Assignment 5

Q.no - 1 The dictionary data structure consists of key-value data pair.

Examples

```
>>> a = {} # empty dictionary
>>>type(a)
< type 'dict' >
>>> book = {"Author":"Lewis Carroll"}
>>> book["Title"] = "Alice's Adventures in Wonderland"
>>> book
{'Title': "Alice's Adventures in Wonderland", 'Author': 'Lewis Carroll'}
>>> contactinfo["Tom"]
{'Email':'tom@gmail.com', 'Phone':61234567}
>>> contactinfo["Sally"]
{'Email':'sally@hotmail.com', 'Phone':67654321}
```

Ans - # Initialize dictionary "contactinfo" with the values
as shown in above examples. Hint: The key is a string
literal while the value is a dictionary type.
contactinfo
={"Tom":{'Email':'tom@gmail.com','Phone':61234567},"Sally":{'Email':'sally@hotmail.com','Phone':67654321}}

Q.no - 2 Dictionary consists of key-value data pair. The key must be of immutable type, like string and tuple.

Examples

```
>>> book = {"Author":"Lewis Carroll"}
>>> book["Title"] = "Alice's Adventures in Wonderland"
>>> book
{'Title': "Alice's Adventures in Wonderland", 'Author': 'Lewis Carroll'}
```

Question:
[MCQ] Which of the following declarations is not valid for 'dict' type?
1) d = {"Name": "Tom" }
2) d = { (1,3,4): 4.5 }
3) d = { ["First", "Last"]: (1,3) }
4) d = { 1: 0.4 }
5) None of the above

Ans - 3

Q.no - 3 Dictionary consists of key-value data pair. Python provides a few ways to add/update key-value pair.

Question:

[MCQ] Suppose "d" is an empty dictionary, which statement does not assign "d" with {"Name":"Tom"}?

1) d = {"Name": "Tom" }
2) d["Name"] = "Tom"
3) d.update({"Name": "Tom" })
4) d.setdefault("Name", "Tom")
5) None of the above.

Ans - 5

Q.no - 4 Dictionary consists of key-value data pair. Python provides a few ways to retrieve key and/or value.

Question:

[MCQ] d = {"a":1, "b":2}. Which of the statements returns [1,2]?

1) d.keys()
2) d.values()
3) d.items()
4) d.popitem()
5) None of the above.

Ans - 2

Q.no - 5 In gene expression, mRNA is transcribed from a DNA template. The 4 nucleotide bases of A, T, C, G corresponds to the U, A, G, C bases of the mRNA. Write a function that returns the mRNA transcript given the sequence of a DNA strand.

Examples

```
>>> mRNAtranscription("ATCGATTG")
"UAGCUAAC"
```

Ans - # Use a dictionary to provide the mapping of DNA to RNA bases.

```
def mRNAtranscription(dna_template):
    dna2rna = { }
    mRNA = ''
    lst=[]
    for base in dna_template:
        dna2rna={'A':'U','T':'A','C':'G','G':'C'}
        mrna = dna2rna.get(base)
        lst.append(mrna)

    return ''.join(lst)
```

Q.no - 6 A DNA strand consisting of the 4 nucleotide bases is usually represented with a string of letters: A,T, C, G. Write a function that computes the base composition of a given DNA sequence.

Examples

```
    >>> baseComposition("CTATCGGCACCCTTTCAGCA")
     {'A': 4, 'C': 8, 'T': 5,  'G': 3 }
     >>> baseComposition("AGT")
     {'A': 1, 'C': 0, 'T': 1,  'G': 1 }
```

Ans -
```
def baseComposition(dna_seq):
    base_dict={'A':0,'C':0,'T':0,'G':0}
    for x in dna_seq:
        if x in 'A':
            base_dict['A']+=1
        if x in 'C':
            base_dict['C']+=1
        if x in 'T':
            base_dict['T']+=1
        if x in 'G':
            base_dict['G']+=1
    return base_dict
```

Q.no - 7 Write a function countLetters(word) that takes in a word as argument and returns a dictionary that counts the number of times each letter appears.

Examples

```
    >>> countLetters('google')
     {'e': 1, 'g': 2, 'l': 1, 'o': 2}
     >>> countLetters('apple')
     {'a': 1, 'e': 1, 'l': 1, 'p': 2}
     >>> countLetters('')
     {}
```

Ans - def countLetters(word):
```
    store={}
    for x in word:
        store[x]=word.count(x)
    return store
```

Q.no - 8 Write a function reverseLookup(dictionary, value) that takes in a dictionary and a value as arguments and returns a sorted list of all keys that contains the value. The function will return an empty list if no match is found.

Examples

```
>>> reverseLookup({'a':1, 'b':2, 'c':2}, 1)
['a']
>>> reverseLookup({'a':1, 'b':2, 'c':2}, 2)
['b', 'c']
>>> reverseLookup({'a':1, 'b':2, 'c':2}, 3)
[]
```

Ans - def reverseLookup(dictionary, value):
```
    lst=[]
    for k,v in dictionary.items():
        if v == value:
            lst.append(k)
    return sorted(lst)
```

Q.no - 9 Write a function invertDictionary(d) that takes in a dictionary as argument and return a dictionary that inverts the keys and the values of the original dictionary.

Examples

```
>>> invertDictionary({'a':1, 'b':2, 'c':3, 'd':2})
{1: ['a'], 2: ['b', 'd'], 3: ['c']}
>>> invertDictionary({'a':3, 'b':3, 'c':3})
{3: ['a', 'c', 'b']}
>>> invertDictionary({'a':2, 'b':1, 'c':2, 'd':1})
{1: ['b', 'd'], 2: ['a', 'c']}
```

Ans - def invertDictionary(d):
```
    rdict = {}
    for k, v in d.iteritems():
        rdict.setdefault(v, []).append(k)
    return rdict
```

Q.no - 10 A sparse vector is a vector whose entries are almost all zero, like [1, 0, 0, 0, 0, 0, 0, 2, 0]. Storing all those zeros wastes memory and dictionaries are commonly used to keep track of just the nonzero entries. For example, the vector shown earlier can be represented as {0:1, 7:2}, since the vector it is meant to represent has the value 1 at index 0 and the value 2 at index 7. Write a function that converts a sparse vector into a dictionary as described above.

Examples

```
>>> convertVector([1, 0, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 4])
 {0: 1, 3: 2, 7: 3, 12: 4}
 >>> convertVector([1, 0, 1 , 0, 2, 0, 1, 0, 0, 1, 0])
 {0: 1, 2: 1, 4: 2, 6: 1, 9: 1}
 >>> convertVector([0, 0, 0, 0, 0])
 {}
```

Ans - def convertVector(numbers):
    sparse_dict = {}
    for k, v in enumerate(numbers):
        if v:
            sparse_dict[k] = v
    return sparse_dict

Q.no - 11 A sparse vector is a vector whose entries are almost all zero, like [1, 0, 0, 0, 0, 0, 0, 2, 0]. Storing all those zeros wastes memory and dictionaries are commonly used to keep track of just the nonzero entries. For example, the vector shown earlier can be represented as {0:1, 7:2}, since the vector it is meant to represent has the value 1 at index 0 and the value 2 at index 7. Write a function that converts a dictionary back to its sparese vector representation.

Examples

```
>>> convertDictionary({0: 1, 3: 2, 7: 3, 12: 4})
 [1, 0, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 4]
 >>> convertDictionary({0: 1, 2: 1, 4: 2, 6: 1, 9: 1})
 [1, 0, 1, 0, 2, 0, 1, 0, 0, 1]
 >>> convertDictionary({})
 []
```

Ans - def convertDictionary(d,default=0):
    if d:
        maxElem = max(d)
        return [d.get(x, default) for x in range(maxElem+1)]
    else:
     return []