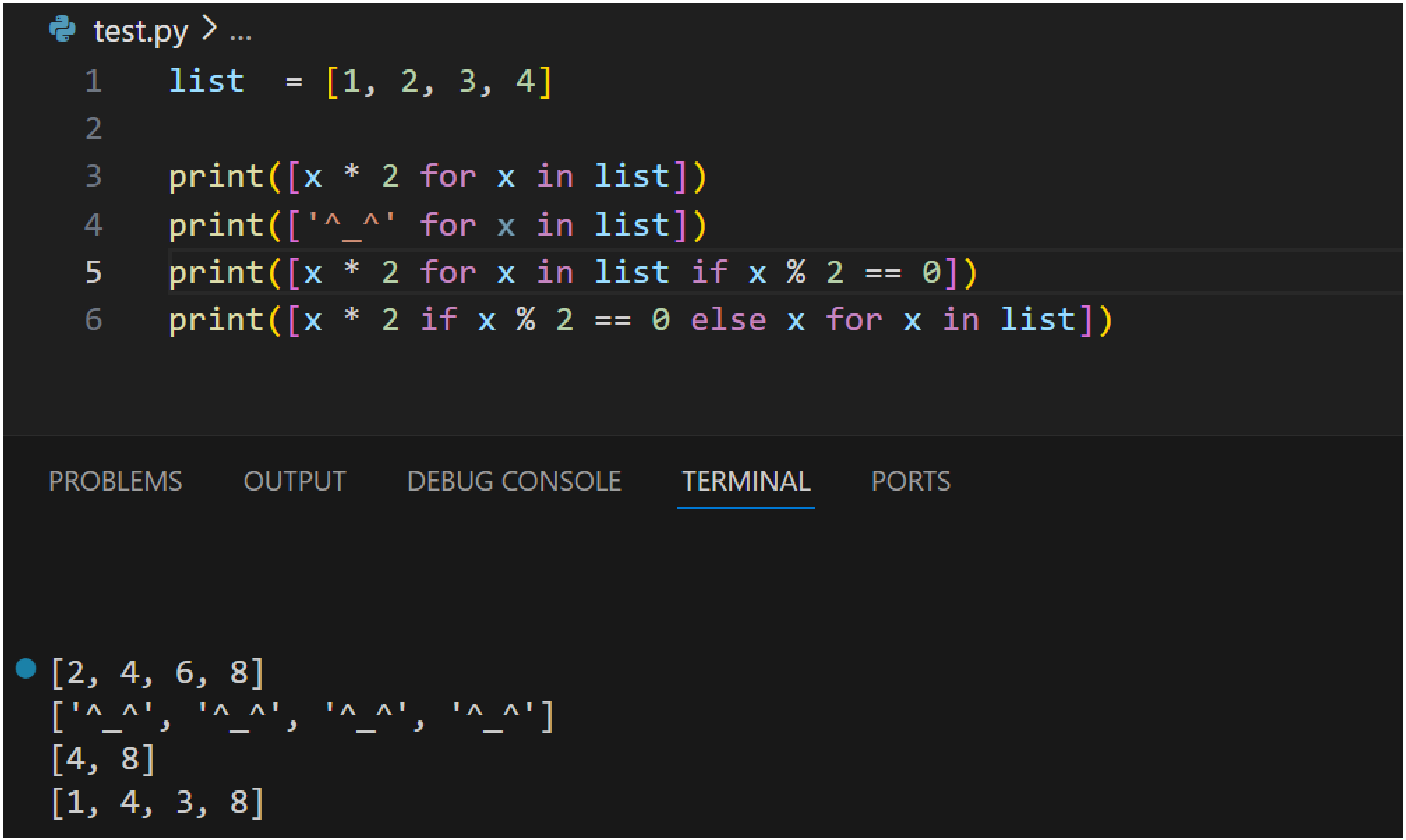
Comprehensions

List comprehension:

[expression for item in iterable (if condition)]

Dictionary comprehension:

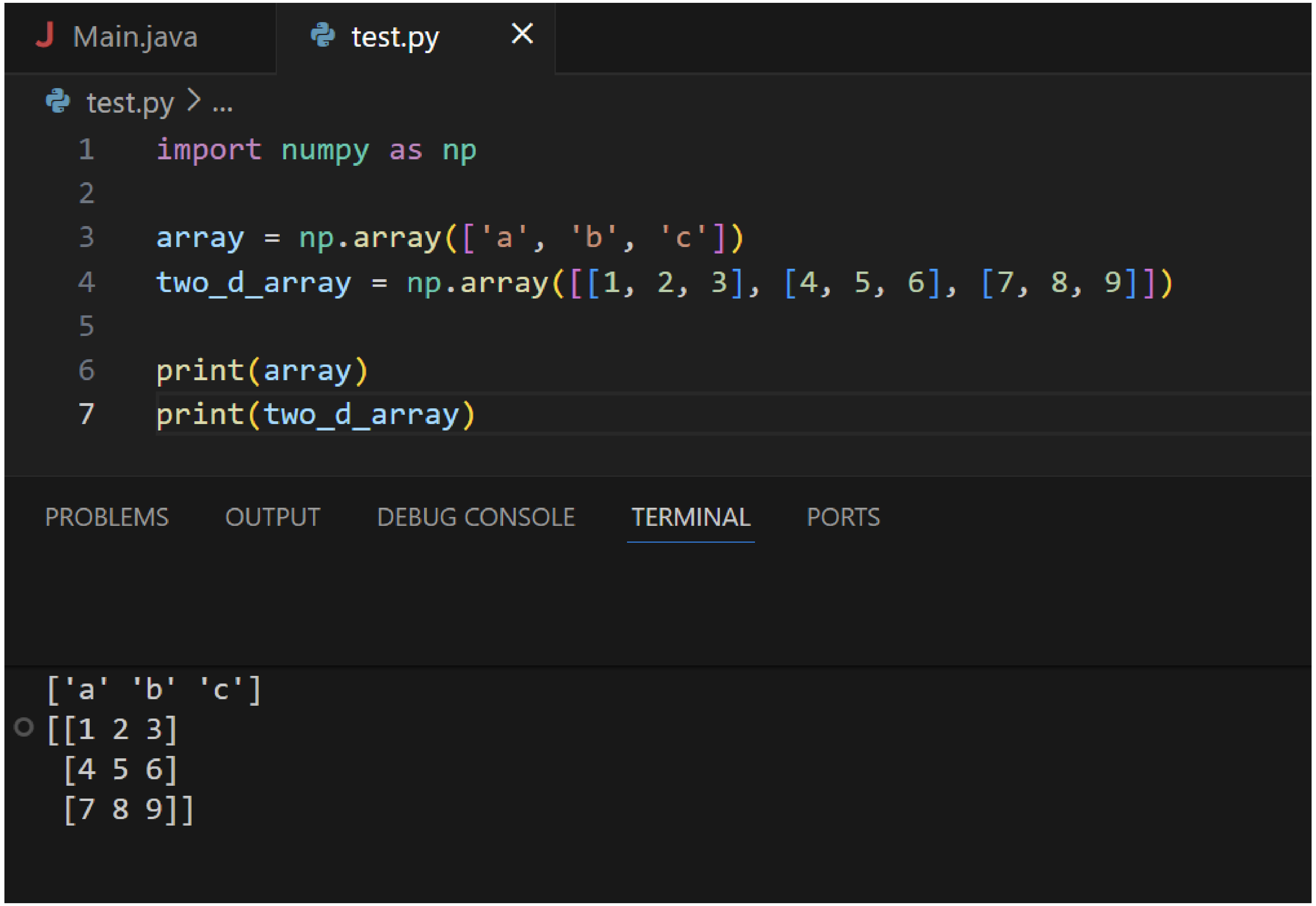
{k: v for item in iterable (if condition)}



```python
list  = [1, 2, 3, 4]

print([x * 2 for x in list])
print(['^_^' for x in list])
print([x * 2 for x in list if x % 2 == 0])
print([x * 2 if x % 2 == 0 else x for x in list])
```

```
[2, 4, 6, 8]
['^_^', '^_^', '^_^', '^_^']
[4, 8]
[1, 4, 3, 8]
```

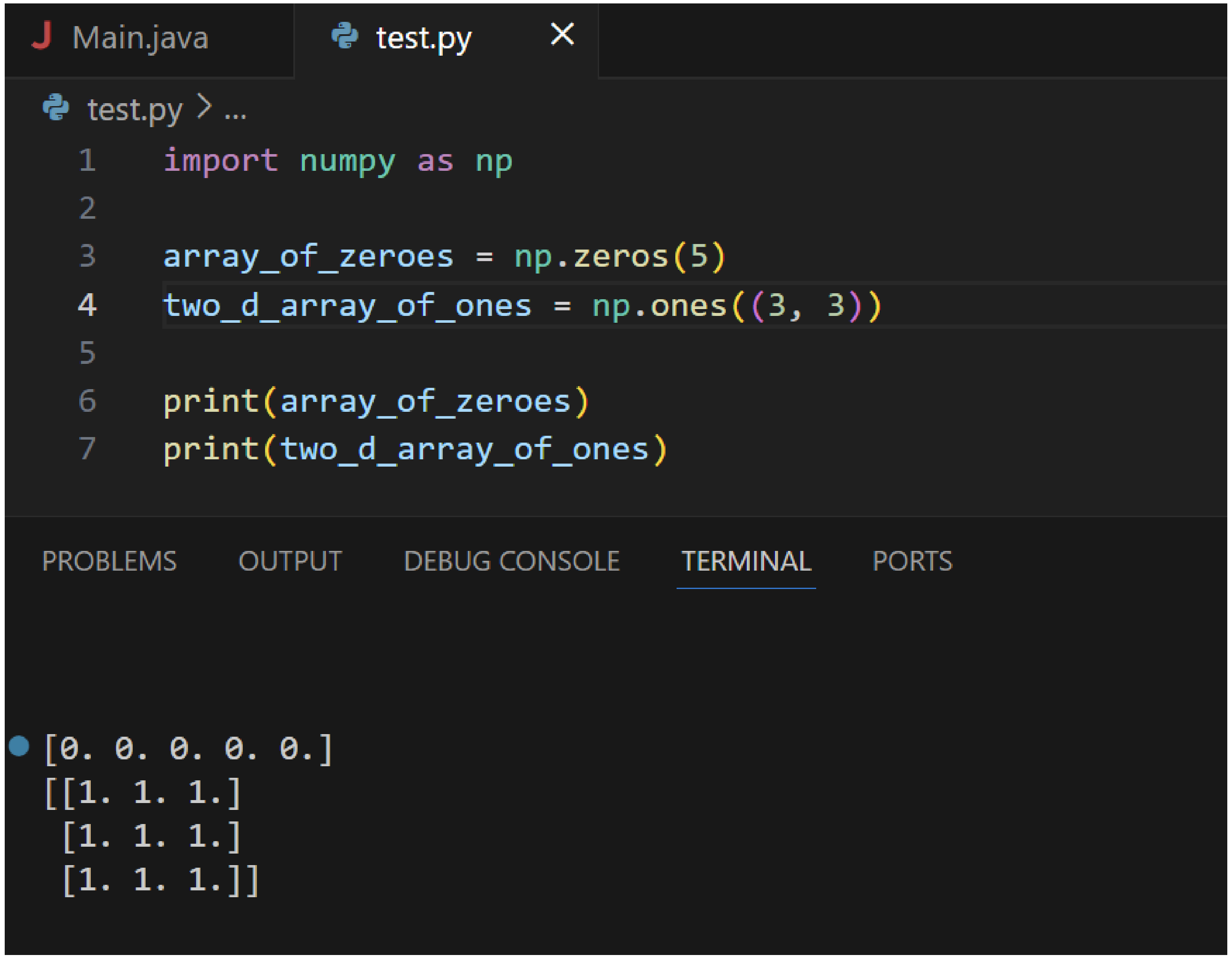
Numpy



```python
import numpy as np

array = np.array(['a', 'b', 'c'])
two_d_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print(array)
print(two_d_array)
```

```
['a' 'b' 'c']
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

More Numpy



```python
import numpy as np

array_of_zeroes = np.zeros(5)
two_d_array_of_ones = np.ones((3, 3))

print(array_of_zeroes)
print(two_d_array_of_ones)
```

```
[0. 0. 0. 0. 0.]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

Even more Numpy?

- Go RTFM:
- https://numpy.org/doc/stable/user/absolute_beginners.html