```
Start coding or generate with AI.
```

PYHTON DATATYPES

+ Code + Text

```
#Lists, Tuples , Sets , Dictionary
list1 = ["apple",12,23]
print(list1)
```

```
['apple', 12, 23]
```

colors = ['red', 'blue', 'green', 'yellow']

Using the colors list defined above, print the:

First element, Second element, Last element, Second-to-last element, Second and third elements, Element at index 4.

```
colors = ['red', 'blue', 'green', 'yellow']
print(colors[0])
print(colors[1])
print(colors[3])
print(colors[2])
print(colors[1:2])
print(colors[4])
```

```
red
blue
yellow
green
['blue']
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-5-8c431fb9a156> in <cell line: 7>()
      5 print(colors[2])
      6 print(colors[1:2])
----> 7 print(colors[4])
      8

IndexError: list index out of range
```

Below is a list with seven integer values representing the daily water level (in cm) in an imaginary lake. However, there is a mistake in the data. The third day's water level should be 693. Correct the mistake and print the changed list.

water_level = [730, 709, 682, 712, 733, 751, 740]

```
water_level = [730, 709, 682, 712, 733, 751, 740]
water_level[2] = 693
print(water_level)
```

```
[730, 709, 693, 712, 733, 751, 740]
```

Add the data for the eighth day to the list from above. The water level was 772 cm on that day. Print the list contents afterwards.

```
water_level.append(772)
print(water_level)
```

```
[730, 709, 693, 712, 733, 751, 740, 772]
```

Double-click (or enter) to edit

Still using the same list, add three consecutive days using a single instruction. The water levels on the 9th through 11th days were 772 cm, 770 cm, and 745 cm. Add these values and then print the whole list.

```
water_level.extend([772,770,745])
print(water_level)
```

```
[730, 709, 693, 712, 733, 751, 740, 772, 772, 770, 745]
```

There are two ways to delete data from a list: by using the index or by using the value. Start with the original water_level list we defined in the second exercise and delete the first element using its index. Then define the list again and delete the first element using its value.

```
water_level = [730, 709, 682, 712, 733, 751, 740]
water_level.pop(0)
print(water_level)
```

```
[709, 682, 712, 733, 751, 740]
```

1. Tuple Creation and Access: Create a tuple named colors with the elements 'red', 'green', and 'blue'. Access the second element of the tuple and print it.

```
colors = ('red', 'green', 'blue')
print(colors[1])
```

```
green
```

2. Immutable Nature: Explain in your own words why tuples are considered immutable. Attempt to modify an element in an existing tuple and observe the resulting error

```
# Tuples are immutable. because if try and modify an element  of the tuple , we get an error
colors = ('red', 'green', 'blue')
colors[1] = 'yellow'
```

3. Tuple Slicing: Given the tuple numbers = (1, 2, 3, 4, 5), use slicing to extract the elements from index 1 to 3 (inclusive). What would be the output of numbers[::-1]?

```
numbers = (1,2,3,4,5)
print(numbers[1:4])
print(numbers[::-1])
```

```
(2, 3, 4)
(5, 4, 3, 2, 1)
```

4. Tuple Concatenation and Repetition: Create two tuples, fruits with elements 'apple', 'banana', and berries with elements 'strawberry', 'blueberry'. Concatenate the two tuples and store the result in a new tuple named Repeat the combined_fruits tuple three times and print the result.

```
fruits = ('apple', 'banana', 'strawberry')
berries = ('blueberry', 'strawberry')
combined_fruits = fruits + berries
print(combined_fruits)
```

```
('apple', 'banana', 'strawberry', 'blueberry', 'strawberry')
```

5. Built-in Tuple Methods: Create a tuple named Use the grades with the elements 90, 85, 92, 88, 95.

   count() method to nd how many times the grade 88 appears in the tuple. Use the index() method to nd the index of the grade 92.

```
grades = (90, 85, 92, 88, 95)
print(grades.count(88))
print(grades.index(92))
```

```
1
2
```

. Advantages of Tuples: Explain one advantage of using tuples over lists in Python. Describe a scenario where the immutability of tuples could be bene cial in a program

7. Multiple Data Types in a Tuple: Create a tuple named mixed_types with elements 'apple', 42, and 3.14. Access and print the second element of the tuple.

```
mixed = ('apple', 42, 3.14)
print(mixed[1])
```

```
42
```

. Conversion: Convert the list ['cat', 'dog', 'rabbit'] into a tuple named animals Print the tuple to verify the conversion.

```
animal = ['cat', 'dog', 'rabbit']
animals = tuple(animal)
print(animals)
```

```
('cat', 'dog', 'rabbit')
```

9. Nested Tuples: Create a tuple outer_tuple with two elements: 'apple' and another tuple ('red', 'green', 'yellow'). Access the second element of the inner tuple and print it

```
outer_tuple = ('apple', ('red', 'green', 'yellow'))
print(outer_tuple[1][1])
```

```
green
```

1.

- You are managing the inventory for a small bookstore. Create a list of book titles available in the store. Add new titles to the list as they arrive. If a book is sold out, remove it from the list. Write a function to check if a specific book is in stock.

You are managing the inventory for a small bookstore. Create a list of book titles available in the store. Add new titles to the list as they arrive. If a book is sold out, remove it from the list. Write a function to check if a specific book is in stock.

```
Start coding or generate with AI.
```

\