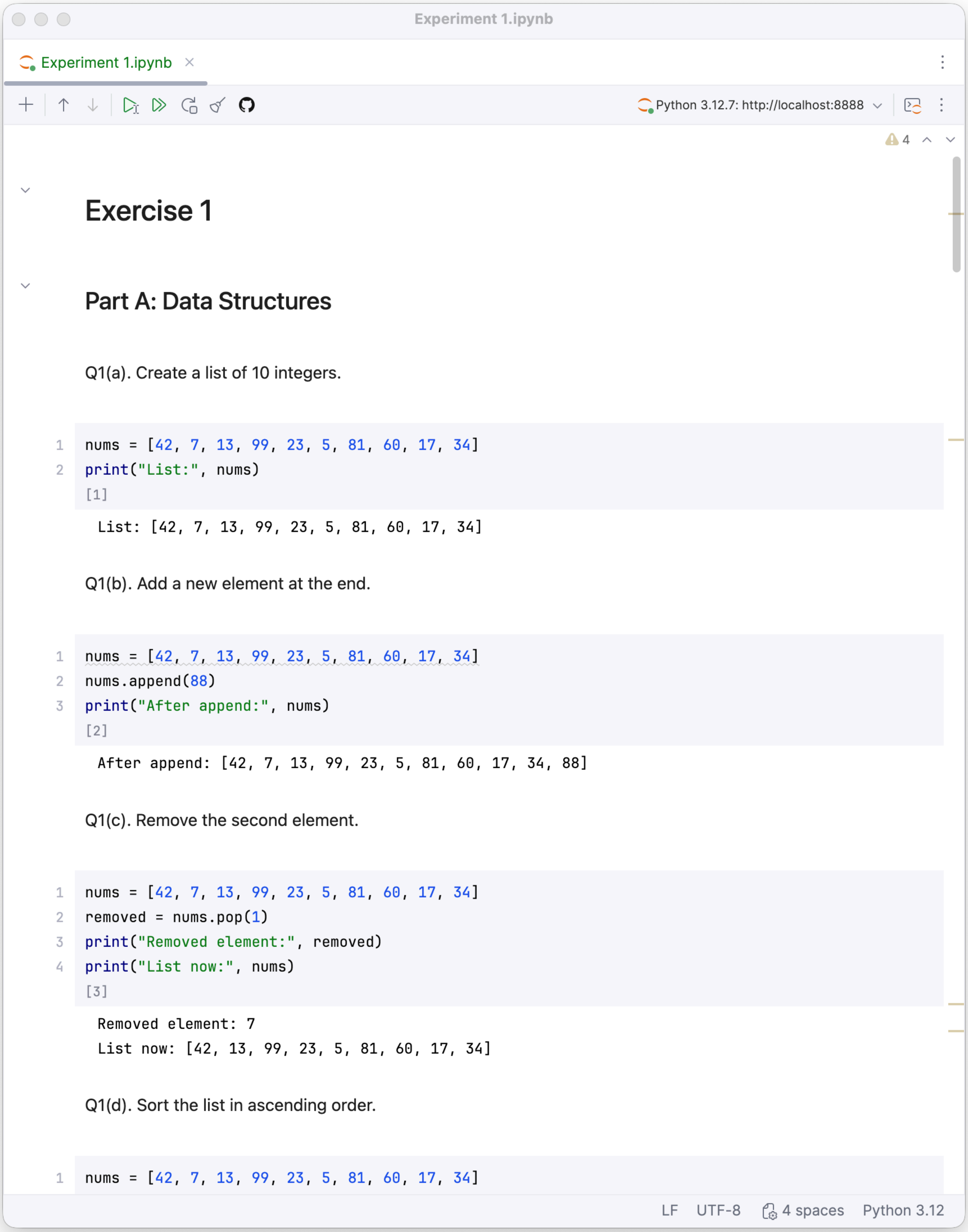

Experiment 1.ipynb
Experiment 1.ipynb
Python 3.12.7: http://localhost:8888
4
LF UTF-8 4 spaces Python 3.12

Exercise 1

Part A: Data Structures

Q1(a). Create a list of 10 integers.

```python
nums = [42, 7, 13, 99, 23, 5, 81, 60, 17, 34]
print("List:", nums)
```
[1]

```
List: [42, 7, 13, 99, 23, 5, 81, 60, 17, 34]
```

Q1(b). Add a new element at the end.

```python
nums = [42, 7, 13, 99, 23, 5, 81, 60, 17, 34]
nums.append(88)
print("After append:", nums)
```
[2]

```
After append: [42, 7, 13, 99, 23, 5, 81, 60, 17, 34, 88]
```

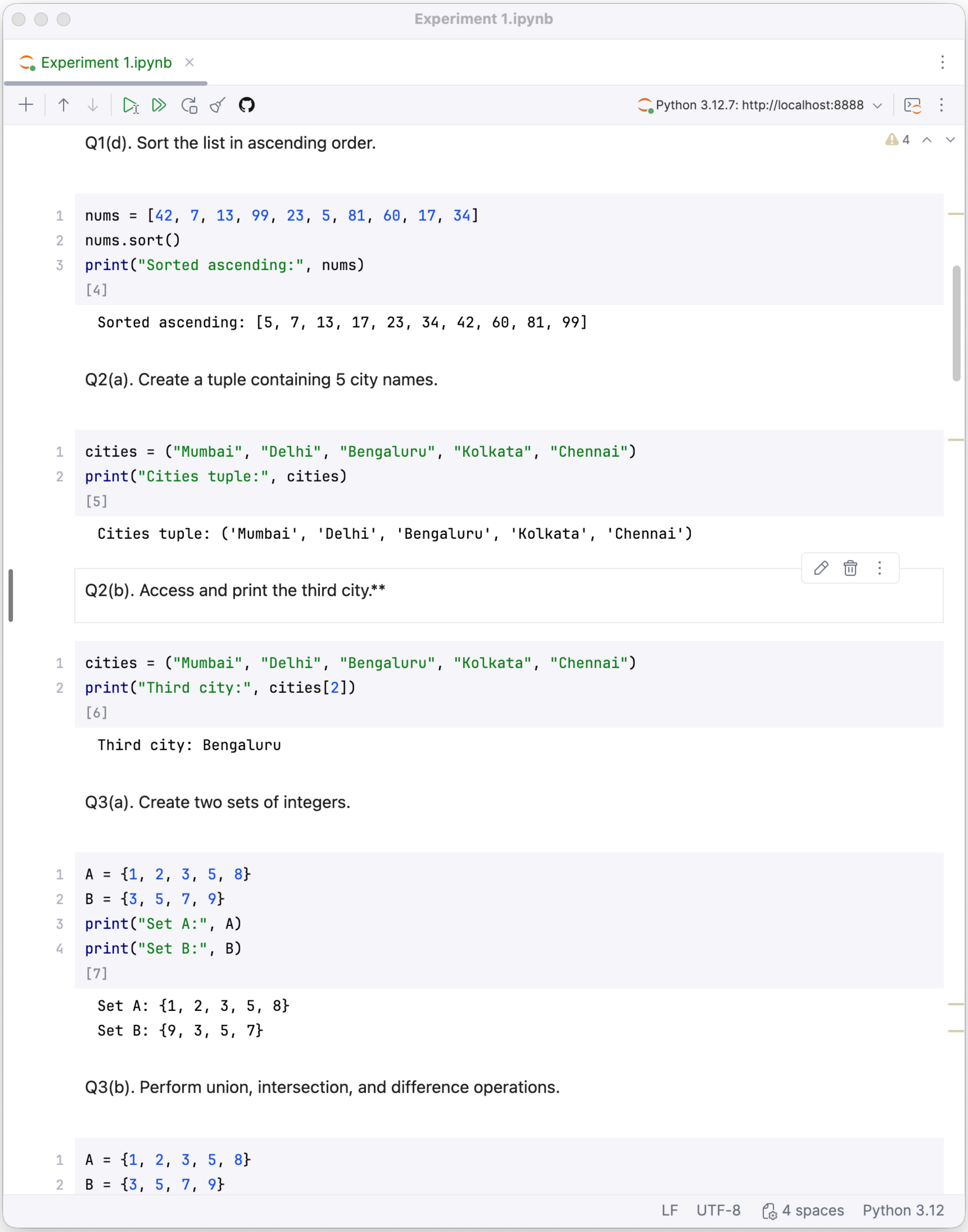
Q1(c). Remove the second element.

```python
nums = [42, 7, 13, 99, 23, 5, 81, 60, 17, 34]
removed = nums.pop(1)
print("Removed element:", removed)
print("List now:", nums)
```
[3]

```
Removed element: 7
List now: [42, 13, 99, 23, 5, 81, 60, 17, 34]
```

Q1(d). Sort the list in ascending order.

```python
nums = [42, 7, 13, 99, 23, 5, 81, 60, 17, 34]
```


Experiment 1.ipynb
Experiment 1.ipynb
Python 3.12.7: http://localhost:8888
4
LF UTF-8 4 spaces Python 3.12

Q1(d). Sort the list in ascending order.

```python
nums = [42, 7, 13, 99, 23, 5, 81, 60, 17, 34]
nums.sort()
print("Sorted ascending:", nums)
```
[4]

```
Sorted ascending: [5, 7, 13, 17, 23, 34, 42, 60, 81, 99]
```

Q2(a). Create a tuple containing 5 city names.

```python
cities = ("Mumbai", "Delhi", "Bengaluru", "Kolkata", "Chennai")
print("Cities tuple:", cities)
```
[5]

```
Cities tuple: ('Mumbai', 'Delhi', 'Bengaluru', 'Kolkata', 'Chennai')
```

Q2(b). Access and print the third city.**

```python
cities = ("Mumbai", "Delhi", "Bengaluru", "Kolkata", "Chennai")
print("Third city:", cities[2])
```
[6]

```
Third city: Bengaluru
```

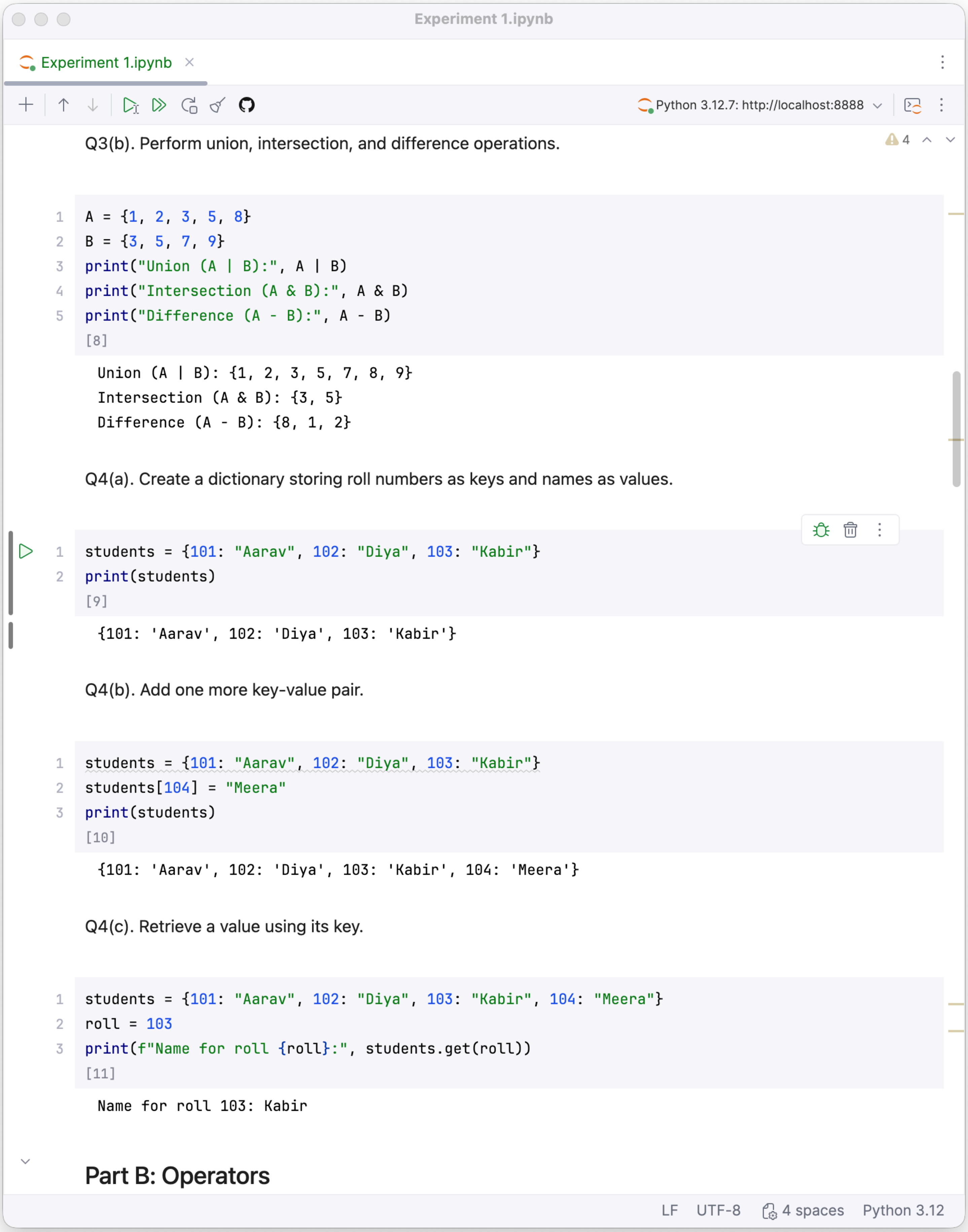
Q3(a). Create two sets of integers.

```python
A = {1, 2, 3, 5, 8}
B = {3, 5, 7, 9}
print("Set A:", A)
print("Set B:", B)
```
[7]

```
Set A: {1, 2, 3, 5, 8}
Set B: {9, 3, 5, 7}
```

Q3(b). Perform union, intersection, and difference operations.

```python
A = {1, 2, 3, 5, 8}
B = {3, 5, 7, 9}
```


Experiment 1.ipynb
Experiment 1.ipynb
Python 3.12.7: http://localhost:8888
LF UTF-8 4 spaces Python 3.12

Q3(b). Perform union, intersection, and difference operations.

```python
A = {1, 2, 3, 5, 8}
B = {3, 5, 7, 9}
print("Union (A | B):", A | B)
print("Intersection (A & B):", A & B)
print("Difference (A - B):", A - B)
```
[8]

```
Union (A | B): {1, 2, 3, 5, 7, 8, 9}
Intersection (A & B): {3, 5}
Difference (A - B): {8, 1, 2}
```

Q4(a). Create a dictionary storing roll numbers as keys and names as values.

```python
students = {101: "Aarav", 102: "Diya", 103: "Kabir"}
print(students)
```
[9]

```
{101: 'Aarav', 102: 'Diya', 103: 'Kabir'}
```

Q4(b). Add one more key-value pair.

```python
students = {101: "Aarav", 102: "Diya", 103: "Kabir"}
students[104] = "Meera"
print(students)
```
[10]

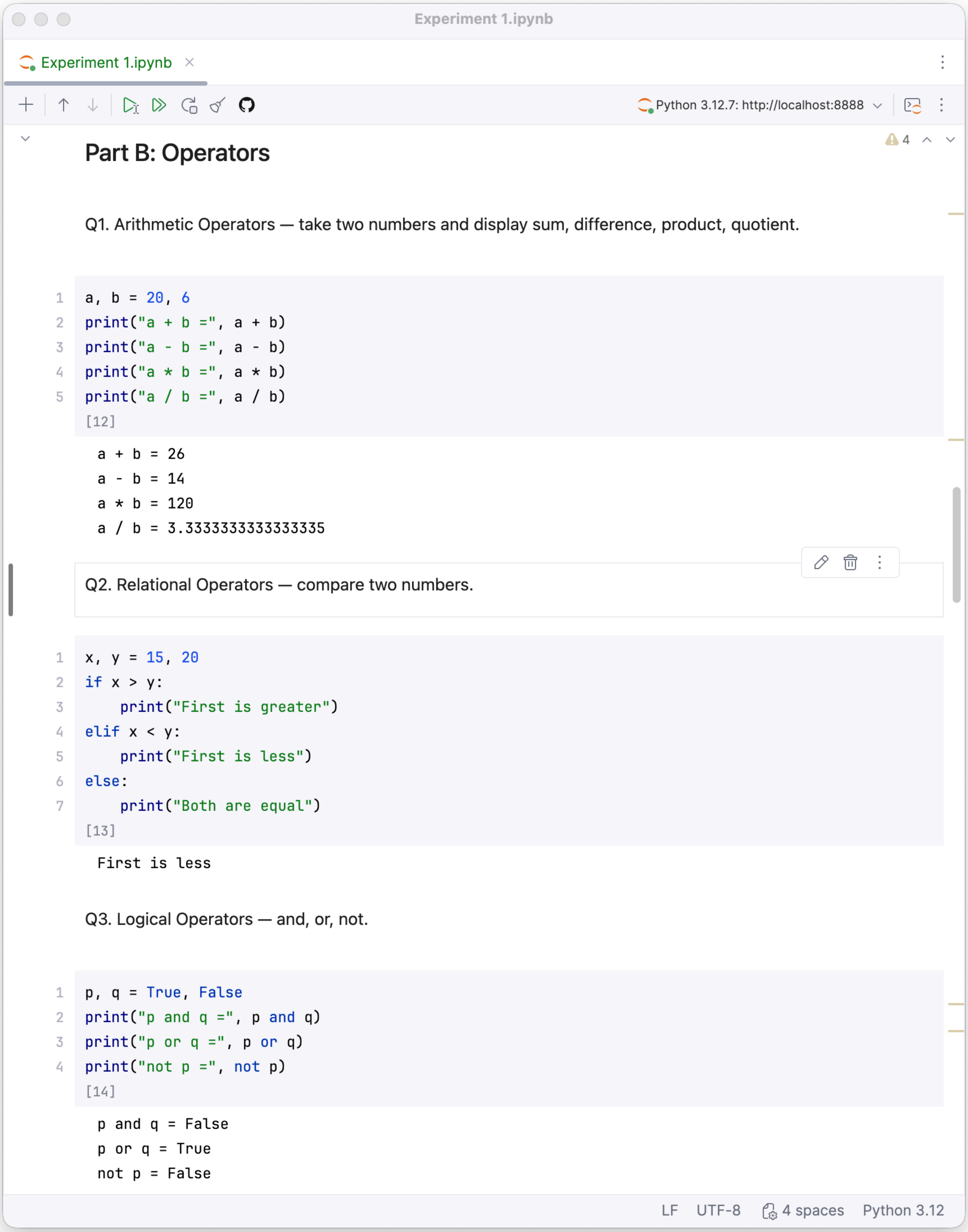
```
{101: 'Aarav', 102: 'Diya', 103: 'Kabir', 104: 'Meera'}
```

Q4(c). Retrieve a value using its key.

```python
students = {101: "Aarav", 102: "Diya", 103: "Kabir", 104: "Meera"}
roll = 103
print(f"Name for roll {roll}:", students.get(roll))
```
[11]

```
Name for roll 103: Kabir
```

Part B: Operators


Experiment 1.ipynb

Python 3.12.7: http://localhost:8888

Part B: Operators

Q1. Arithmetic Operators — take two numbers and display sum, difference, product, quotient.

```python
a, b = 20, 6
print("a + b =", a + b)
print("a - b =", a - b)
print("a * b =", a * b)
print("a / b =", a / b)
```
[12]

```
a + b = 26
a - b = 14
a * b = 120
a / b = 3.3333333333333335
```

Q2. Relational Operators — compare two numbers.

```python
x, y = 15, 20
if x > y:
    print("First is greater")
elif x < y:
    print("First is less")
else:
    print("Both are equal")
```
[13]

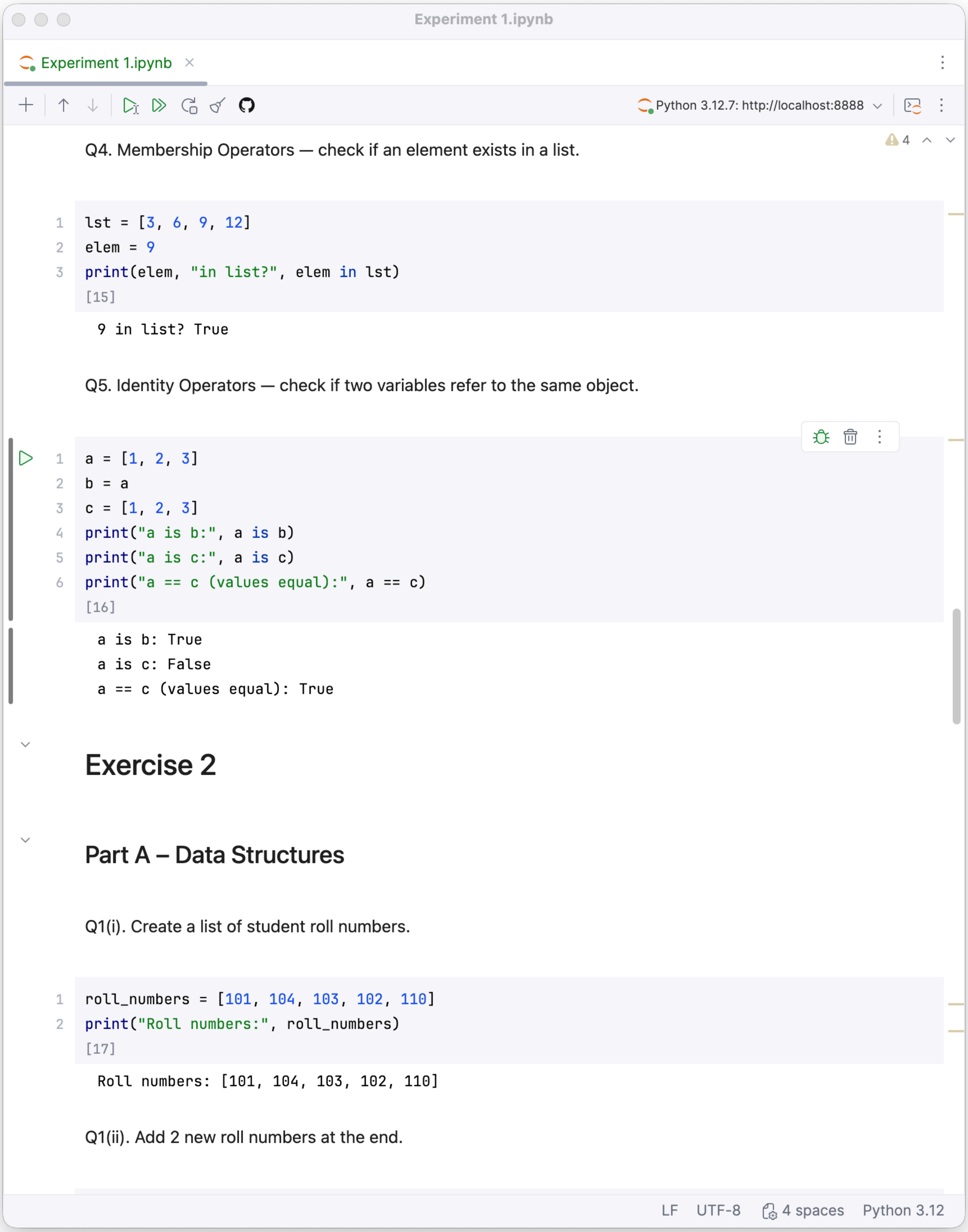
```
First is less
```

Q3. Logical Operators — and, or, not.

```python
p, q = True, False
print("p and q =", p and q)
print("p or q =", p or q)
print("not p =", not p)
```
[14]

```
p and q = False
p or q = True
not p = False
```

LF UTF-8 4 spaces Python 3.12


Experiment 1.ipynb

Python 3.12.7: http://localhost:8888

Q4. Membership Operators — check if an element exists in a list.

```python
lst = [3, 6, 9, 12]
elem = 9
print(elem, "in list?", elem in lst)
```
[15]

```
9 in list? True
```

Q5. Identity Operators — check if two variables refer to the same object.

```python
a = [1, 2, 3]
b = a
c = [1, 2, 3]
print("a is b:", a is b)
print("a is c:", a is c)
print("a == c (values equal):", a == c)
```
[16]

```
a is b: True
a is c: False
a == c (values equal): True
```

Exercise 2

Part A – Data Structures

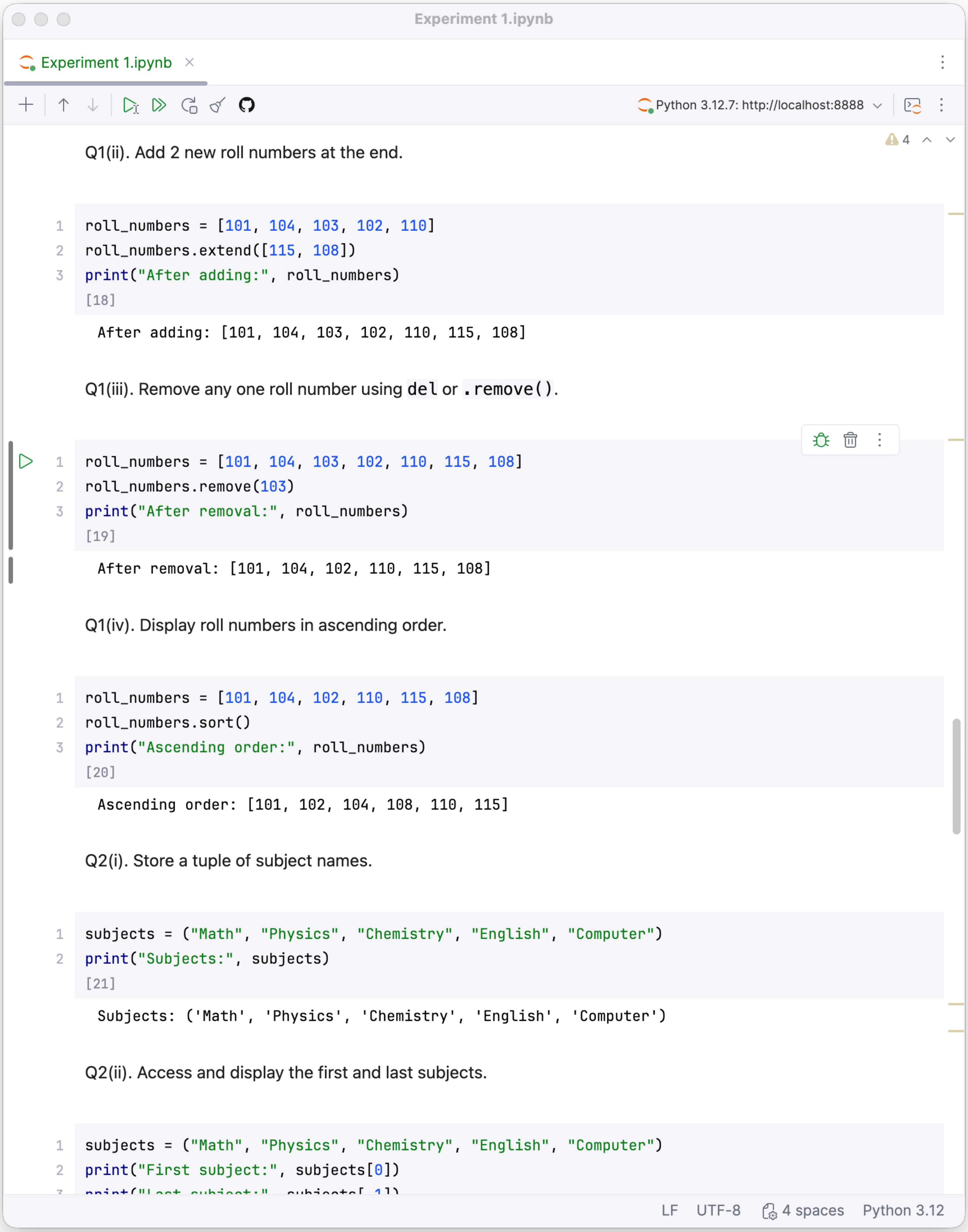
Q1(i). Create a list of student roll numbers.

```python
roll_numbers = [101, 104, 103, 102, 110]
print("Roll numbers:", roll_numbers)
```
[17]

```
Roll numbers: [101, 104, 103, 102, 110]
```

Q1(ii). Add 2 new roll numbers at the end.

LF UTF-8 4 spaces Python 3.12



Q1(ii). Add 2 new roll numbers at the end.

```
roll_numbers = [101, 104, 103, 102, 110]
roll_numbers.extend([115, 108])
print("After adding:", roll_numbers)
[18]
```

```
After adding: [101, 104, 103, 102, 110, 115, 108]
```

Q1(iii). Remove any one roll number using `del` or `.remove()`.

```
roll_numbers = [101, 104, 103, 102, 110, 115, 108]
roll_numbers.remove(103)
print("After removal:", roll_numbers)
[19]
```

```
After removal: [101, 104, 102, 110, 115, 108]
```

Q1(iv). Display roll numbers in ascending order.

```
roll_numbers = [101, 104, 102, 110, 115, 108]
roll_numbers.sort()
print("Ascending order:", roll_numbers)
[20]
```

```
Ascending order: [101, 102, 104, 108, 110, 115]
```

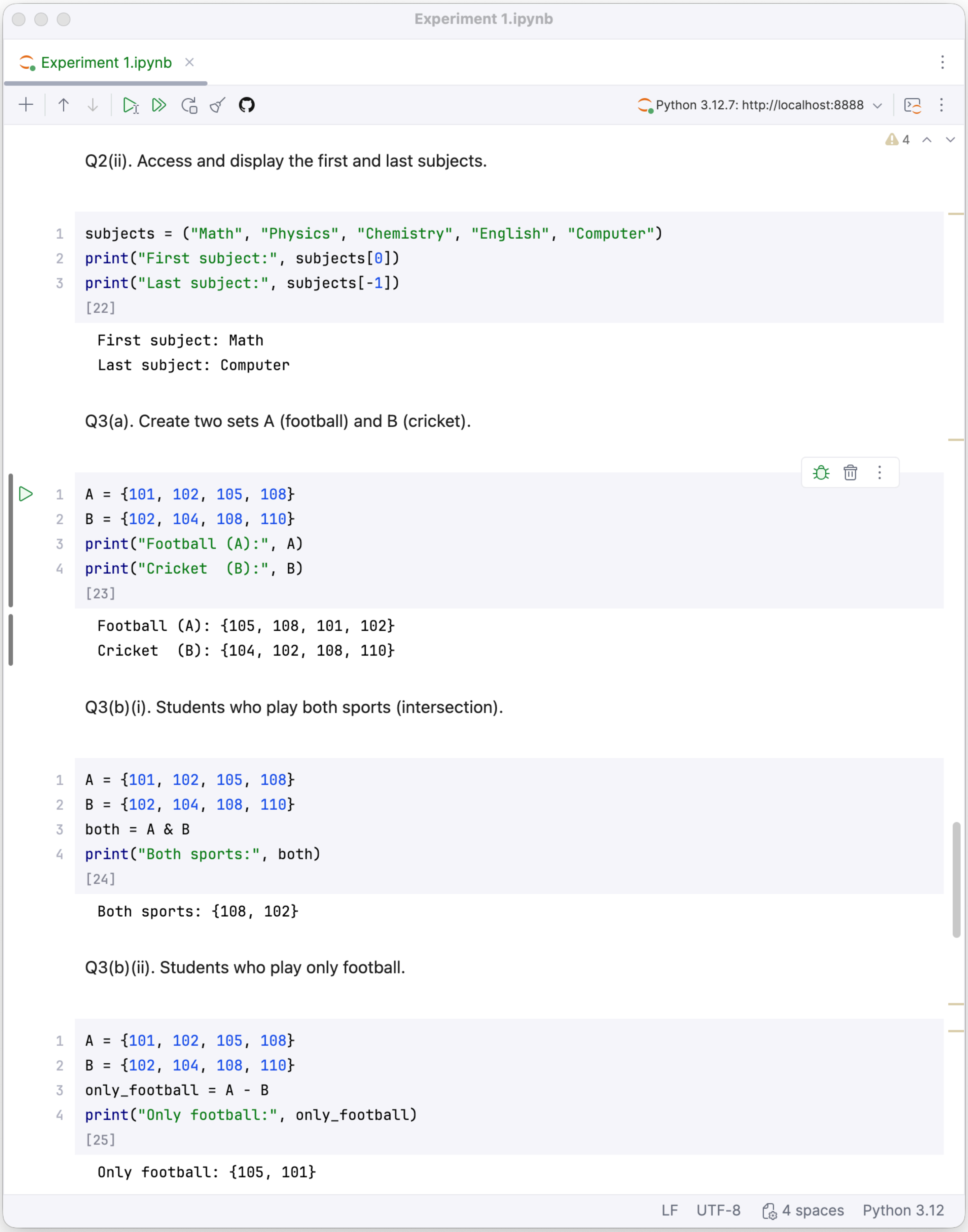
Q2(i). Store a tuple of subject names.

```
subjects = ("Math", "Physics", "Chemistry", "English", "Computer")
print("Subjects:", subjects)
[21]
```

```
Subjects: ('Math', 'Physics', 'Chemistry', 'English', 'Computer')
```

Q2(ii). Access and display the first and last subjects.

```
subjects = ("Math", "Physics", "Chemistry", "English", "Computer")
print("First subject:", subjects[0])
```


Experiment 1.ipynb
Experiment 1.ipynb
Python 3.12.7: http://localhost:8888
4
LF UTF-8 4 spaces Python 3.12

Q2(ii). Access and display the first and last subjects.

```python
subjects = ("Math", "Physics", "Chemistry", "English", "Computer")
print("First subject:", subjects[0])
print("Last subject:", subjects[-1])
```
[22]

```
First subject: Math
Last subject: Computer
```

Q3(a). Create two sets A (football) and B (cricket).

```python
A = {101, 102, 105, 108}
B = {102, 104, 108, 110}
print("Football (A):", A)
print("Cricket  (B):", B)
```
[23]

```
Football (A): {105, 108, 101, 102}
Cricket  (B): {104, 102, 108, 110}
```

Q3(b)(i). Students who play both sports (intersection).

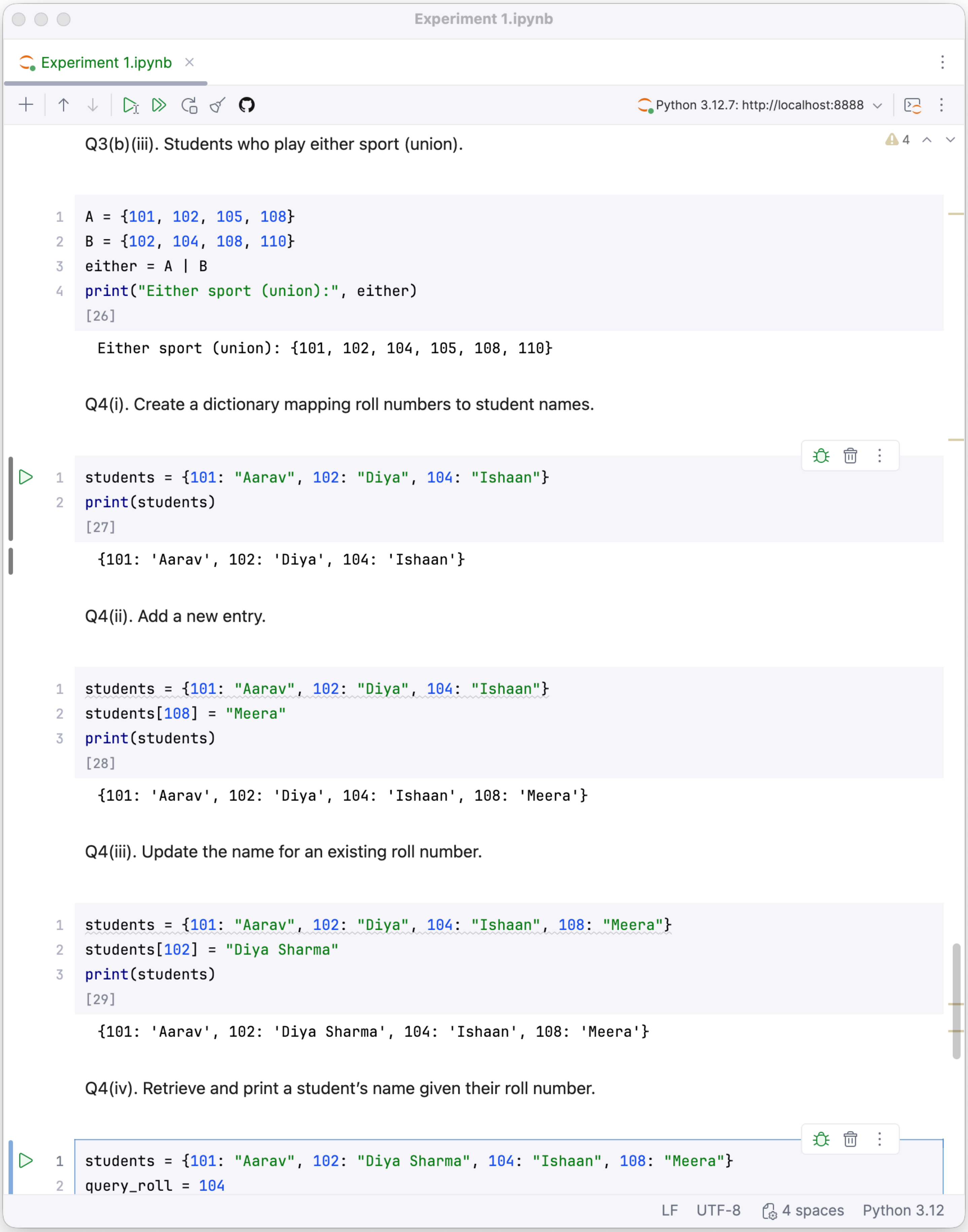
```python
A = {101, 102, 105, 108}
B = {102, 104, 108, 110}
both = A & B
print("Both sports:", both)
```
[24]

```
Both sports: {108, 102}
```

Q3(b)(ii). Students who play only football.

```python
A = {101, 102, 105, 108}
B = {102, 104, 108, 110}
only_football = A - B
print("Only football:", only_football)
```
[25]

```
Only football: {105, 101}
```



Q3(b)(iii). Students who play either sport (union).

```
A = {101, 102, 105, 108}
B = {102, 104, 108, 110}
either = A | B
print("Either sport (union):", either)
[26]
```

```
Either sport (union): {101, 102, 104, 105, 108, 110}
```

Q4(i). Create a dictionary mapping roll numbers to student names.

```
students = {101: "Aarav", 102: "Diya", 104: "Ishaan"}
print(students)
[27]
```

```
{101: 'Aarav', 102: 'Diya', 104: 'Ishaan'}
```

Q4(ii). Add a new entry.

```
students = {101: "Aarav", 102: "Diya", 104: "Ishaan"}
students[108] = "Meera"
print(students)
[28]
```

```
{101: 'Aarav', 102: 'Diya', 104: 'Ishaan', 108: 'Meera'}
```

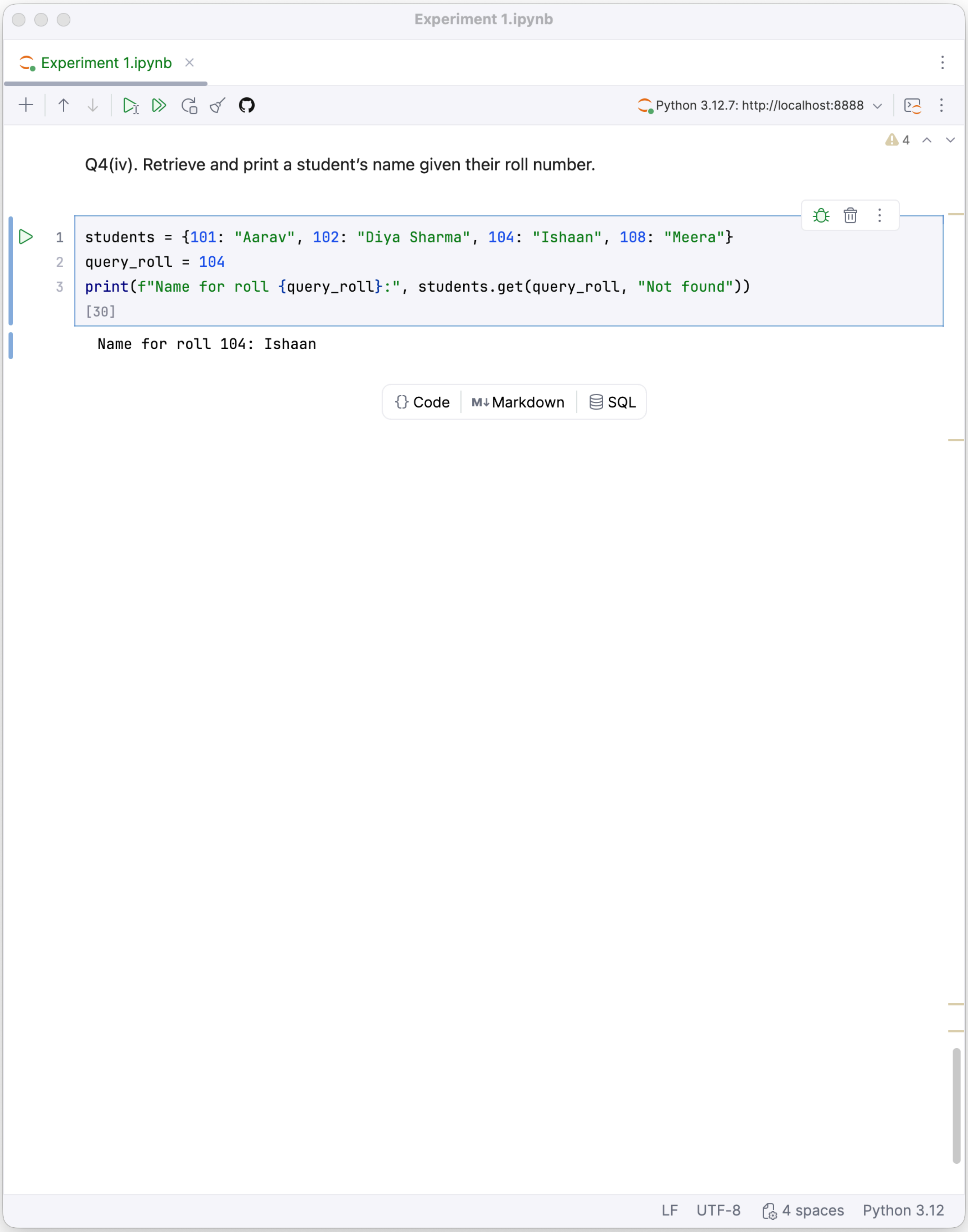
Q4(iii). Update the name for an existing roll number.

```
students = {101: "Aarav", 102: "Diya", 104: "Ishaan", 108: "Meera"}
students[102] = "Diya Sharma"
print(students)
[29]
```

```
{101: 'Aarav', 102: 'Diya Sharma', 104: 'Ishaan', 108: 'Meera'}
```

Q4(iv). Retrieve and print a student's name given their roll number.

```
students = {101: "Aarav", 102: "Diya Sharma", 104: "Ishaan", 108: "Meera"}
query_roll = 104
```



Q4(iv). Retrieve and print a student's name given their roll number.

```
students = {101: "Aarav", 102: "Diya Sharma", 104: "Ishaan", 108: "Meera"}
query_roll = 104
print(f"Name for roll {query_roll}:", students.get(query_roll, "Not found"))
[30]
```

```
Name for roll 104: Ishaan
```