28

Q1

- [ ] A Tuple is an mutable list.
- [ ] Tuples are slower when compared to lists.
- [x] Tuples can be used as keys in dictionaries.
- [x] Elements of a tuple are enclosed in parenthesis.
- [ ] Addition and deletion of elements is possible in a tuple.
- [x] It is possible to create tuples which contain mutable objects, such as lists.

Q2

Tuple10.py

```
1  l=list(map(str,input("data: ").split(",")))
2  print("list:",l)
3  t=tuple(l)
4  print("tuple:",t)
```

Q1

- [ ] A tuple can be created using the list function.
- [ ] Trying to access an element equivalent to the length of a tuple returns the last element in the tuple.
- [x] The result of a concatenation of two tuples when assigned to one of the tuple creates a new tuple.
- [x] The result of a repetition of a tuple when assigned to the same tuple will result in a new tuple.
- [ ] The membership and comparison operations on tuples result return a boolean value YES or NO.

Q2

Tuple01.py

```
l=list(map(str,input("data: ").split(",")))
print("list:",l)
t=tuple(l)
print("tuple:",t)
try:
    ele=int(input("index: "))
    print("element:",t[ele])
except:
    print("enter valid index")
```

Q3

- [ ] The output of the following code: ('ac') * 2 is ('ac', 'ac').
- [x] The output of the following code: ('ac',) * 2 is ('ac', 'ac').
- [x] (1, 2, 3) > (1, 0, 3) is True.
- [ ] t1 = tuple({1:10, 2:20}) will result in (10, 20).

Q4

Tuple02.py

```
t=tuple(map(str,input("data1: ").split(",")))
val=int(input("value: "))
print("tuple *  {0} = {1}".format(val,t*val))
t2=tuple(map(str,input("data2: ").split(",")))
print("concatenation: {0}".format(t+t2))
```

Q5

ıe tuple or

Tuple03.py

```
t=tuple(map(str,input("data: ").split(",")))
print("tuple:",t)
val=str(input("value: "))
if val in t:
    print(True)
else:
    print(False)
```

Q6

Tuplechng.py

```
mytup = ('a', 'b', 'c', 'd')
print("mytup =", mytup)

# add elements to the mytup
t=([1,2,3],)

mytup+=t
print("mytup =", mytup)
print("mytup[4][1] = 4")
mytup[4][1]=4
# write your code here
print("mytup =", mytup)
```

Q7

Tupledel.py

```
mytup = ('a', 'b', 'c', 'd', [1, 2, 3])
print("mytup =", mytup)
print("del mytup[4][2]")

# delele the element 3 from the mytup
l=list(mytup)
del l[4][2]
mytup=tuple(l)
print("mytup =", mytup)
print("del mytup[4] will give TypeError")
```

Q8

If the

Tuple4.py

```
t=tuple(map(str,input("data: ").split(",")))
ind=int(input("index: "))

if(ind+len(t)<0) or (ind>len(t)-1):
    print("enter valid index")
    exit()

val=str(input("value: "))
l=list(t)
l[ind]=val
t=tuple(l)
print("tuple:",t)
```

Q9

is not

Tuple5.py

```
t=tuple(map(str,input("data: ").split(",")))
print("tuple:",t)

i=int(input("index: "))
if(i+len(t)<0) or (i>len(t)-1):

    print("enter valid index")
    exit()


l=list(t)
l.pop(i)
t=tuple(l)
print("after removing:",t)
```

Q10

Tuple9.py

```
t1=tuple(map(str,input("data1: ").split(",")))
t2=tuple(map(str,input("data2: ").split(",")))
if(t1==t2):
    print(True)
else:
    print(False)
```

Q11

Tuple11.py

```
t=tuple(map(str,input("data: ").split(",")))
ele=str(input("element: "))

try:
    l=list(t)
    l.remove(ele)

    print("before deletion:",t)


    t=tuple(l)
    print("after deletion:",t)
except:
    print("enter existed element")
```

Q12

ements

Tuple8.py

```
t=tuple(map(str,input("data: ").split(",")))
si=int(input("start index: "))
ei=int(input("end index: "))

if(si+len(t)<0 or ei+len(t)<0 or si>len(t)-1 or ei>len(t)-1):
    print("enter valid index")
else:
    print("tuple in given range:",t[si:ei])
```

29

Q1

- [ ] A set is an ordered collection of unique items.
- [x] Members in a set are immutable.
- [ ] Elements can be added, removed or changed from the set .
- [ ] Numerical index can be used with set.

Q2

- [x] Set does not maintain the order of insertion
- [ ] Elements cannot be added to a Frozenset, but can be deleted.
- [ ] {1, 2, 3, 4}.add({90, 80}) will result in {1, 2, 3, 4, 90, 80}.
- [ ] s1 = {1, 2, 3, 4} s2 = frozenset([90, 80]) s1.add(s2) will give an error.

Q3

SetTest1.py

```
1  #Write your code here
2  d=set(map(str,input("data1: ").split(",")))
3  s=sorted(d)
4  print("sorted set:",s)
```

Q1

the

SetAdd.py

```
s=set(map(str,input("data1: ").split(",")))
ele=str(input("element: "))
s.add(ele)
sort=sorted(s)
print("sorted set after adding:",sort)

l=list(map(str,input("data2: ").split(",")))
s.update(l)
sort=sorted(s)
print("sorted set after updating:",sort)
```

Q2

) and

Setremdel.py

```
d=set(map(str,input("data1: ").split(",")))
ele=str(input("element to discard: "))

if(ele in d):
    d.discard(ele)
    sort=sorted(d)
    print("sorted set after discarding:",sort)
else:
    print("not in set")
    exit()

rem=str(input("element to remove: "))
if(rem in d):

    d.remove(rem)
    sort=sorted(d)
    print("sorted set after removing:",sort)
else:

    print("not in set")

```

Q3

SetMemb.py

```
s=set(map(str,input("data1: ").split(",")))
sort=sorted(s)
print("sorted set:",sort)
ele=str(input("element: "))
if(ele in sort):
    print("is {0} in set: {1}".format(ele,True))
else:
    print("is {0} in set: {1}".format(ele,False))
```

Q4

SetMathUnion.py

```
s1=set(map(str,input("data1: ").split(",")))
s2=set(map(str,input("data2: ").split(",")))
s3=set(map(str,input("data3: ").split(",")))

st1=sorted(s1)
st2=sorted(s2)
st3=sorted(s3)

print("set1 sorted:",st1)
print("set2 sorted:",st2)
print("set3 sorted:",st3)

u1=s1.union(s2)
us1=sorted(u1)
print("union of set1, set2:",us1)
# u2=u1.union(s3)
u2=u1 | s3
us2=sorted(u2)
print("union of set1, set2, set3:",us2)
print("set1 | set2:",us1)
print("set1 | set2| set3:",us2)

```

Q5

SetMathint.py

```
s1=set(map(str,input("data1: ").split(",")))
s2=set(map(str,input("data2: ").split(",")))

st1=sorted(s1)
st2=sorted(s2)
print("set1 sorted:",st1)
print("set2 sorted:",st2)

its=s1&s2
itsSort=sorted(its)
print("Intersection:",itsSort)
print("sorted set after (set1 & set2):",itsSort)
print("sorted set1 after (set1 &= set2):",itsSort)
print("sorted set2 after (set1 &= set2):",st2)

```

Q6

SetMathDiff.py

```
s1=set(map(str,input("data1: ").split(",")))
s2=set(map(str,input("data2: ").split(",")))

st1=sorted(s1)
st2=sorted(s2)
print("set1 sorted:",st1)
print("set2 sorted:",st2)

d1=s1.difference(s2)
ds1=sorted(d1)
print("difference using difference():",ds1)
s1.difference_update(s2)
st1=sorted(s1)
print("difference using difference_update():",st1)
print("difference using (set1 - set2):",ds1)
print("difference using (set1 -= set2):",ds1)
```

Q7

SetMathsymdiff.py

```
d1=set(map(str,input("data1: ").split(",")))
d2=set(map(str,input("data2: ").split(",")))

sd=d1^d2
sdSort=sorted(sd)
sd1=sorted(d1)
sd2=sorted(d2)

print("symmetric difference:",sdSort)
print("set1 ^ set2:",sdSort)
print("set1 sorted:",sdSort)
print("set2 sorted:",sd2)

d3=set(map(str,input("data3: ").split(",")))
d4=set(map(str,input("data4: ").split(",")))

d3^=d4
sd3=sorted(d3)
print("set3 after (set3 ^= set4):",sd3)
sd4=sorted(d4)
print("set4 after (set3 ^= set4):",sd4)


```

Q1

- [ ] x = {1, 2, 3}y = x and y = x.copy() will produce the same result.
- [ ] s1 = {1, 2}s2 = s1.add(5) will result in s2 = {1, 2, 5}
- [x] s1 = {1, 2}s1.add(5)s2 = s1.copy() will result in s1 = {1, 2, 5} and s2 = {1, 2, 5}
- [ ] You can create nested sets.
- [x] You can have frozen sets as members of a set.

Q2

SetIsSubset.py

```
1  s1=set(map(str,input("data1: ").split(",")))
2  s2=set(map(str,input("data2: ").split(",")))
3  s3=set(map(str,input("data3: ").split(",")))
4
5  print("set1 sorted:",sorted(s1))
6  print("set2 sorted:",sorted(s2))
7  print("set3 sorted:",sorted(s3))
8
9  print("is set1 subset of set2?",s1.issubset(s2))
10 print("is set2 subset of set1?",s2.issubset(s1))
11 print("is set1 subset of set3?",s1.issubset(s3))
12 print("is set3 subset of set1?",s3.issubset(s1))
13 print("is set2 subset of set3?",s2.issubset(s3))
14 print("is set3 subset of set2?",s3.issubset(s2))
```

Q3

SetDisjoint.py

```
d1=set(map(str,input("data1: ").split(",")))
d2=set(map(str,input("data2: ").split(",")))
d3=set(map(str,input("data3: ").split(",")))

print("set1 sorted:",sorted(d1))
print("set2 sorted:",sorted(d2))
print("set3 sorted:",sorted(d3))

print("is set1, set2 disjoint?",d1.isdisjoint(d2))
print("is set1, set3 disjoint?",d1.isdisjoint(d3))
print("is set2, set3 disjoint?",d2.isdisjoint(d3))
```

Q4

Sets1.py

```
d1=set(map(str,input("data1: ").split(",")))
d2=set(map(str,input("data2: ").split(",")))
d3=set(map(str,input("data3: ").split(",")))

print("engineers:",sorted(d1))
print("programmers:",sorted(d2))
print("managers:",sorted(d3))

emp=(d1.union(d2)).union(d3)
print("employees:",sorted(emp))

ele=str(input("element into engineers: "))
d1.add(ele)
print("engineers:",sorted(d1))
print("employees.issuperset of engineers:",emp.issuperset(d1))
print("let us update employees from engineers")
emp.update(d1)
print("employees.issuperset of engineers:",emp.issuperset(d1))
rem=str(input("element to remove from engineers, programmers, managers, employees: "))
d1.discard(rem)
d2.discard(rem)
d3.discard(rem)

print("engineers:",sorted(d1))
print("programmers:",sorted(d2))
print("managers:",sorted(d3))

emp=(d1.union(d2)).union(d3)
print("employees:",sorted(emp))
```

31

Q1 — 100%

- [ ] The keys in a dictionary can be mutable.
- [ ] The values in a dictionary have to be immutable.
- [x] A dictionary is collection of disordered elements and each element is in the form a Key and a Value pair.
- [ ] A tuple can be a key, if they only contain strings, numbers and lists.
- [x] The dict() function is used to construct a dictionary.

Q2 — 100%

- [ ] The elements in a dictionary can be either keys or values but not both.
- [ ] The indexing operator [] is used to access a key using the value as the parameter inside the indexing operator.
- [x] A value in a dictionary can be updated using the indexing operator along with the key.
- [x] A For loop is used to iterate the elements of a dictionary.
- [ ] Deleting an element in a dictionary only deletes the value but not the key.

Q1

- [x] In Python 3 zip() function returns an iterator of tuples true or false?
- [x] If there is no arguments zip() function returns an empty iterator.
- [ ] While creating dictionaries we need only keys.
- [x] sorted(dict.items()) function prints the arguments in sorted order.

Q2

DictCreate.py

```
# Write your code here
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))

print("list1:",l1)
print("list2:",l2)

d=dict(zip(l1,l2))

print("dictionary:",sorted(d.items()))
```

Q3

ListToDict.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
d=dict(zip(l1,l2 ))
if(len(l1)==len(l2)):
    print(sorted(d.items()))
else:
    print("length should be equal")
```

Q4

Dictdel.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
d=dict(zip(l1,l2))
k=str(input("key: "))
print("value:",d.get(k))
```

Q5

print

Dictmemb.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
d=dict(zip(l1,l2))

key=str(input("key: "))
if key in d:
    print("True")
else:
    print("False")
```

Q6

Dictiter.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))

d=dict(zip(l1,l2))

for i,j in sorted(d.items()):
    print(i,j)
```

Q7

AssignExample3.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
d=dict(zip(l1,l2))
for i,j in sorted(d.items()):
    print(i,"->",j)

```

Q8

d.

DictremDelPop.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))

d=dict(zip(l1,l2))

key=str(input("key: "))

if key in d.keys():
    print("value:",d[key])
    d.pop(key)
    print("dictionary:",sorted(d.items()))
else:
    print("key does not exist")
```

Q9

MaxMinOfDict.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
# d=dict(zip(l1,l2))
# l=list(d.values())
l2.sort()
print("max:",l2[-1])
print("min:",l2[0])
```

Q10

Dictupd.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
d=dict(zip(l1,l2))

key=str(input("key: "))
if key in d.keys():
    val=str(input("value: "))
    d[key]=val
    print("sorted dictionary:",sorted(d.items()))
else:
    print("key does not exist")
```

32

Q1

Dictfuncs.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
d=dict(zip(l1,l2))
print("all(dict1):",all(d))
print("any(dict1):",any(d))
print("len(dict1):",len(d))
sl=sorted(d)
print("sorted(dict1):",sl)
print("key,value of dictionary: ")
for i,j in sorted(d.items()):
    print("{0}:{1}".format(i,j))

```

Q2

DictReverse.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
d1=dict(zip(l1,l2))
d2=dict(zip(l2,l1))
print("before exchange:",sorted(d1.items()))
print("after exchange:",sorted(d2.items()))
```

Q3

put.

DictTuple.py

```

troupe = {('Cleese', 'John') : [1, 2, 3],
          ('Chapman', 'Graham') : [4, 5, 6],
          ('Idle', 'Eric') : [7, 8, 9],
          ('Jones', 'Terry') : [10, 11, 12],
          ('Gilliam', 'Terry') : [13, 14, 15, 16, 17, 18],
          ('Palin', 'Michael') : [19, 20]}

sk=sorted(troupe.items())
for i,j in sk:
    print(i[1],i[0],j)
```

Q4

DictValAdd.py

```python
#Program to combine two dictionaries
data1 = input("Enter integer elements separated by ,(comma) for keys of dict1: ")
data2 = input("Enter integer elements separated by ,(comma) for values of dict1: ")
list1 = data1.split(",")
list2 = data2.split(",")
for i in range(len(list1)):
    list1[i] = int(list1[i])

for i in range(len(list2)):
    list2[i] = int(list2[i])
dict1 = dict(zip(list1, list2))

data3 = input("Enter integer elements separated by ,(comma) for keys of dict2: ")
data4 = input("Enter integer elements separated by ,(comma) for values of dict2: ")
list3 = data3.split(",")
list4 = data4.split(",")
for i in range(len(list3)):
    list3[i] = int(list3[i])

for i in range(len(list4)):
    list4[i] = int(list4[i])
dict2 = dict(zip(list3, list4))

dict3 = {}
'''
Write your code here to complete your solution.

'''
# dict3=dict1+dict2
for i in dict1:
    if i in dict2:
        dict2[i]=dict2[i]+dict1[i]
    else:
        pass


dict3.update(dict1)
dict3.update(dict2)
print(sorted(dict3.items()))


```

Q5

DictSeq.py

```python
l=list(map(int,input("seq: ").split(",")))
d=dict()
for i in l:
    d[i]=l.count(i)
print("sorted dictionary:",sorted(d.items()))
```

Q6

DictKey.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
l3=list(map(str,input("data3: ").split(",")))
l4=list(map(str,input("data4: ").split(",")))

d1=dict(zip(l1,l2))
d2=dict(zip(l3,l4))
key=str(input("key: "))

check1=False
check2=False
if key in d1:
    check1=True

if key in d2:
    check2=True
if(check1==True and check2==True):
    print("key present in both dictionaries")
    exit()
if(check1==True):
    print("key present in first dictionary")
    exit()
if(check2==True):
    print("key present in second dictionary")
else:
    print("key is not present")
```

Q7

Histogram.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))

d=dict(zip(l1,l2))
l=sorted(d.items())
print("dictionary with key order")
for i in sorted(d.keys()):
    print(i,d[i])

print("dictionary with value order")
for i,j in sorted(d.items(),key=lambda item:item[1]):
    print(j,i)
```

Q8

DictConcate.py

```
l1=list(map(int,input("data1: ").split(",")))
l2=list(map(int,input("data2: ").split(",")))
l3=list(map(int,input("data3: ").split(",")))
l4=list(map(int,input("data4: ").split(",")))

d1=dict(zip(l1,l2))
d2=dict(zip(l3,l4))

for i in d1:
    if i in d2:
        d2[i]=d1[i]+d2[i]

d1.update(d2)
print("concatenation:",sorted(d1.items()))
```

Q9

Dictcommonkey.py

```
l1=list(map(int,input("data1: ").split(",")))
l2=list(map(int,input("data2: ").split(",")))
l3=list(map(int,input("data3: ").split(",")))
l4=list(map(int,input("data4: ").split(",")))

d1=dict(zip(l1,l2))
d2=dict(zip(l3,l4))

d3=dict()

for i in d1:
    if i in d2:
        d3[i]=d1[i]+d2[i]
print("common keys:",sorted(d3.items()))
```

33

Q1

Dictmethods.py

```
d1=list(map(str,input("data1: ").split(",")))
d2=list(map(str,input("data2: ").split(",")))
d3=list(map(str,input("data3: ").split(",")))
d4=list(map(str,input("data4: ").split(",")))

sd=dict(zip(d1,d2))
sd2=dict(zip(d3,d4))
print("sorted dictionary:",sorted(sd.items()))
print("copy of sorted dictionary:",sorted(sd.items()))
print("sorted keys of dictionary:",sorted(sd.keys()))
print("sorted values of dictionary:",sorted(sd.values()))
sd.update(sd2)
print("sorted dictionary after updation:",sorted(sd.items()))
```

Q2

Content to be reproduced

Lists2Dict.py

```
dct = {'1':'apple', '2':'orange', '3':'mango', '4':'banana'}
print("dct_keys = dct.keys()")
dct_keys = dct.keys()
print("dct_values = dct.values()")
dct_values = dct.values()
print("dct_keys_list = list(dct_keys)")
dct_keys_list = list(dct_keys)
print("dct_values_list = list(dct_values)")
dct_values_list = list(dct_values)
print("dct_keys_list =", sorted(dct_keys_list))
print("dct_values_list =", sorted(dct_values_list))
print("Combining both the keys and values list using zip");
print("zip(dct_keys_list, dct_values_list)")
zip_result = zip(dct_keys_list, dct_values_list)
print("Creating a list out of the zip result using list function")
zip_result_list = list(zip_result)
print("zip_result_list =", sorted(zip_result_list))
print("Finally creating the dictionary from the zip result list")
dict2 = dict(zip_result_list)
print("dict2 =", sorted(dict2.items()))
```

Type here

Lists2Dict.py

```
dct = {'1':'apple', '2':'orange', '3':'mango', '4':'banana'}
print("dct_keys = dct.keys()")
dct_keys = dct.keys()
print("dct_values = dct.values()")
dct_values = dct.values()
print("dct_keys_list = list(dct_keys)")
dct_keys_list = list(dct_keys)
print("dct_values_list = list(dct_values)")
dct_values_list = list(dct_values)
print("dct_keys_list =", sorted(dct_keys_list))
print("dct_values_list =", sorted(dct_values_list))
print("Combining both the keys and values list using zip");
print("zip(dct_keys_list, dct_values_list)")
zip_result = zip(dct_keys_list, dct_values_list)
print("Creating a list out of the zip result using list function")
zip_result_list = list(zip_result)
print("zip_result_list =", sorted(zip_result_list))
print("Finally creating the dictionary from the zip result list")
dict2 = dict(zip_result_list)
print("dict2 =", sorted(dict2.items()))
```

Q3

Dict2Lists.py

```
l1=list(map(str,input("data1: ").split(",")))
l2=list(map(str,input("data2: ").split(",")))
d=dict(zip(l1,l2))
print("elements:",sorted(d.items()))
print("sorted keys:",sorted(d.keys()))
print("sorted values:",sorted(d.values()))

```

34

Q1

ListCompre1.py

```
s=str(input("str: "))
l=[i for i in s]
print(l)
```

Q2

NestedCompre2.py

```
l=[i for i in range(0,50) if (i%2==0 and i%3==0)]
print(l)
```

Q3

Mattran.py

```
matrix = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]

# write your code here
l=[[matrix[j][i] for j in range(len(matrix))] for i in range(len(matrix[0]))]
print("matrix:")
print(matrix)
print("transposition using nested while:")
print(l)
print("transposition using nested for:")
print(l)
print("transposition single list comprehension:")
print(l)
print("transposition double list comprehension:")
print(l)
```

Q4

nple.

ListcompuserInput.py

```
l=list(map(int,input("data: ").split()))
print("contents:",l)
```

35

Q1

SetComprehension1.py

```
n1=int(input("n1: "))
n2=int(input("n2: "))

s=set()
i=n1
cnt=0
while(cnt<n2-n1):
    if(i%2==0):
        s.add(i**2)
    else:
        s.add(i**3)

    i+=1
    cnt+=1
print(sorted(s))

```

Q2

NestedSetComp.py

```
py={(z,y,x) for x in range(1,31) for y in range(x,31) for z in range(y,31) if x**2+y**2==z**2}
print(sorted(py))

```

36

Q1

DictCompre1.py

```

d1={i:chr(i) for i in range(ord("A"),ord("Z")+1)}
d2={chr(i):i for i in range(ord("A"),ord("Z")+1)}
print(sorted(d1.items()))
print(sorted(d2.items()))
```

alue.

37

Q1

Round.py

```
1  import math
2  num = float(input("num: "))
3  if num - int(num) >=.5:
4      print ("result:", math.ceil(num))
5  else:
6      print("result:", math.trunc(num))
```

Q1

- [x] **random()** function returns random values between 0.0 and 1.0 only.
- [ ] If we are not specifying **step** in **randrange()**, interpreter raises an Error.
- [ ] using **shuffle()** function we can reshuffle the tuple.

38

Q1

String1.py

```
1  s=str(input("str: "))
2  print(s)
```

Q1

String1.py

```
1  str1 = "This is my first String"
2  # print the string
3  # print Letter 'f' using Positive Index
4  # print Letter 'S' using Negative Index.
5  print(str1)
6  print(str1[11])
7  print(str1[-6])
```

Q2

String2.py

```
1  str = "How are you?"
2  print("String is", str)
3  # print 'are' in String using Slicing with Positive Index
4  print("are")
5  # print 'w a' in String using Slicing with Positive Index
6  print("w a")
7  # print 'you' in String using Slicing with Negative Index
8  print("you")
9  # print 'uoy' in the string using slicing and Negative indexes
10 print("uoy")
11 # print 'you?' in String using Slicing with Negative Index
12 print("you?")
```

Q3

f the

StringTest9.py

```
s=str(input("input: "))
if(len(s)<3):
    print("output:",s)
else:
    print("output:",s[0:2],end="")
    print(s[len(s)-2:])
```

Q3

the

StringTest9.py

```
s=str(input("input: "))
if(len(s)<3):
    print("output:",s)
else:
    print("output:",s[0:2],end="")
    print(s[len(s)-2:])
```

Q4

to the

StringTest8.py

```
s=str(input("str: "))

l=s[1:]
l=s[1:len(l)]
print("output:",l)
# ans=s.lstrip(s[0]).rstrip(s[-1])
# print("output:",ans)
```

Q5

StringTest6.py

```
s=str(input("str: "))
temp=""
sw=""
if(len(s)>1):
    st=s[0]
    e=s[-1]
    sw=e+s[1:-1]+st
    # s=sw

if(len(s)==1):
    print(s)
elif(len(s)==0):
    print("null")
else:
    print("output:",sw)
```

Q6

StrinTest4.py

```
s1=str(input("str1: "))
s2=str(input("str2: "))

res=s1+s2+s2+s1
print("result:",res)
```

Q7

StringTest5.py

```
s=str(input("str: "))
n=int(input("num: "))

if(len(s)<=n or n<0):
    print("num should be positive, less than the length of str")
else:

    r=s[:n]+s[n+1:]
    print("output:",r)
```

Q8

String3.py

```
1  s1=str(input("str1: "))
2  s2=str(input("str2: "))
3  print(s1,s2)
```

as

Q9

StringTest10.py

```
1  s1=str(input("str1: "))
2  s2=str(input("str2: "))
3  if(len(s1)==1 and len(s2)==1):
4      print("null")
5      exit()
6
7  l1=s1[1:]
8  l2=s2[1:len(s2)]
9
10 print("output:",l1+l2)
```

ı, and

Q10

- [ ] str = "Coding", r in str, returns **True**
- [ ] 's' in str, returns **True**
- [x] 'z' not in str, returns **True**

Q11

String4.py

```
1  s=str(input("str: "))
2  print(s*4)
3  r=s[::-1]
4  print(r*3)
```

Q12

StrinTest2.py

```
s=str(input("str: "))
if(len(s)<3):
    print("result:",s)
else:
    t=s[:3]
    print("result:",t*3)
```

Q13

tring

StringTest3.py

```
s=str(input("str: "))
n=int(input("num: "))
print("result:",s*n)
```

Q14

ks like

StringTest7.py

```
s=str(input("str: "))
n=int(input("num: "))
if(n<0):
    print("num should be positive, less than length of str")
else:
    a=s[:n]
    print("result:",a*n)
```

Q15

eated.

- [x] Strings are **Immutable** (we can't assign or change the value)
- [ ] We can perform deletion of part of a String using **delete()**.
- [ ] Deletion of individual characters of a string is possible based on index.

39

Q1

- [x] In **Python**, we can convert strings of Uppercase letters into lower cas
- [x] print('python is simple'.title()) gives output like **Python** Is Simple.
- [ ] a = "python is my favourite" a[3] = 'l' , replaces 'l' with 'h'.
- [ ] a = "python" + 3.7 returns output as **python3.7**.
- [ ] print('abcpyxyzpython', 3, 10) gives output as error.
- [x] print('20.3'.isnumeric()) it returns output as **False**.

Q2

StringEx1.py

```
1  str1 = input("str: ")
2  # make string str1 into all upper case letters.
3  print(str1.upper())
4  # make string str1 into only every word of first lette
5  print(str1.title())
6  # split every word of a string str1 with space.
7  print(str1.split())
8  # fill str1 with '%' special character 25 width
9  print(str1.center(25,'%'))
10 # make string str1 into small letters.
11 print(str1.lower())
12
13 str2 = '@'
14 # join string str2 with str1
15 print(str2.join(str1))
16 # replace a word 'Strings' with 'Tuples'.
17 print(str1.replace("Strings","Tuples"))
```

Q3

- [ ] str = "PYTHOn" print(str.isupper()) returns output as **True**.
- [x] print("\nhello") prints the output hello on a new line.
- [x] str = '123', the below code is correct to convert string into int. pri
- [x] str = "hello world", print(str.isalnum()) returns **False**.

Q4

StringEx1.py

```
1  s1=str(input("str1: "))
2  s2=str(input("str2: "))
3  print(s1*3+s2)
```

Q5

100%

Submit

StringEscape.py

```
1  print("Python provides built-in libraries\nUsing Python we can implement more and more applications\n\tPython is Robust")
```

Q6

shorter

StringTest13.py

```
1   s1=str(input("str1: "))
2   s2=str(input("str2: "))
3
4   l1=len(s1)
5   l2=len(s2)
6
7   if(l1==l2):
8       print("strings are same length")
9   elif(l1<l2):
10      print(s1+s2+s1)
11  else:
12      print(s2+s1+s2)
```

Q7

StringEx3.py

```
s=str(input("str: "))
if(s.startswith("Python")):
    print("valid")
else:
    print("invalid")
print("character with min val:",min(s))
print("character with max val:",max(s))
```

Q8

StringTest1.py

```
# Note: Python and python are different.
s=str(input("str: "))
if(s.startswith("Python")):
    print("str is:",s)
else:
    print("str after adding 'Python':","Python "+s)
```

Q9

StringEx4.py

```
s=str(input("str: "))
print(s[::-1])
```

Q10

StringTest23.py

Successfully saved.

```
#Program to remove punctuation marks from a string
import string
punctuations = string.punctuation
result = " "
str = "List - []\n tuple - ()\n Dictionary - {}\n Comment - #\n Multiply - *\n not - !\n and

#write your code here for removing punctuation
for i in str:
    if i not in punctuations:
        result+=i


print("Set of punctuations in string.punctuation is:",punctuations ) # print punctuations

print("String after removing all Punctuation's is:", result) # print result here
```

Q11

StringTest11.py

```
s=str(input("str1: "))
s1=str(input("str2: "))
print("count:",s.count(s1))
```

Q12

StringTest12.py

```
# write a program to print the
s=str(input("str: "))

res=" "
for i in s:
    res+=i*2
print("result:",res)
```

Q13

StringTest19.py

```
s=str(input("str: "))
l=len(s)

if(l%2==0):
    print("first half str of even length:",s[:l//2])
else:
    print("second half str of odd length:",s[l//2+1:])
```

Q14

StringTest20.py

```
#Write your code here
s=str(input("str: "))
l=len(s)
if(l==1):
    print(s)
if(l==0):
    print("null")
print("first:",s[::2])
print("second:",s[1::2])
print("original:",s)
```

Q15

StringTest21.py

```
# Write your code here
s=str(input("str: "))
res=""
k=0
l=len(s)
while(k<=l):
    res+=s[:k]
    k+=1

print("incremental order:",res)
```

Q16

StringTest22.py

```
# Write your code here
s=str(input("str with hyphens: "))
for i in s.split('-'):
    print(i,end="")
```

Q17

Stringprint.py

```
import string

s='a'
print("Character\t ASCII Code")
while(s<='z'):
    print(s,"\t\t",ord(s))
    s=chr(ord(s)+1)
s="A"
while(s<='Z'):
    print(s,"\t\t",ord(s))
    s=chr(ord(s)+1)
```

Q18

string.

CharCount.py

```
import string
s=str(input("str: "))
sort=''.join(dict.fromkeys(sorted(s)))
l=[]
for i in sort:
    print("'{0}'\t{1}".format(i,s.count(i)))
    l.append(i)
print(l)
```

Q19

CountDigit.py

```
s=str(input("str: "))
l=[]
for i in s:
    if i not in l:
        l.append(i)
        print("{0} \t {1}".format(i,s.count(i)))

```

40

Q1

- [x] A function is defined once but called many number of times.
- [x] Function is a block of statements to do one or more tasks.
- [ ] By writing different functions in a program, increases the length of the code.

Q2

- [x] Th keyword **def** always occurs as the start of function header.
- [x] The **function_name** follows the same rules as any identifiers of **Python**.
- [ ] Every function should have a **return statement** which returns some value.

Q3

Func_Ex.py

```
1  def helloworld():
2
3      # write your code here
4      print("Hello World\nGood morning\nHave a nice day\nThe function ends")
5
6  helloworld()
```

Q4

Func_Ex3.py

```python
def add(x, y):
    # add x, y and print the result
    return x+y

def sub(x, y):
    # subtract x, y and print the result
    return x-y

def mul(x, y):
    # multiply x, y and print the result
    return x*y

# take inputs x and y from the user
# call the functions add,sub, mul
x=int(input("x: "))
y=int(input("y: "))
print(add(x,y))
print(sub(x,y))
print(mul(x,y))
```

Q5

Func_Ex4.py

```python
# write your code here
a=int(input("a: "))
b=int(input("b: "))
# i know
print(a+b)
print(a-b)
print(a*b)

```

Q6

- [x] A comment that occurs in the first line of the function body after the colon(:) is known as **Docstring**

Correct!

- [x] This docstring is available in the program as a **__doc__** attribute.

Correct!

- [ ] A docstring should have only **one** line.

Incorrect, Docstring can have **multiple** lines.

- [x] A docstring is written between **triple quotes** """

Correct!

Q7

of

Func_Ex13.py

```
a=int(input("a: "))
b=int(input("b: "))
# i know
print("sum, average: ({0}, {1})".format(a+b,(a+b)/2))
print("subtraction:",a-b)
print("multiplication:",a*b)
```

Q8

Func_Ex14.py

```
a=int(input("a: "))
b=int(input("b: "))
print("addition:",a+b)
print("subtraction:",a-b)
```