Day_03 Datatypes(Sets & Frozen Sets & Comparison Operators & If-Else Statement & Functions & Lambda Functions)

Sets

Definition

A set is an unordered collection of items. Every set element is unique (no duplicates) and must be immutable (cannot be changed). However, a set itself is mutable. We can add or remove items from it. Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc.

Purpose

A Set is an unordered collection data type that is iterable, mutable and has no duplicate elements. The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set.

Importance

Because sets cannot have multiple occurrences of the same element, it makes sets highly useful to efficiently remove duplicate values from a list or tuple and to perform common math operations like unions and intersections.

Strengths

The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set. This is based on a data structure known as a hash table. Since sets are unordered, we cannot access items using indexes like we do in lists.

Because sets cannot have multiple occurrences of the same element, it makes sets highly useful to efficiently remove duplicate values from a list or tuple and to perform common math operations like unions and intersections

They help a lot when there is a need for this uniqueness of the elements that need to be processed.

the property that defines sets is the possibility to apply the methods of intersection, union, difference and symmetric difference between them (like sets in mathematics I guess). Mind you cannot do these operations with lists or dictionaries (you have to convert them in sets beforehand).

Weakness

Since set items are not indexed, sets don't support any slicing or indexing operations.Sets do not support indexing, slicing, or other sequence-like behavior. There are currently two built-in set types, set, and frozenset. The set type is mutable - the contents can be changed using methods like add() and remove()

Example 18 :

Task:

create a simple set from {} and using set function add value to the set also use update,discard and remove function on it.

In [ ]:
```
numbersSet={3.2,3.06,0.29,0.36,1}
rollNoList
newNumbersSet=set(rollNoList)
```

In [ ]:
```
print(newNumbersSet)
```

In [ ]:
```
type(newNumbersSet)
```

In [ ]:
```
numbersSet.add(3)
```

In [ ]:
```
print(numbersSet)
```

In [ ]:
```
numbersSet.update({3,3,4})
```

In [ ]:
```
numbersSet
```

In [ ]:
```
numbersSet.update({2.2,2.3,3.7})
```

In [ ]:
```
numbersSet
```

In [ ]:
```
numbersSet.discard(1)
```

In [ ]:
```
print(numbersSet)
```

In [ ]:
```
numbersSet.remove(0.36)
```

In [ ]:
```
print(numbersSet)
```

In [ ]:
```
numbersSet.discard(1)
```

In [ ]:
```
print(numbersSet)
```

In [ ]:
```
numbersSet.remove(1)
```

Frozen Sets

Definition

Frozen set is just an immutable version of a Python set object. While elements of a set can be modified at any time, elements of the frozen set remain the same after creation. Due to this, frozen sets can be used as keys in Dictionary or as elements of another set.

Purpose

The frozenset() is an inbuilt function is Python which takes an iterable object as input and makes them immutable. Simply it freezes the iterable objects and makes them unchangeable.

In Python, frozenset is same as set except its elements are immutable. This function takes input as any iterable object and converts them into immutable object. The order of element is not guaranteed to be preserved.

Importaance

there main ability that they are immutable makes them very important as you want to have a data set that if once is is declayred no one can add something onto it.

Strengths

set cannot have mutable elements but frozen set have

Set is mutable in nature can be modified but frozen set cannot

set cannot be used as key in dictionary but frozen sets can be used

set cannot have set as element frozen set can have

frozenset can be used as dictionary key

Example 19 :

Task:

Use Frozen set as an element of set

In [ ]:
```
#frozenset as an element of set
S1 = frozenset([1,2,3,4])
S2 = {S1}
print(S2)
```

Example 20 :

Task:

Use Frozen sets as a Dictionary

In [ ]:
```
#frozenset can be used as dictionary key
S1=frozenset([1,2,3,4])
D1={S1:"1234"}
print(D1)
```

Weakness

major disadvantage of a frozenset is that since they are immutable, it means that you cannot add or remove values.

Comparison Operator & Logical AND OR NOT operator

Example 21 :

Task:

use different opertors to check different conditions and there output

In [ ]:
```
5>6
```

In [ ]:
```
6<8
```

In [ ]:
```
15>=15
```

In [ ]:
```
15>=14
```

In [ ]:
```
15<=16
```

In [ ]:
```
"Umair" == "Usama"
```

In [ ]:
```
"Umair" == "Umair"
```

In [ ]:
```
"Umair" != "Usama"
```

In [ ]:
```
(3.06<4) or ("Umair" == "Usama")
```

In [ ]:
```
("Umair"=="Umair") and (5<6)
```

If Statements

Definition

If statements are control flow statements which helps us to run a particular code only when a certain condition is satisfied. For example, you want to print a message on the screen only when a condition is true then you can use if statement to accomplish this in programming.

Purpose

Python if Statement is used for decision-making operations. It contains a body of code which runs only when the condition given in the if statement is true. ... When you want to justify one condition while the other condition is not true, then you use "if statement

Importance

If statement in Python tells the program what to do if the condition is true. In case the condition is false, the program just goes on to execute what comes after if statements

Strengths

The major advantage of if else statements is that they help us decide what to do if one of our expected output/decisoin gets changed for example if the user is logged in show him logout button else show him log in button.

They help our code by stoping the chrashing of our system if we did not get the desired result we will simply print the message on the screen instead of throwing the same wrong data and getting error.

It helps us bettor our code by applying certain conditions

Weakness

Weakness of the if else statements is that if we have so many conditions to check or so many answeres to check agains the same conditions than it will get really complicated for ourself to read the code and for the compiler to compile it.

if the number of if-else statements increase it will load the system and delayed the execution time.

Example 22 :

Task:

Use if-else statement to check if the number is even or odd

In [ ]:
```
number=39
if(number%2==0):
    print("{} is even.".format(number))
else:
    print("{} is Odd.".format(number))
```

Example 23 :

Task:

Use if statement to check if the student's age from the studentDictionary initialize above is greater than 18 or not.

In [ ]:
```
if((studentsDict['Student1']['age'])>18):
    print("{name} you can watch Horror Movie".format(name=studentsDict['Student1']['name']))
```

Loops in Python

Example 24 :

Task:

Use for loop to print Fibonachi series untill the limit.

In [ ]:
```
firstNumber=0
secondNumber=1
numberThree=0
limit=10
fibList=[0,1]
for i in range(1,limit):
    numberThree=firstNumber+secondNumber
    firstNumber=secondNumber
    secondNumber=numberThree
    fibList.append(numberThree)
```

In [ ]:
```
fibList
```

Example 25 :

Task:

Use While Loop to find out that if the string is palindrom or not

In [ ]:
```
givenString="radar"
i=0
check=False
while(i<(len(givenString)/2)):
    check=((givenString[i]==givenString[(len(givenString)-1)-i]))
    if(i!=0 and check==False):
        print("{string} is a not palindrom".format(string=givenString))
        break
    i=i+1
if(check==True):
    print("{string} is a palindrom".format(string=givenString))
```

Functions

Definition

function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

Purpose

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

importance

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

Strengths

Reducing duplication of code

Decomposing complex problems into simpler pieces

Improving clarity of the code

Reuse of code

Information hiding

Weakness

the function is the wraped piece of code that is used to perform sppecific function when needed the only weakness of it is the wrong way of using them.

Example 26 :

Task:

use functions to create a function that print the values of tuple

In [ ]:
```
def printResult():
    i=0
    while(i <(len(studentsResult))):
        print("Name:{} CGPA:{}".format(studentsResult[i][0],studentsResult[i][1]))
        i=i+1
```

In [ ]:
```
printResult()
```

Example 27 :

Task:

use functions to create a function that check if the string is palindrom or not.

In [ ]:
```
def checkPalindrom(givenString):
    i=0
    check=False
    while(i<(len(givenString)/2)):
        check=((givenString[i]==givenString[(len(givenString)-1)-i]))
        if(i!=0 and check==False):
            print("{string} is a not palindrom".format(string=givenString))
            return False
            break
        i=i+1
    if(check==True):
        print("{string} is a palindrom".format(string=givenString))
        return True
```

In [ ]:
```
checkPalindrom("radar")
```

Example 28 :

Task:

use functions to create a function to print the fibonachi series taking limit as argument

In [ ]:
```
def printFibonachi(limit):
    firstNumber=0
    secondNumber=1
    numberThree=0
    fibList=[0,1]
    for i in range(1,limit):
        numberThree=firstNumber+secondNumber
        firstNumber=secondNumber
        secondNumber=numberThree
        fibList.append(numberThree)
    print(fibList)
    return fibList
```

In [ ]:
```
printFibonachi(15)
```

Lambda Functions

Definition

a lambda function is a single-line function declared with no name, which can have any number of arguments, but it can only have one expression. Such a function is capable of behaving similarly to a regular function declared using the Python's def keyword. Often times a lambda function is passed as an argument to another function.

Purpose

We use lambda functions when we require a nameless function for a short period of time. In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as arguments). Lambda functions are used along with built-in functions like filter() , map() etc.

Importance

A lambda is part of a very important abstraction mechanism which deals with higher order functions. To get proper understanding of its value, please watch high quality lessons from Abelson and Sussman, and read the book SICP

Strengths

The code is simple and clear.

No additional variables are added.

Save Memory by declaring and initializing less variables

Weakness

Lambda expressions are a strange and unfamiliar syntax to many Python programmers.

Lambda functions themselves lack names and documentation, meaning that the only way to know what they do is to read the code.

Lambda expressions can only contain one statement, so some readable language features, such as tuple unpacking, cannot be used with them.

Lambda functions can often be replaced with existing functions in the standard library or Python built-in functions.

Example 29 :

Task:

use lambda function to create a square function and create a list of 10 numbers adn apply the lambda function of square and cube on it.

In [ ]:
```
square=lambda number:number**2
```

In [ ]:
```
squareList=[]
```

In [ ]:
```
numberList=list(range(1,10))
```

In [ ]:
```
for number in numberList:
    squareList.append(square(number))
```

In [ ]:
```
squareList
```

In [ ]:
```
[(lambda x: x**3)(x) for x in range(10)]
```

In [ ]: