In [1]:
```
#1 Discuss string slicing and provide examples
```

```
* String- Characters arranged in an unchangeable order.
          Each character in the string can be accessed by its index or
position.
          Substrings can be extracted using slicing.
```

In [2]:
```
#examples of string slicing
string1= 'I am a good student'
string1[:4]
```

Out[2]:
```
'I am'
```

In [3]:
```
string1[5:11]
```

Out[3]:
```
'a good'
```

In [5]:
```
string1[12:19]
```

Out[5]:
```
'student'
```

In [6]:
```
#2 Explain the key features of lists in Python
```

```
* List- Ordered collection of elements.
        It can hold items of various data types (numbers, strings etc).
        They are mutable.
        List is vesatile for storing and managing collection that might
change.
```

In [7]:
```
#example
l=[]
type(l)
```

Out[7]:
```
list
```

In [8]:
```
#3 Describe how to access, modify, and delete elements in a list with exam
```

```
* You can add, remove, or modify elements within the lists using
indexing and slicing.
```

In [10]:
```
#example of adding or modifying an element to the list>> three ways
# first with the help of keyword append>>it will just add the value to the
_name=['Sanjay', 'Ajay']
_name.append('Dhananjay')
_name
```

Out[10]:
```
['Sanjay', 'Ajay', 'Dhananjay']
```

In [11]:
```
# Second with the help of keyword insert>>it takes position according to i
veg_list=['cauliflower','cabbage','carrot']
veg_list.insert(2,'potato')
veg_list
```

Out[11]:
```
['cauliflower', 'cabbage', 'potato', 'carrot']
```

In [14]:
```
# Third with the help of keyword extend
employee_id=[1,2,3,4,5]
alphabets=['A','B','C','D','E']
employee_id.extend(alphabets)
employee_id
```

Out[14]:
```
[1, 2, 3, 4, 5, 'A', 'B', 'C', 'D', 'E']
```

In [15]:
```
#example of how to access an element from the list
# we can use indexing(negative or positive) or slicing(negative or positiv
world=['train','aeroplane',21,True, 4+4j,1.7]
world[3]
```

Out[15]:
```
True
```

In [16]:
```
world[-1]
```

Out[16]:
```
1.7
```

In [17]:
```
world[2:]
```

Out[17]:
```
[21, True, (4+4j), 1.7]
```

In [18]:
```
world[:-5]
```

Out[18]:
```
['train']
```

In [19]:
```
#example of how to delete an element from the list
# Keyword delete>> it will delete whole items which are present in the lis
_book=['author','pages','cover']
del _book
```

In [20]:
```
_book
```

```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
<ipython-input-20-0f27ca4e6a2e> in <module>
----> 1 _book

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

```
In [21]: #4 Compare and contrast tuples and lists with examples
```

```
* List- Ordered, mutable collections of elements,heterogenous. Think of
shopping lists or task lists. Lists can hold items of various data
types (numbers, strings, even other lists).

* Tuple- Ordered, immutable collections of elements, similar to lists.
However, once created, the items in a tuple cannot be changed. They
provide a secure way to store data that shouldn't be modified.
```

```
In [22]: #examples of tuple and list
         #List>>it is mutable means it can be modified after creation.
         # represented by square bracket []
         list=['data structure','algorithm','web']
         list[2]='pwskills'
         list

Out[22]: ['data structure', 'algorithm', 'pwskills']
```

```
In [23]: #tuple>> it is immutable it cannot be modified after creation.
         # represented by circular bracket()
         tuple=('data structure','algorithm','web')
         tuple[-1]

Out[23]: 'web'
```

```
In [24]: tuple[-1]='pwskills'

         ---------------------------------------------------------------------------
         --
         TypeError                                 Traceback (most recent call las
         t)
         <ipython-input-24-aa6f6cf97425> in <module>
         ----> 1 tuple[-1]='pwskills'

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

```
In [25]: #5 Describe the key features of sets and provide examples of their use
```

```
* Sets- Unordered collections of unique elements. The order doesn't
matter, and duplicate entries are not allowed.
   Sets are useful for checking membership (if an item exists) or
finding the intersection/difference between sets.
   Represented by curly bracket{1,34}
   It can also take any value.
```

In [26]:
```
#example of sets>>how they are unordered and unique
# so we cannot take out element by indexing because position is not fixed.
_attendence={'present','absent','leave','leave','leave','absent'}
_attendence
```

Out[26]:
```
{'absent', 'leave', 'present'}
```

In [27]:
```
#6 Discuss the use cases of tuples and sets in Python programming
```

```
* Tuples are used in-
(i)  Immutable Data Storage- immutable, meaning once created, they
cannot be modified. This makes them ideal for storing data that should
not change throughout the program.

(ii) Storing heterogenous data-Tuples can hold elements of different
data types.
```

In [29]:
```
#Example: Storing the coordinates of a point (x, y) in a 2D space.
point=(10,20)
type(point)
# now these co-ordinates are fixed.
```

Out[29]:
```
tuple
```

In [31]:
```
#Example: Storing a person’s name, age, and email in a single tuple.
tuple1=('sanjay','ajay')
age=(26,30)
email=('san@gmail.com','aj@gmail.com')
tuple=(name,age,email)
tuple
```

Out[31]:
```
(('Sanjay', 'Ajay'), (26, 30), ('san@gmail.com', 'aj@gmail.com'))
```

```
* Sets are used in-
(i)  Removing Duplicates from a Set- When you need to eliminate
duplicate elements from a list, converting the list to a set is a quick
and efficient way to do so.

(ii) Set Operations (Union, Intersection, Difference)- which are useful
in fields like data analysis, database management, and computational
biology.
```

In [38]:
```
#Example- It can be used in lucky draw because any number can come random
lucky_no= {1,5,7,9,11}
lucky_no.pop()
```

Out[38]:
```
1
```

In [43]:
```
lucky_no.pop()
```

Out[43]:
```
5
```

In [44]:
```
lucky_no
```

Out[44]:
```
{7, 9, 11}
```

In [45]:
```
#7 Describe how to add, modify, and delete items in a dictionary with exam
```

```
* Dictionary- Keys are unique and immutable.
  represented by {}
  collection of data values, used to store data values like a map.
```

In [46]:
```
d={}
type(d)
```

Out[46]:
```
dict
```

In [47]:
```
d={"key":"value"}
type(d)
```

Out[47]:
```
dict
```

In [53]:
```
#Example- Database Record Representation
Record={'Id':1,
        'name':'Sanjay',
        'age':26,
        'email':'san@gmail.com'}
```

In [60]:
```
#Example of adding element to the dictionary
d={'name':'Sanjay','email':'san@gmail.com','contact':123456789}
d1={'game':'football'}
d.update(d1)
d
```

Out[60]:
```
{'name': 'Sanjay',
 'email': 'san@gmail.com',
 'contact': 123456789,
 'game': 'football'}
```

In [4]:
```
#Example of modifying element to the dictionary
d={'name':'Sanjay','email':'san@gmail.com','contact':123456789}
d['name']='Kumar'
d
```

Out[4]:
```
{'name': 'Kumar', 'email': 'san@gmail.com', 'contact': 123456789}
```

In [5]:
```
#Example of delete element to the dictionary
d={'name':'Sanjay','email':'san@gmail.com','contact':123456789}
del d['contact']
d
```

Out[5]:
```
{'name': 'Sanjay', 'email': 'san@gmail.com'}
```

```
In [6]: #8  Discuss the importance of dictionary keys being immutable and provide
```

```
* Importance of dictionary keys-
 We can use string, number, tuple as a key in Python dictionary because
it can not be modified.
  Tuples are immutable and can therefore be used as dictionary keys.
 Python dictionary keys must always be immutable, which means you can
not update the dictionary key     in runtime..
```

```
In [7]: #Example
        mydict = { [1, 2]: '12' }
        print(mydict[[1, 2]])
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-7-0ffcdce56159> in <module>
      1 #Example
----> 2 mydict = { [1, 2]: '12' }
      3 print(mydict[[1, 2]])

TypeError: unhashable type: 'list'
```

```
In [8]: d={{1,2,3}:'Sanjay'}
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-8-bfefa71e6bf9> in <module>
----> 1 d={{1,2,3}:'Sanjay'}

TypeError: unhashable type: 'set'
```

So only string and numbers can be the key of the dictionary.

```
In [ ]:
```