1. **What are data structures, and why are they important.**

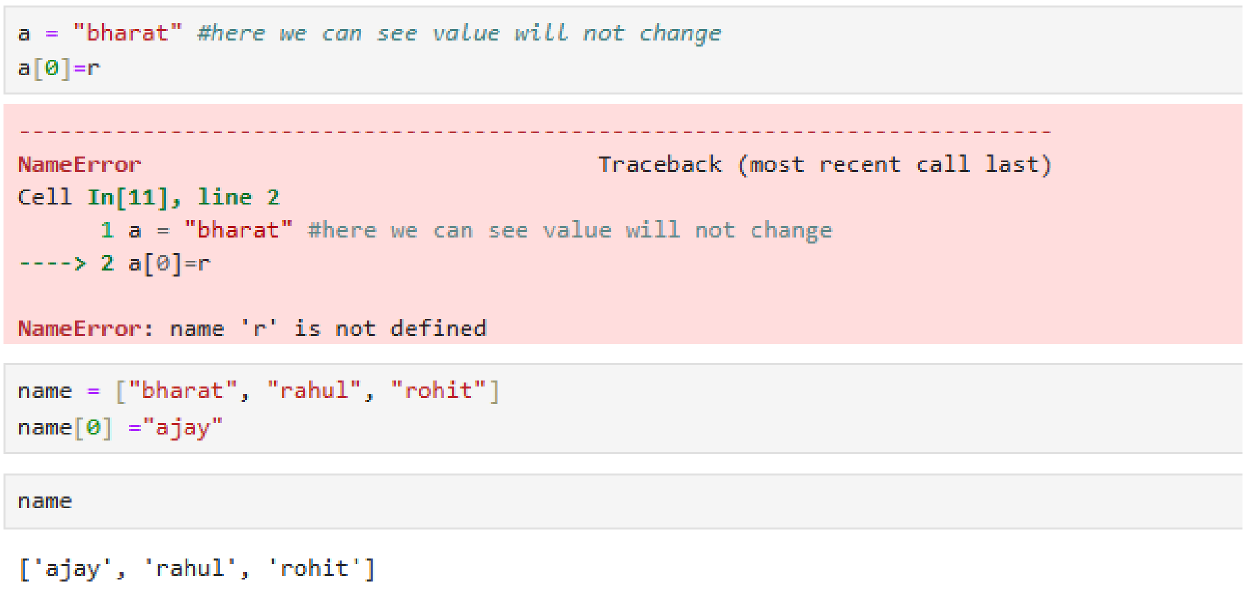
Data structures are the ways of organizing and storing data so that if can accessed and implemented effectively.

This makes codes more efficient, easier to understand and better able to handle large amounts of data.

2. **Explain the difference between mutable and immutable data types with examples.**

Mutable data types are those whose values we can change like list.

immutable data types are those whose values we can not change like sting, tuple etc.



```
a = "bharat" #here we can see value will not change
a[0]=r
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 2
      1 a = "bharat" #here we can see value will not change
----> 2 a[0]=r

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

```
name = ["bharat", "rahul", "rohit"]
name[0] ="ajay"
```

```
name
```

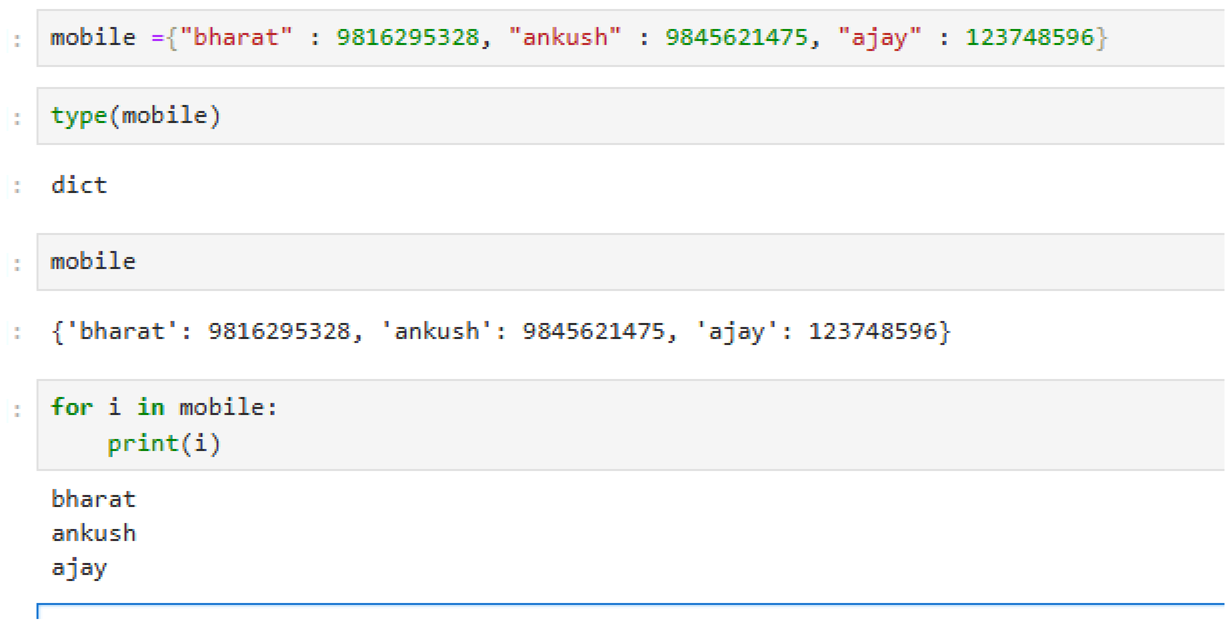
```
['ajay', 'rahul', 'rohit']
```

3. **What are the main differences between lists and tuples in Python.**

Main difference is that lists are mutable while tuples are immutable data types.

4. **Describe how dictionaries store data.**

Dictionaries stores data in the key and value pair.



```
mobile ={"bharat" : 9816295328, "ankush" : 9845621475, "ajay" : 123748596}
```

```
type(mobile)
```

```
dict
```

```
mobile
```

```
{'bharat': 9816295328, 'ankush': 9845621475, 'ajay': 123748596}
```

```
for i in mobile:
    print(i)
```

```
bharat
ankush
ajay
```

5. Why might you use a set instead of a list in Python.

We use due to uniqueness, performance. Avoid duplicate values. No concept of indexing.

6. What is a string in Python, and how is it different from a list.

A String in python is a sequence of characters enclosed with in single quotes, double quotes or triple quotes.

Strings are immutable while list is mutable.

7. How do tuples ensure data integrity in Python.

Due to immutability factor tuples ensures data integrity. E.g we need to provide security in some cases like, Account number, Aadhar number ect. Which values should not be changed.

8. What is a hash table, and how does it relate to dictionaries in Python?

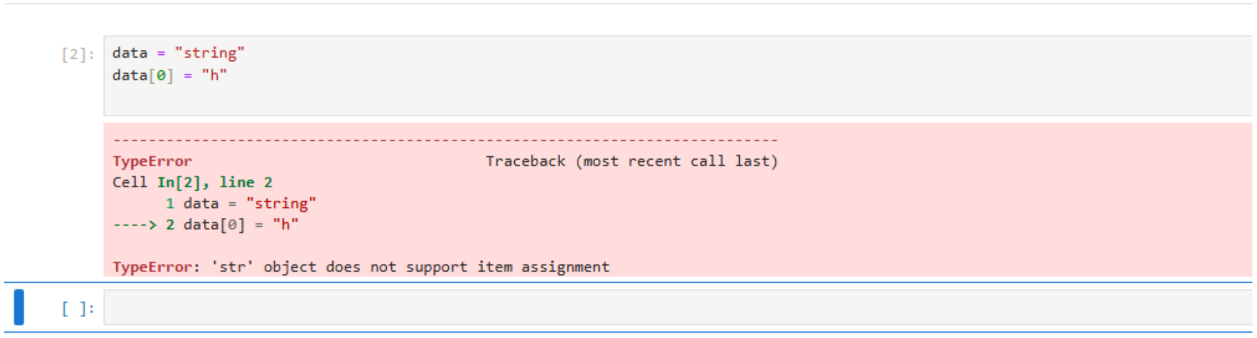
A hash table is a data structure that maps keys to values using a hash function. This function computes an index into an array of buckets or slots, from which the desired value can be found. Hash tables are widely used for their efficiency in performing insertions, deletions, and lookups.

9. Can lists contain different data types in Python?

Yes list can contain different data types because it is heterogeneous data structures.

10. Explain why strings are immutable in Python?

Strings are immutable in python because we can not change the value of particular memory block elements.



```
[2]: data = "string"
     data[0] = "h"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[2], line 2
      1 data = "string"
----> 2 data[0] = "h"

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

```
[ ]:
```

11. What advantages do dictionaries offer over lists for certain tasks?

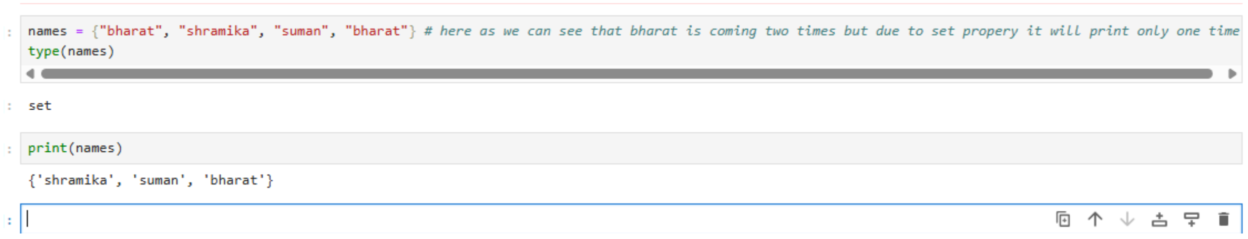
Maintain order, making them great when sequence matters. - Allow duplicates—ideal for collections where repetition is key. - Quick access to elements by index, perfect for iteration. - Perfect for key-value pairs—associate unique keys with values.

12. Describe a scenario where using a tuple would be preferable over a list.

Due to immutability factor tuples ensures data integrity. E.g we need to provide security in some cases like, Account number, Aadhar number ect. Which values should not be changed.

13. How do sets handle duplicate values in Python?

It is the property of the sets in python that remove duplicate values.



```
: names = {"bharat", "shramika", "suman", "bharat"} # here as we can see that bharat is coming two times but due to set propery it will print only one time
  type(names)
```

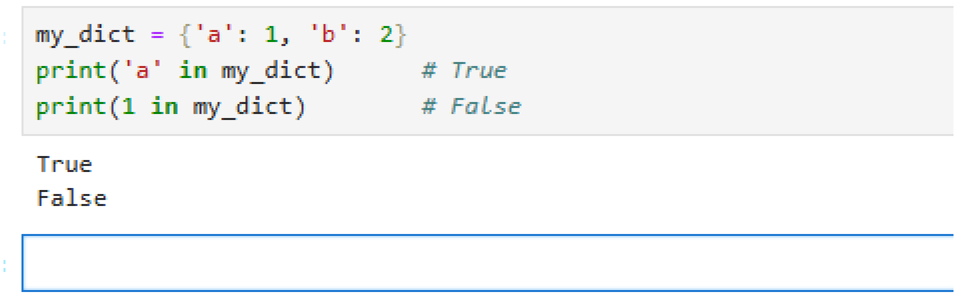
```
: set
```

```
: print(names)
```

```
{'shramika', 'suman', 'bharat'}
```

14. How does the “in” keyword work differently for lists and dictionaries?

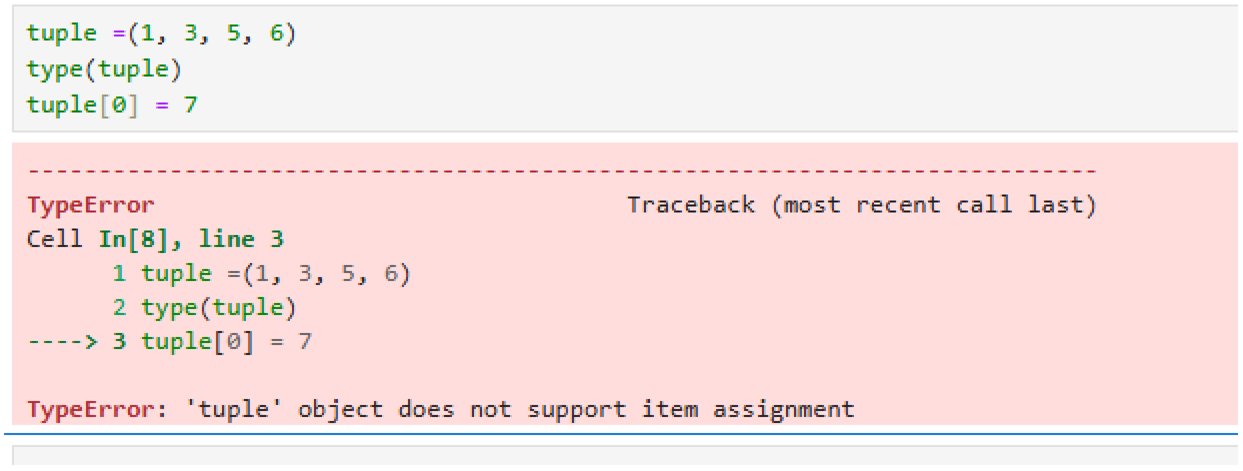
It gives us Boolean value.


```python
my_dict = {'a': 1, 'b': 2}
print('a' in my_dict)      # True
print(1 in my_dict)        # False
```

```
True
False
```

15. Can you modify the elements of a tuple? Explain why or why not?

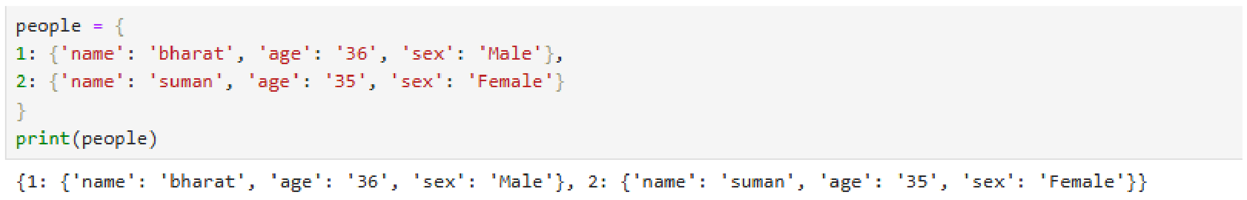
Due to immutability factor tuples ensures data integrity. We can not modify the elements of a tuple.


```python
tuple =(1, 3, 5, 6)
type(tuple)
tuple[0] = 7
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[8], line 3
      1 tuple =(1, 3, 5, 6)
      2 type(tuple)
----> 3 tuple[0] = 7

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

16. What is a nested dictionary, and give an example of its use case?

A nested dictionary in Python is a dictionary inside another dictionary. This allows you to store complex data structures in a single dictionary. Each key in the outer dictionary can map to another dictionary, enabling hierarchical data storage.


```python
people = {
1: {'name': 'bharat', 'age': '36', 'sex': 'Male'},
2: {'name': 'suman', 'age': '35', 'sex': 'Female'}
}
print(people)
```

```
{1: {'name': 'bharat', 'age': '36', 'sex': 'Male'}, 2: {'name': 'suman', 'age': '35', 'sex': 'Female'}}
```

17. Describe the time complexity of accessing elements in a dictionary.

Accessing elements in a **dictionary** in Python is generally **very fast**—and that's thanks to how dictionaries are implemented under the hood using **hash tables**.

```python
my_dict = {'apple': 1, 'banana': 2, 'cherry': 3}
print(my_dict['banana'])  # O(1)
```

18. In what situations are lists preferred over dictionaries?

Lists are preferred over dictionaries when **order, indexing, and simple sequences** are important. Here are some specific situations where using a **list** makes more sense than a dictionary

Where data does not need labels or keys.

19. Why are dictionaries considered unordered, and how does that affect data retrieval?

Because their **primary purpose** is **fast key-based access**, not sequencing.

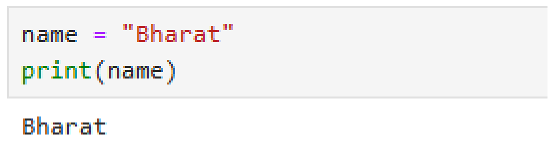
Since **Python 3.7+**, dictionaries **do** preserve insertion order **as an implementation detail**, and in Python 3.8+ it's officially part of the language spec.

20. Explain the difference between a list and a dictionary in terms of data retrieval.

List is accessed by index position while dictionary is accessed by key.

Practical questions

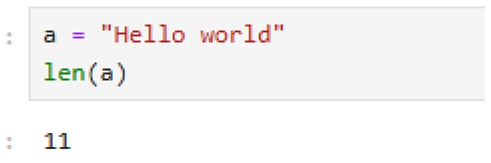
1. Write a code to create a string with your name and print it.



```
name = "Bharat"
print(name)
```

```
Bharat
```

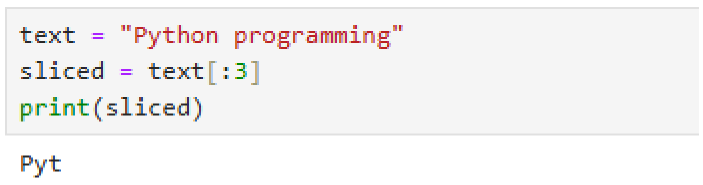
2. Write a code to find the length of the string "Hello World"



```
a = "Hello world"
len(a)
```

```
11
```

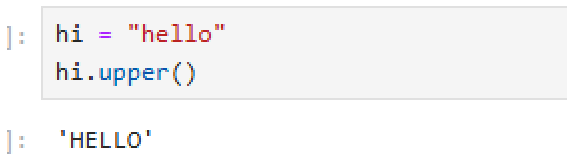
3. Write a code to slice the first 3 characters from the string "Python Programming"



```
text = "Python programming"
sliced = text[:3]
print(sliced)
```

```
Pyt
```

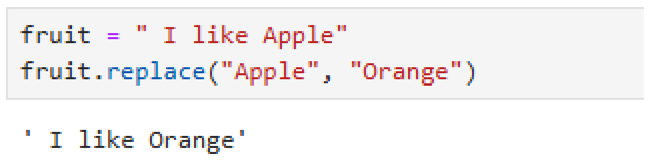
4. Write a code to convert the string "hello" to uppercase



```
hi = "hello"
hi.upper()
```

```
'HELLO'
```

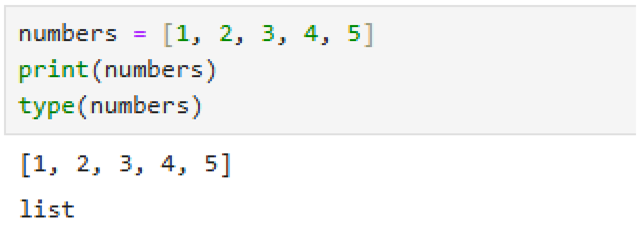
5. Write a code to replace the word "apple" with "orange" in the string "I like apple"



```
fruit = " I like Apple"
fruit.replace("Apple", "Orange")
```

```
' I like Orange'
```

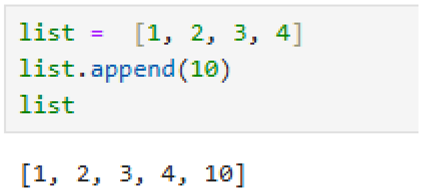
6. Write a code to create a list with numbers 1 to 5 and print it



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

```
[1, 2, 3, 4, 5]
list
```

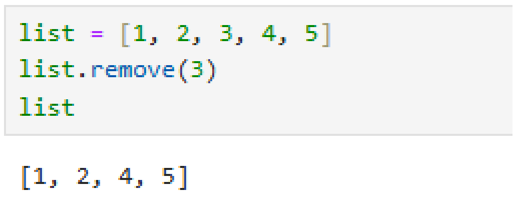
7. Write a code to append the number 10 to the list [1, 2, 3, 4]



```
list =  [1, 2, 3, 4]
list.append(10)
list
```

```
[1, 2, 3, 4, 10]
```

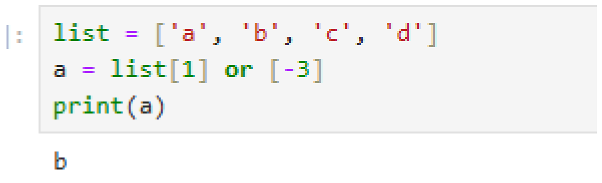
8. Write a code to remove the number 3 from the list [1, 2, 3, 4, 5]



```
list = [1, 2, 3, 4, 5]
list.remove(3)
list
```

```
[1, 2, 4, 5]
```

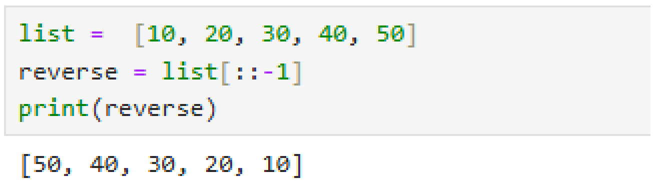
9. Write a code to access the second element in the list ['a', 'b', 'c', 'd']



```
list = ['a', 'b', 'c', 'd']
a = list[1] or [-3]
print(a)
```

```
b
```

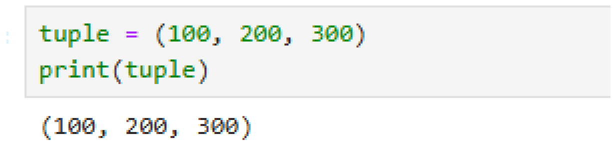
10. Write a code to reverse the list [10, 20, 30, 40, 50].



```
list =  [10, 20, 30, 40, 50]
reverse = list[::-1]
print(reverse)
```

```
[50, 40, 30, 20, 10]
```

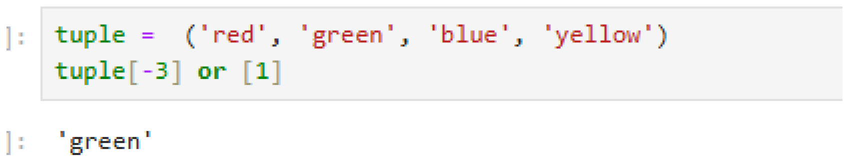
11. Write a code to create a tuple with the elements 100, 200, 300 and print it.



```
tuple = (100, 200, 300)
print(tuple)
```

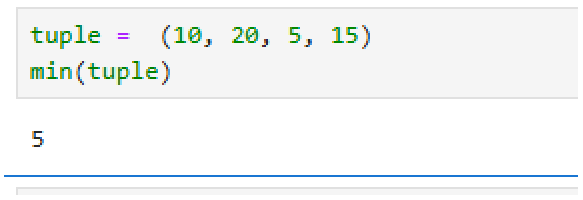
```
(100, 200, 300)
```

12. Write a code to access the second-to-last element of the tuple ('red', 'green', 'blue', 'yellow').



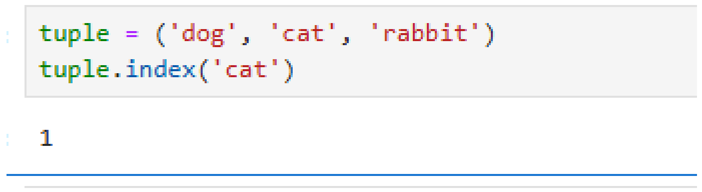
```
]: tuple =  ('red', 'green', 'blue', 'yellow')
   tuple[-3] or [1]

]:  'green'
```

13. Write a code to find the minimum number in the tuple (10, 20, 5, 15).



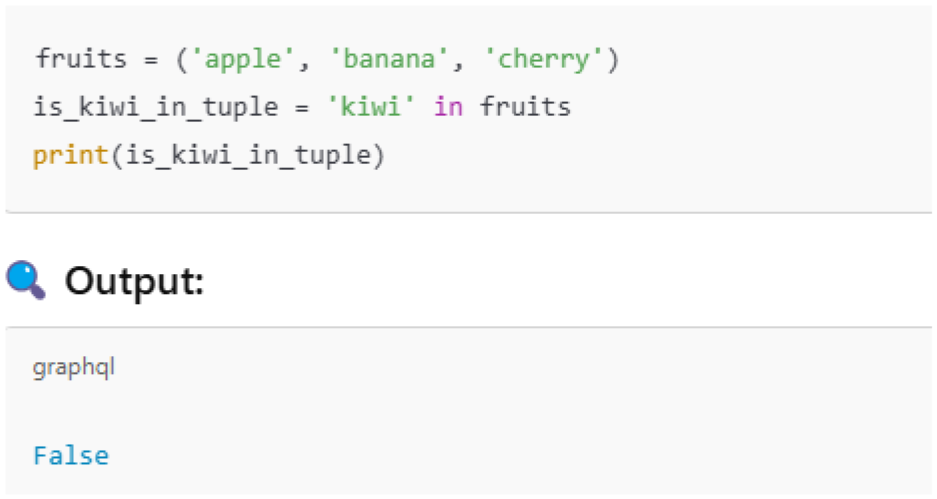
```
tuple =  (10, 20, 5, 15)
min(tuple)

5
```

14. Write a code to find the index of the element "cat" in the tuple ('dog', 'cat', 'rabbit')



```
tuple = ('dog', 'cat', 'rabbit')
tuple.index('cat')

1
```

15. Write a code to create a tuple containing three different fruits and check if "kiwi" is



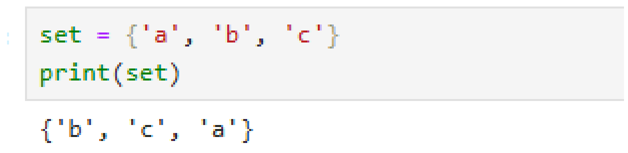
```
fruits = ('apple', 'banana', 'cherry')
is_kiwi_in_tuple = 'kiwi' in fruits
print(is_kiwi_in_tuple)
```

🔍 **Output:**

```
graphql

False
```

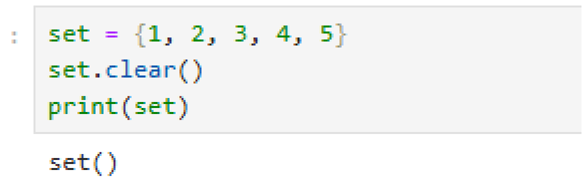
in it.

16. Write a code to create a set with the elements 'a', 'b', 'c' and print it.



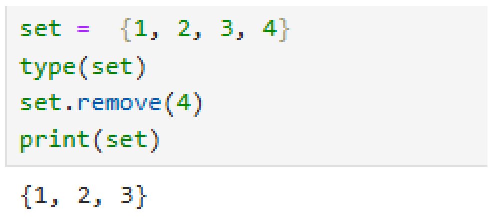
```
set = {'a', 'b', 'c'}
print(set)

{'b', 'c', 'a'}
```

17. Write a code to clear all elements from the set {1, 2, 3, 4, 5}.



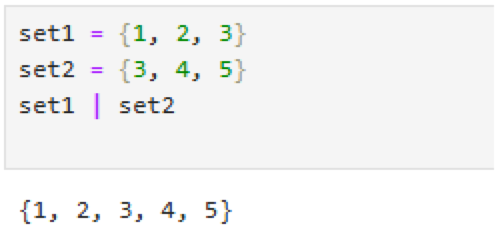
```
:  set = {1, 2, 3, 4, 5}
   set.clear()
   print(set)

   set()
```

18. Write a code to remove the element 4 from the set {1, 2, 3, 4}



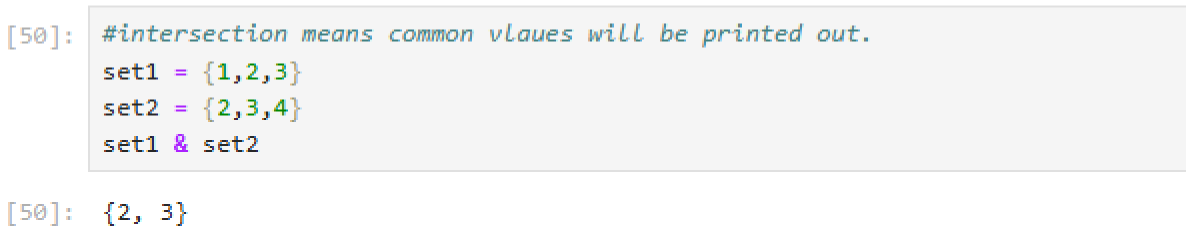
```
set =  {1, 2, 3, 4}
type(set)
set.remove(4)
print(set)

{1, 2, 3}
```

19. Write a code to find the union of two sets {1, 2, 3} and {3, 4, 5}.



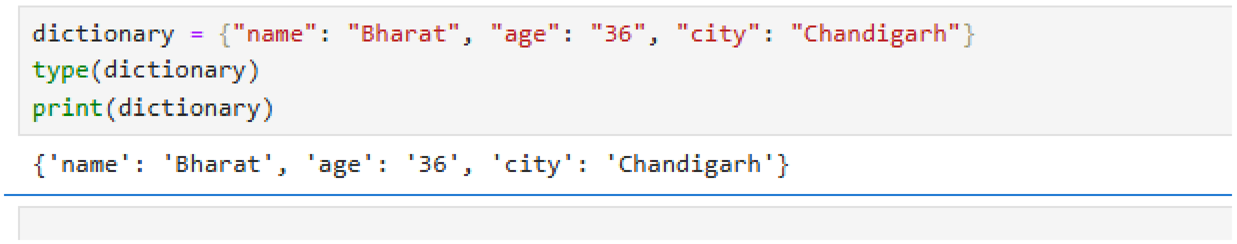
```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
set1 | set2


{1, 2, 3, 4, 5}
```

20. Write a code to find the intersection of two sets {1, 2, 3} and {2, 3, 4}.



```
[50]:  #intersection means common vlaues will be printed out.
       set1 = {1,2,3}
       set2 = {2,3,4}
       set1 & set2

[50]:  {2, 3}
```

21. Write a code to create a dictionary with the keys "name", "age", and "city", and print it



```
dictionary = {"name": "Bharat", "age": "36", "city": "Chandigarh"}
type(dictionary)
print(dictionary)

{'name': 'Bharat', 'age': '36', 'city': 'Chandigarh'}
```

22. Write a code to add a new key-value pair "country": "USA" to the dictionary {'name': 'John', 'age': 25}.

```python
dictionary =  {'name': 'John', 'age': 25}
dictionary['country'] = 'USA'
print(dictionary)
```

```
{'name': 'John', 'age': 25, 'country': 'USA'}
```

23. Write a code to access the value associated with the key "name" in the dictionary {'name': 'Alice', 'age': 30}.

```python
dictionary = {'name': 'Alice', 'age': 30}
dictionary["name"]
```

```
'Alice'
```

24. Write a code to remove the key "age" from the dictionary {'name': 'Bob', 'age': 22, 'city': 'New York'}.

```python
person = {'name': 'Bob', 'age': 22, 'city': 'New York'}
del person['age']
print(person)
```

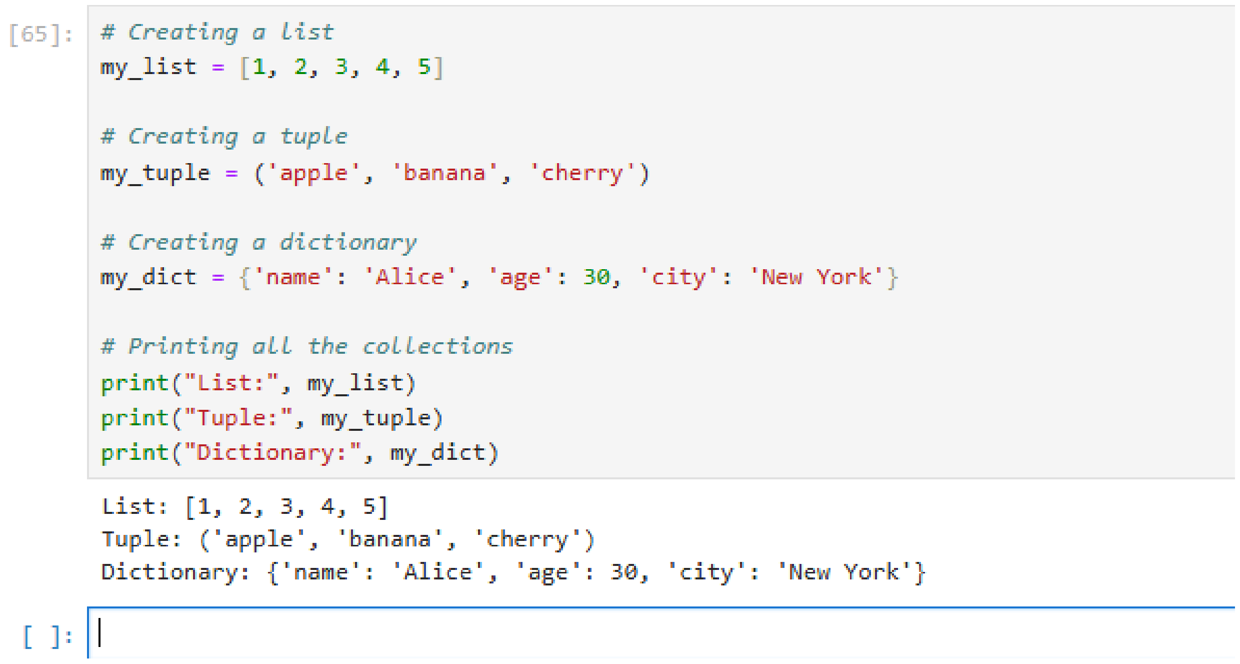
```
{'name': 'Bob', 'city': 'New York'}
```

25. Write a code to check if the key "city" exists in the dictionary {'name': 'Alice', 'city': 'Paris'}.

```python
person = {'name': 'Alice', 'city': 'Paris'}
key_exists = 'city' in person
print(key_exists)
```

```
True
```

26. Write a code to create a list, a tuple, and a dictionary, and print them all.

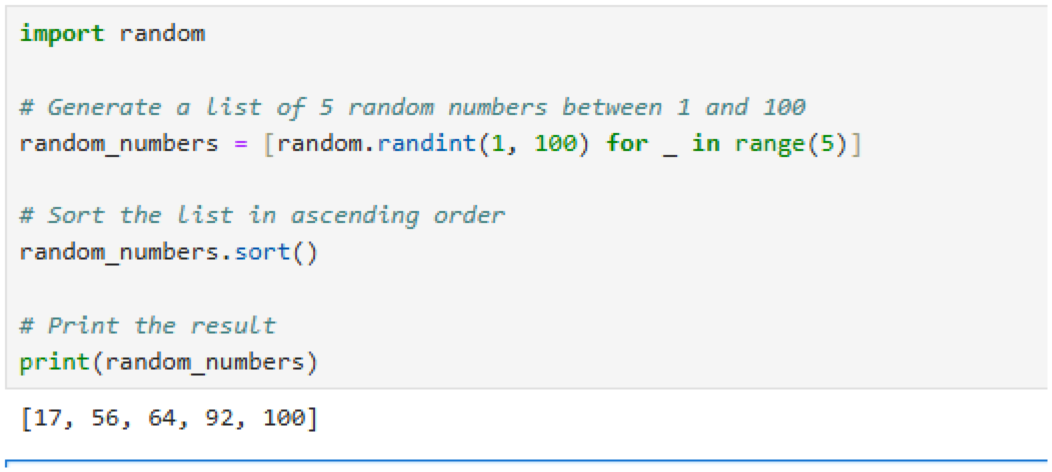


```
[65]:
# Creating a list
my_list = [1, 2, 3, 4, 5]

# Creating a tuple
my_tuple = ('apple', 'banana', 'cherry')

# Creating a dictionary
my_dict = {'name': 'Alice', 'age': 30, 'city': 'New York'}

# Printing all the collections
print("List:", my_list)
print("Tuple:", my_tuple)
print("Dictionary:", my_dict)
```

```
List: [1, 2, 3, 4, 5]
Tuple: ('apple', 'banana', 'cherry')
Dictionary: {'name': 'Alice', 'age': 30, 'city': 'New York'}
```

```
[ ]:
```

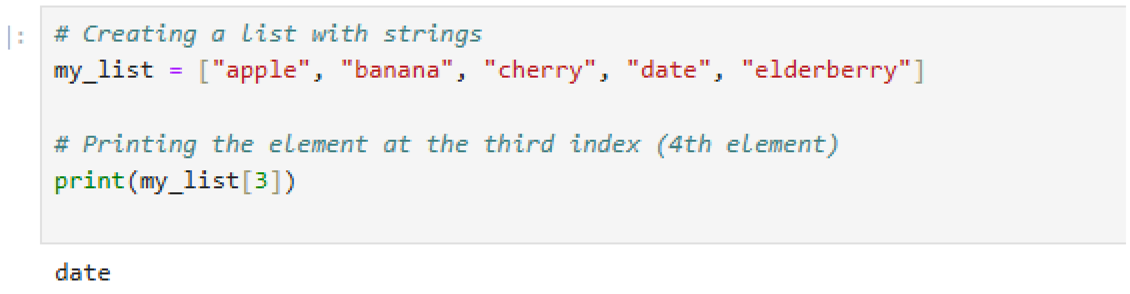
27. Write a code to create a list of 5 random numbers between 1 and 100, sort it in ascending order, and print the result.(replaced)



```
import random

# Generate a list of 5 random numbers between 1 and 100
random_numbers = [random.randint(1, 100) for _ in range(5)]

# Sort the list in ascending order
random_numbers.sort()

# Print the result
print(random_numbers)
```

```
[17, 56, 64, 92, 100]
```

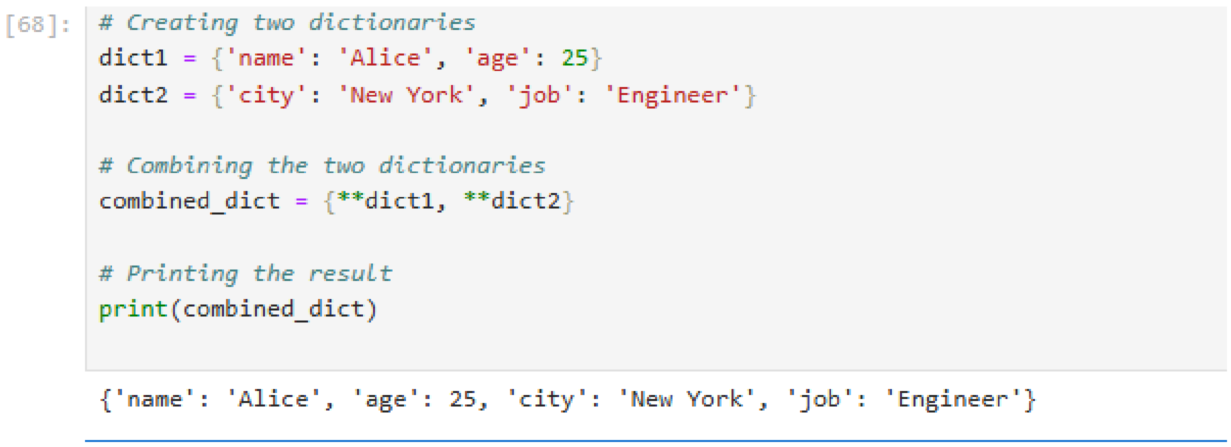
28. Write a code to create a list with strings and print the element at the third index



```
# Creating a list with strings
my_list = ["apple", "banana", "cherry", "date", "elderberry"]

# Printing the element at the third index (4th element)
print(my_list[3])
```

```
date
```

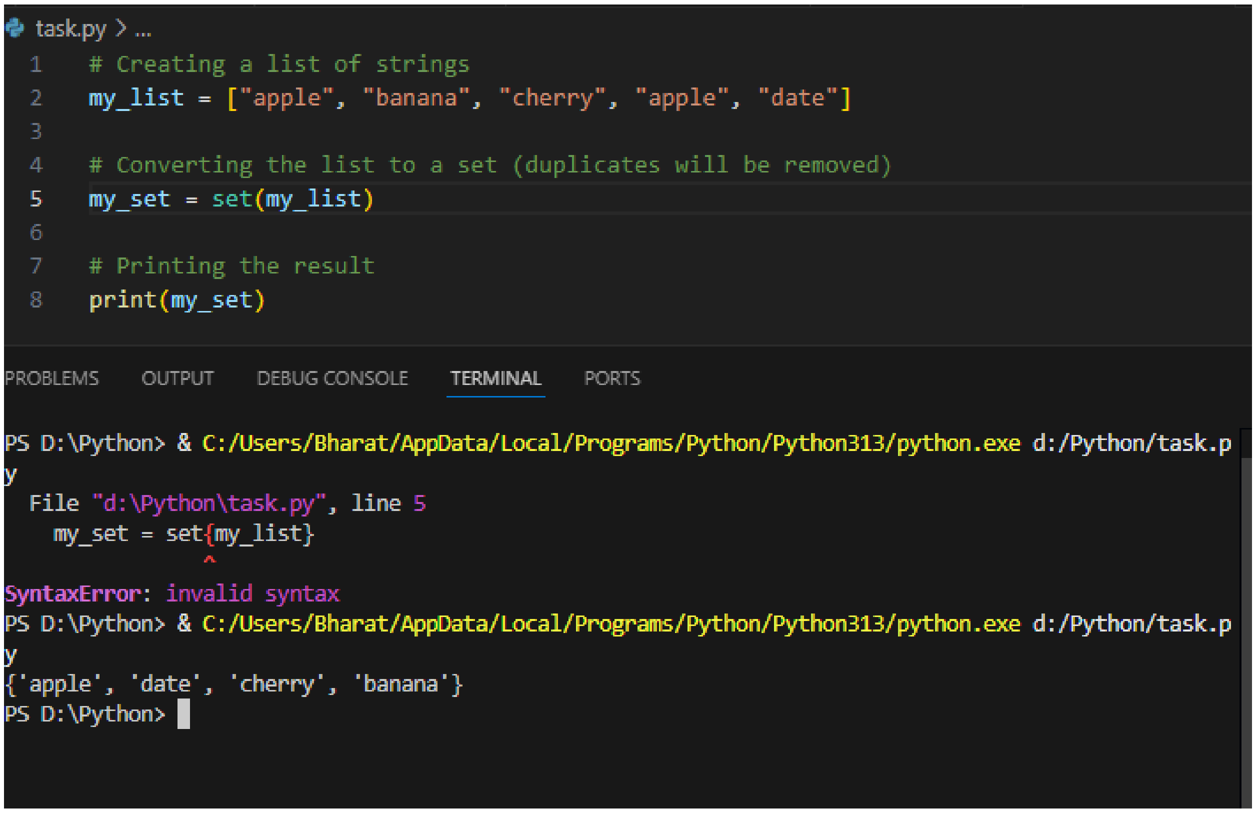
29. Write a code to combine two dictionaries into one and print the result.



```python
# Creating two dictionaries
dict1 = {'name': 'Alice', 'age': 25}
dict2 = {'city': 'New York', 'job': 'Engineer'}

# Combining the two dictionaries
combined_dict = {**dict1, **dict2}

# Printing the result
print(combined_dict)
```

```
{'name': 'Alice', 'age': 25, 'city': 'New York', 'job': 'Engineer'}
```

30. Write a code to convert a list of strings into a set



```python
# Creating a list of strings
my_list = ["apple", "banana", "cherry", "apple", "date"]

# Converting the list to a set (duplicates will be removed)
my_set = set(my_list)

# Printing the result
print(my_set)
```

```
PS D:\Python> & C:/Users/Bharat/AppData/Local/Programs/Python/Python313/python.exe d:/Python/task.p
y
  File "d:\Python\task.py", line 5
    my_set = set{my_list}
                ^
SyntaxError: invalid syntax
PS D:\Python> & C:/Users/Bharat/AppData/Local/Programs/Python/Python313/python.exe d:/Python/task.p
y
{'apple', 'date', 'cherry', 'banana'}
PS D:\Python> 
```