

Q1.

Code Snippet 1: Variable Name Typo

Code:

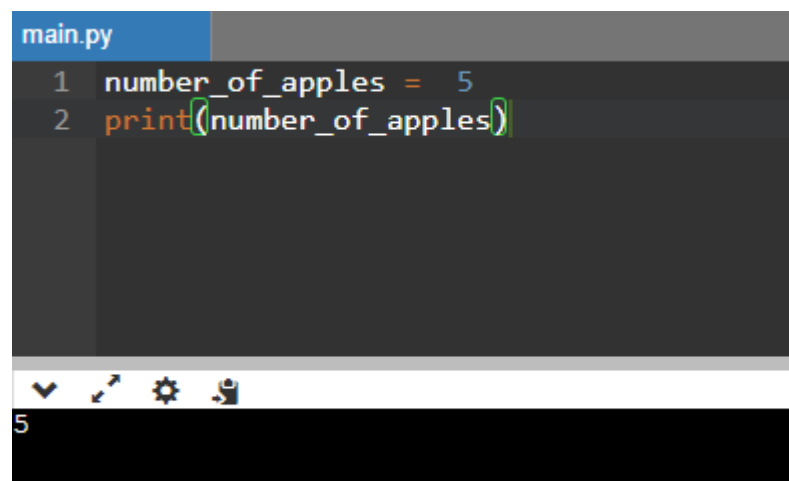
```
python

number_of_apples = 5
print(number_of_apple)
```

Error:

The error in the above code is NameError : name 'number\_of\_apple' is not defined

Corrected Code:

A screenshot of a code editor window titled 'main.py'. The editor contains two lines of Python code: '1 number\_of\_apples = 5' and '2 print(number\_of\_apples)'. The code is syntax-highlighted. Below the editor, there is a toolbar with icons for a dropdown menu, a cursor, a gear (settings), and a document. At the bottom, a black output console displays the number '5' in white text.

```
main.py

1 number_of_apples = 5
2 print(number_of_apples)

5
```

Error Explanation:

A NameError occurs in Python when a variable or name is used in code without being defined. This can happen if there's a typo in the variable name or if the variable hasn't been assigned a value before it's used.

In this code, number\_of\_apple is mistakenly used instead of number\_of\_apples, leading to a NameError because number\_of\_apple hasn't been defined. To fix this error, ensure that variable names are correctly spelled and defined before use.

Q2.

Code Snippet 2: Accessing List Elements Out of Range

Code:

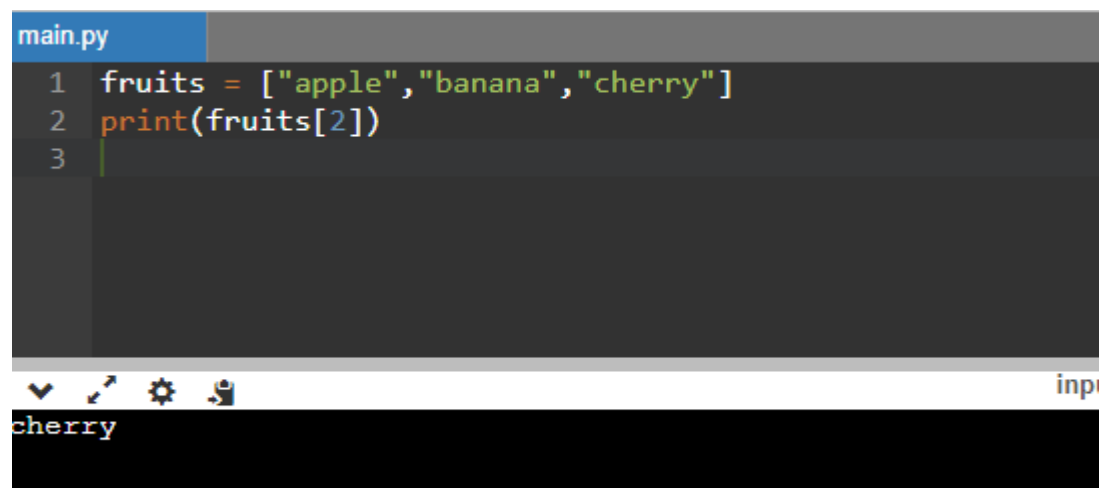
```
python

fruits = ["apple", "banana", "cherry"]
print(fruits[3])
```

Error:

The error in the above code is IndexError : list index out of range

Corrected code to print last element in the list:

A screenshot of a code editor window. The title bar at the top says 'main.py'. The editor contains three lines of Python code: '1 fruits = ["apple", "banana", "cherry"]', '2 print(fruits[2])', and '3'. Below the code editor, there is a toolbar with icons for undo, redo, settings, and a search icon. To the right of the toolbar, the text 'input' is partially visible. At the bottom of the window, a black output console shows the word 'cherry' in a light blue font.

Error Explanation:

In Python, list indices start from 0. This means that if we have a list with three elements, the indices for those elements would be 0, 1, and 2.

Even though the length of the list is 3, we access the last element using index 2 because indices start from 0 and go up to n-1, where n is the length of the list.

Q3.

### Debugging Exercise 3: Function Not Behaving as Expected

```
python

def find_average(numbers):
    sum = 0
    for number in numbers:
        sum += number
    average = sum / len(numbers)
    return average

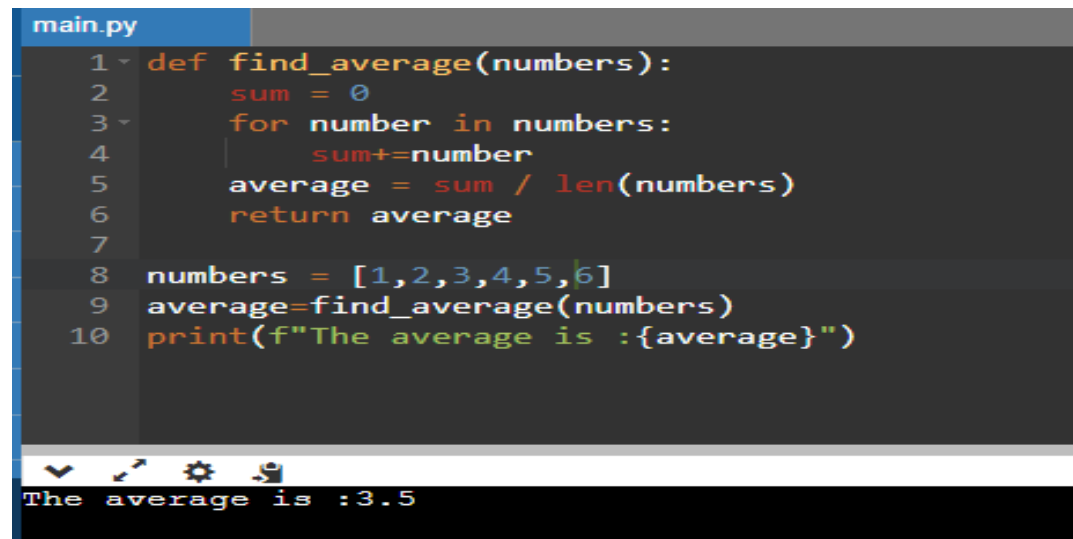
numbers = [1, 2, 3, 4, 5, "6"]
average = find_average(numbers)
print(f"The average is: {average}")
```

Error:

The error in the above code is

TypeError: unsupported operand type(s) for +=: 'int' and 'str'

Correct Code:

A screenshot of a code editor window titled 'main.py'. The code is as follows:

```
1 def find_average(numbers):
2     sum = 0
3     for number in numbers:
4         sum+=number
5     average = sum / len(numbers)
6     return average
7
8 numbers = [1,2,3,4,5,6]
9 average=find_average(numbers)
10 print(f"The average is :{average}")
```

The output at the bottom of the editor is 'The average is :3.5'.

```
main.py
1 def find_average(numbers):
2     sum = 0
3     for number in numbers:
4         sum+=number
5     average = sum / len(numbers)
6     return average
7
8 numbers = [1,2,3,4,5,6]
9 average=find_average(numbers)
10 print(f"The average is :{average}")

The average is :3.5
```

Explanation:

The error is a `TypeError`. This error occurs because the code attempts to add a string to an integer, which is not a valid operation in Python. The list `numbers` contains a mix of integers and a string `"6"`. When the `find_average` function iterates through this list and tries to add each element to the `sum`, it encounters `"6"` which is a string. Python does not allow adding a string to an integer using the `+=` operator, resulting in a `TypeError`. To fix this error, we need to ensure that all elements in the list are of the same data type (preferably numerical) before performing arithmetic operations on them.

Q4.

#### Exercise 4: Incorrect Dictionary Usage

Code:

```
python

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)

print(student_records)
```

Error:

The error you can expect is an 'AttributeError'. This error occurs because the code attempts to append a score to a value that is not a list (or iterable), causing an attribute-related issue.

Correct Code:

```
main.py
1 def update_record(records, name, score):
2     if name in records:
3         records[name].append(score)
4     else:
5         records[name] = [score]
6
7 student_records = {"Alice": [88, 92], "Bob": [70, 85]}
8 update_record(student_records, "Charlie", 91)
9 update_record(student_records, "Alice", 95)
10 print(student_records)
```

input

```
{'Alice': [88, 92, 95], 'Bob': [70, 85], 'Charlie': [91]}
```

Error Explanation:

The error raised as the Charlie is not present in the given dictionary and when we add a new value from second time to the key we can't append the value as its type was a integer not a list so to fix the error we have change the code in else block in the function i.e to records [name]=[score]