## Code Snippet 1

**Error :**

There's a typo in the variable name. The code tries to print `number_of_apple`, but it should be `number_of_apples`.

**Corrected Code :**

```
number_of_apples = 5

print(number_of_apples)
```

Explanation :

The mistake is a simple typo. We defined `number_of_apples` with a value of 5, but when we tried to print it, we accidentally wrote `number_of_apple`, which doesn't exist. Fixing the typo to `number_of_apples` solves the problem.

## Code Snippet 2

**Error :**

The code is trying to access an element at index 3, but the list `fruits` only has indices 0, 1, and 2. This will cause an `IndexError`.

**Corrected Code :**

```
fruits = ["apple", "banana", "cherry"]

print(fruits[2])
```

Explanation :

The list `fruits` has three elements: "apple", "banana", and "cherry". These elements are at indices 0, 1, and 2 respectively. Trying to access `fruits[3]` is out of bounds since there is no fourth element in the list. The corrected code accesses the third element (index 2), which is "cherry".

## Debugging Exercise 3: Function Not Behaving As Expected

**Error :**
The list `[1, 2, 3, 4, 5, "6"]` contains a string `"6"`, which would cause a `TypeError` when attempting arithmetic operations.

**Corrected Code :**

```
def find_average(numbers):
    sum = 0
    for number in numbers:
        sum += number
    average = sum / len(numbers)
    return average
numbers = [1, 2, 3, 4, 5, 6]       # corrected part
average = find_average(numbers)
print(f"The average is: {average}")
```

Explanation :
The error in the original code was caused by including a string `"6"` in the list `numbers`, which cannot be processed numerically. This corrected version of the code uses a list of integers `[1, 2, 3, 4, 5, 6]` to calculate the average correctly. The `find_average` function iterates through each number in the list, computes the sum, and divides by the number of elements to determine the average. Finally, the average is printed in a clear message format.

## Exercise 4: Incorrect Dictionary Usage

**Error :**

The key `"Alice "` (with a trailing space) does not match the key `"Alice"` in the `student_records` dictionary, so a new entry is created instead of updating the existing one.

**Corrected Code :**

```
def update_record(records, name, score):
    if name in records:
        records[name].append(score)
    else:
        records[name] = [score]
```

```
student_records = {"Alice": [88.92], "Bob": [70, 85]}


update_record(student_records, "Charlie", 91)
update_record(student_records, "Alice", 95)  # Corrected key without trailing space
print(student_records)
```

**Explaination :**

The error in the original code occurs when attempting to update `"Alice "` (with a trailing space) in the `student_records` dictionary. Due to the space, the function `update_record` creates a new entry instead of updating the existing one for `"Alice"`. The corrected code adjusts the key to `"Alice"` (without the space), ensuring that the function correctly updates Alice's record with the new score of 95. The updated `student_records` dictionary is then printed to show the correct entries.