```
#1)discuss string slicing and provide example
#ANSWER- string slicing is a powerful feature in python that allows you to extract a porti
#of string by specifying a range of indices the basic syntax for slicing
#EXAMPLES OF STRING SLICING
text="hello world"
print(text[0:5])
print(text[7:12])# here i understood using the following syntax we can slice the string according to our uses
```

```
hello
orld
```

Generate randomly select 5 items from a list Close

```
#2)EXPLAIN KEY FEATURES OF LIST IN PYTHON.
#ANSWER- list in python are versatile and powerful data structure that allow you to store collections of items here are some key feature
#list maintain the order  of elements as they were added you can acess elements using their index
#list are mutable meaning you can change their content after creation
#list can grow and shrink in size you can add or remove elements as nedded
#list can contain elements of different data types including integers strings and even other lists


#3)describe how to acess modify and delete elements in a list with examples\
#acessing elements:you can acess elements in a list using their index python uses zero based indexing
#so thr first
#example
my_list=['apple','banana','orange','grapes']
#acessing elements here
print(my_list[0])
print(my_list[2])
#acessing last element
print(my_list[-1])
```

```
apple
orange
grapes
```

```
#modifying element
my_list=['apple','banana','orange','grapes']
my_list[1]='mango'
print(my_list)
```

```
['apple', 'mango', 'orange', 'grapes']
```

```
#deleting element
my_list=['apple','banana','orange','grapes']
del my_list[2]
print(my_list)
```

```
['apple', 'banana', 'grapes']
```

```
#4)COMPARE AND CONTRAST TUPLES AND LISTS WITH EXAMPLES
#ANSWER-tuples and lists are both built in data structure in python that can store collection of items   however they have distinct char
#1)MUTABILITY- LISTS:mutable meaning you can modify their content
#tuples:immutable meaning once a tuple is created its contents cannot be changed
#examples,#list
my_list=[1,2,3]
my_list[1]=0# modifying an element
print(my_list)
#tuples
my_tuple=(1,2,3)# this code will raise an error
my_tuple[1]=0
```

```
[1, 0, 3]
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-f4a1aec4c209> in <cell line: 11>()
      9 #tuples
     10 my_tuple=(1,2,3)
---> 11 my_tuple[1]=0

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

Next steps: Explain error

```
#syntax LIST:defined using square brackets []
#tuples: define using parentheses()
#example
```

```
my_list=[1,2,3]
my_tuple=(1,2,3)


#performance
#list:generally have higher memory overhead due to their mutability
#tuples:more memory efficient and can be faster for iteration since they are immutable


#5)DESCRIBE KEY FEATURES OF SETS AND PROVIDE EXAMPLE OF THEIR USE
#ANSWER- sets are a built in data structure in python that store unordered collections of unique elements they have several key features
#sets do not maintain any particular order
#the elements in sets cannot be indexed or sliced
#sets automatically eleminates duplicate entries if you add duplicate items
#sets are mutable
#because sets are unordered you cannot acess elements using indices you can check



#USED CASES OF SETS
#removing duplicate elemrnt
my_list=[1,2,2,3,4,4,5]
unique_set=set(my_list)
print(unique_set)
```

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

```
my_set={1,2,3}
my_set.add(4)
print(my_set)
```

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

```
set={1,2,3}
set2={3,4,5}
union_set=set.union(set2)
print(union_set)
```

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

```
#6)DISCUSS THE USED CASES OF TUPLES AND SETS IN PYTHON PROGRAMMING
#ANSWER-uses cases of tuples
location_data={
    (40.7128,-74.0060):'new york city',
    (34.0522,118.2437):'los angeles'
}


employee=("john doe",25,"mumbai")


values=(1,2,3,4,5,6,7,8,9,10)
evens=tuple(filter(lambda x:x%2==0,values))
print(evens
```

```
  File "<ipython-input-13-f8958738beb6>", line 3
    print(evens
                ^
SyntaxError: incomplete input
```

Next steps: Fix error

```
#used cases of sets
my_set={1,2,3}
if 3 in my_set:
    print("3 is in the set")
```

```
3 is in the set
```

```
dataset_a={1,2,3,4,5}
dataset_b={4,5,6,7,8}
all_unique=dataset_a|dataset_b
print(all_unique)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

```
#7)DESCRIBE HOW TO ADD ,MODIFY,AND DELETE ITEM IN DICTIONARY WITH EXAMPLE
#ANSWER- adding items example
```

```
#you can add a new key value pair to dictionary by simply assigning a value to new ksy
my_dict={'name':'alice','age':25}
my_dict['city']='new york'
print(my_dict)
```

```
{'name': 'alice', 'age': 25, 'city': 'new york'}
```

```
#modifying items
#to modify an existing value you can assign an new value to existing one
my_dict['age']=31
print(my_dict)
```

```
{'name': 'alice', 'age': 31, 'city': 'new york'}
```

```
#deleting items
del my_dict['city']
print(my_dict)
```

```
{'name': 'alice', 'age': 31}
```

```
#DISCUSS THE IMPORTANCE OF DICTIONARY KEYS BEING IMMUTABLE AND PROVIDE EXAMPLES
#ANSWER-in pyhon dictionary krys must br immutable meaning they cannot change after being created
#this immutability is crucial for several reasons
#the requirements for dictionary keys to be immutable ensures the integrity efficiency and predictability
#of the dictionary operations using immutable types like strings numbers tuples as keys allows python to manage dictionaries effectivel
```

```
#examples
my_dict={'name':'alice','age':25}
print(my_dict['name'])
```

```
alice
```

```
my_dict={(1,2):'value1',(3,4):'value2'}
print(my_dict[(1,2)])
```

```
value1
```

```
my_dict={{'key':'value'}:'dict'}#this will rase atype error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-b01cd4fdbe0d> in <cell line: 1>()
----> 1 my_dict={{'key':'value'}:'dict'}#this will rase atype error

TypeError: unhashable type: 'dict'
```

Next steps: Explain error

Start coding or generate with AI.