Lab 6: Tuples and Sets in Python

Objectives

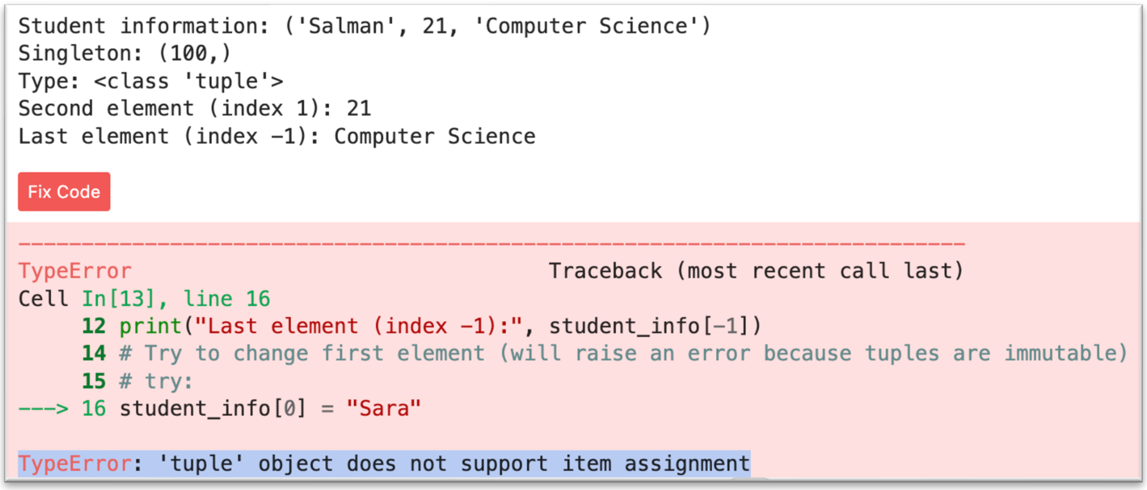
- Create and manipulate tuples and sets.
- Understand the differences between mutable and immutable data types.
- Perform set operations such as union, intersection, difference, and symmetric difference.
- Use tuple unpacking and return multiple values from functions.

Lab Tasks

Task 1: Tuples Basics

- Create a tuple named **`student_info`** with the following elements: **`"Salman", 21, "Computer Science"`**
- Create a singleton tuple with the value **`100`**. Print its type to confirm it's a tuple.
- Access the second element of **`student_info`** using positive indexing and the last element using negative indexing.
- Try to change the first element of **`student_info`** to "**`Sara`**" and observe the error message.

Example output:



```
Student information: ('Salman', 21, 'Computer Science')
Singleton: (100,)
Type: <class 'tuple'>
Second element (index 1): 21
Last element (index -1): Computer Science
```

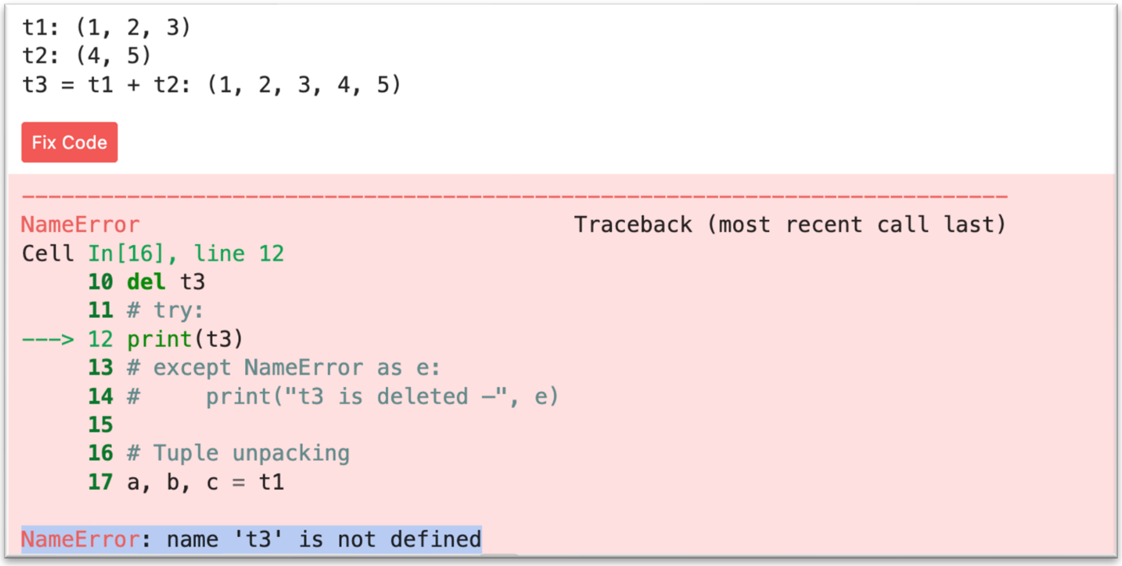
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 16
     12 print("Last element (index -1):", student_info[-1])
     14 # Try to change first element (will raise an error because tuples are immutable)
     15 # try:
---> 16 student_info[0] = "Sara"

TypeError: 'tuple' object does not support item assignment
```

Task 2: Tuple Operations

- Create two tuples: **`t1 = (1, 2, 3)`** and **`t2 = (4, 5).`** Concatenate them into a new tuple **`t3`**.
- **Delete** the tuple **`t3`** using the **`del`** keyword and try printing it afterward.
- Use <u>tuple unpacking to assign values</u> from **`t1`** to variables **`a, b,`** and **`c`**. Print each variable.
- Write a function **`get_student()`** that <u>returns a tuple with</u> **`name`**, **`age`**, and **`major`**. <u>Call the function</u> and unpack the result.

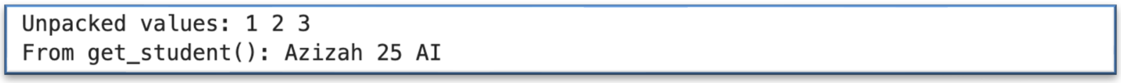
Example output:



```
t1: (1, 2, 3)
t2: (4, 5)
t3 = t1 + t2: (1, 2, 3, 4, 5)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[16], line 12
     10 del t3
     11 # try:
---> 12 print(t3)
     13 # except NameError as e:
     14 #     print("t3 is deleted -", e)
     15 
     16 # Tuple unpacking
     17 a, b, c = t1

NameError: name 't3' is not defined
```

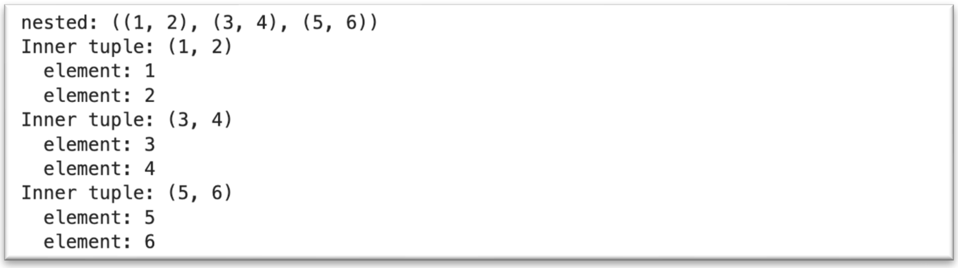


```
Unpacked values: 1 2 3
From get_student(): Azizah 25 AI
```

Task 3: Nested Tuples

- Create a nested tuple: **`nested = ((1, 2), (3, 4), (5, 6))`**
- Use a **`for`** loop to **<u>print each inner tuple</u>** and **<u>its elements</u>** individually.

Example output:



```
nested: ((1, 2), (3, 4), (5, 6))
Inner tuple: (1, 2)
  element: 1
  element: 2
Inner tuple: (3, 4)
  element: 3
  element: 4
Inner tuple: (5, 6)
  element: 5
  element: 6
```

Task 4: Sets Basics

- Create a set **`numbers = {1, 2, 2, 3, 4, 4}`** and print it. Explain the output.
- Convert the list **[1, 2, 2, 3]** and the tuple **(4, 5, 5, 6)** into **sets** using the **`set()`** function.

Example output:

```
numbers set: {1, 2, 3, 4}
Explanation: duplicates are removed; sets keep unique elements only.
set([1,2,2,3]) -> {1, 2, 3}
set((4,5,5,6)) -> {4, 5, 6}
```

Task 5: Set Operations

- Create two sets: **`A = {1, 2, 3, 4}`** and **`B = {3, 4, 5, 6}`**
- Perform and **print the results of union** using **`|`** and `A.`**`union`**`(B)`
- Perform and **print the results of intersection** using **`&`** and `A.`**`intersection`**`(B)`
- Perform and **print the results of difference** using **`-`** and `A.`**`difference`**`(B)`
- Perform and **print the results of symmetric difference** using **`^`** and `A.`**`symmetric_difference`**`(B)`

Example output:

```
A: {1, 2, 3, 4}
B: {3, 4, 5, 6}
A | B: {1, 2, 3, 4, 5, 6}
A.union(B): {1, 2, 3, 4, 5, 6}
A & B: {3, 4}
A.intersection(B): {3, 4}
A - B: {1, 2}
A.difference(B): {1, 2}
A ^ B: {1, 2, 5, 6}
A.symmetric_difference(B): {1, 2, 5, 6}
```

Challenge Task: Student Data Organizer

Step 1: Create Student Records

- Create a list of tuples, where each tuple contains:

  - Student name (string)
  - Age (int)
  - Major (string)
- Example:

```
students = [
    ("Ali", 20, "CS"),
    ("Sara", 21, "Physics"),
    ("Omar", 22, "Engineering")
]
```

Step 2: Analyze Course Enrollment

- Assume the following course sets for each student:
  - `ali_courses = {"Math", "CS", "Physics"}`
  - `sara_courses = {"CS", "Physics", "Chemistry"}`
  - `omar_courses = {"Math", "CS", "Biology"}`

- Perform the following operations:
  - Create a **set of all unique courses** offered using set **`union`**.
  - Find the **common courses among all students** using set **`intersection`**.
  - Find the **courses only Ali is taking** (**`difference`** between Ali's set and others).
- Example output:

```
All unique courses: {'CS', 'Chemistry', 'Physics', 'Math', 'Biology'}
Common to everyone: {'CS'}
Only Ali takes: set()
```

Step 3: Tuple Assignment

- Use tuple unpacking to extract and **print each student's name and major**:

```
for student in students:
    name, _, major = student  #underscore to ignore age
    print(name, "-", major)
```

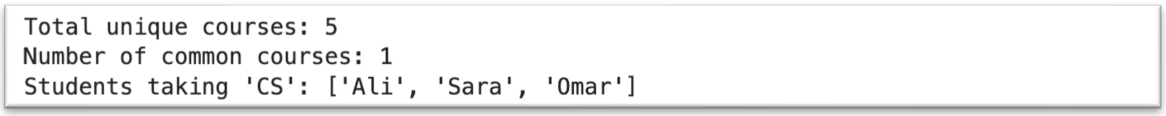
- Example output:

```
Ali - CS
Sara - Physics
Omar - Engineering
```

Step 4: Function Return

- Write a function **`get_course_stats()`** that <u>returns</u>:
  - Total number of unique courses. Using `|` and `len()`
  - Number of common courses. Using `&` and `len()`
  - A list of students taking 'CS'

- <u>Return these values as a tuple</u> and <u>unpack them when calling the function</u>:
  `total, common, cs_students =` **`get_course_stats()`**
- Example output:



```
Total unique courses: 5
Number of common courses: 1
Students taking 'CS': ['Ali', 'Sara', 'Omar']
```