In [9]:
```
# Initialize a dictionary "emp_info" with below details
# In - emp_info['Tom']
# Out - {'email':'tom_latham019@gmail.com', 'Phone': +1987654321, 'City': 'C

# In - emp_info['Kathy']
# Out - {'email':'kathy_abram897@gmail.com', 'Phone': +1887654321, 'City': '

emp_info={'Tom':{'email':'tom_latham019@gmail.com', 'Phone': +1987654321, 'C
emp_info['Tom']
emp_info['Kathy']
```
executed in 22ms, finished 10:12:28 2021-01-13

Out[9]:
```
{'email': 'kathy_abram897@gmail.com', 'Phone': 1887654321, 'City': 'New York'}
```

In [4]:
```
# Create a dictionary out of below inputs
# lst1 = ['emp1', 'emp2', 'emp3']
# emp_key = ['e_name', 'e_id', 'e_sal']
# emp1_val = ['John', 'SG101', '$10,000']
# emp2_val = ['Smith', 'SG102', '$9,000']
# emp3_val = ['Peter', 'SG103', '$9,500']

# Expected Output:- {'emp1':{'e_name':'John', 'e_id':'SG101', 'e_sal':$10,00
#                    'emp2':{'e_name':'Smith', 'e_id':'SG102', 'e_sal':$9,00
#                    'emp3':{'e_name':'Peter', 'e_id':'SG103', 'e_sal':$9,50

lst1 = ['emp1', 'emp2', 'emp3']
emp_key = ['e_name', 'e_id', 'e_sal']
emp1_val = ['John', 'SG101', '$10,000']
emp2_val = ['Smith', 'SG102', '$9,000']
emp3_val = ['Peter', 'SG103', '$9,500']

my_dict={}
my_lst=[]

emp_val=[emp1_val, emp2_val, emp3_val]

for i in range(0,len(emp_val)):
    my_lst.append(dict(zip(emp_key,emp_val[i])))

dict(zip(lst1,my_lst))
```
executed in 22ms, finished 10:07:37 2021-01-13

Out[4]:
```
{'emp1': {'e_name': 'John', 'e_id': 'SG101', 'e_sal': '$10,000'},
 'emp2': {'e_name': 'Smith', 'e_id': 'SG102', 'e_sal': '$9,000'},
 'emp3': {'e_name': 'Peter', 'e_id': 'SG103', 'e_sal': '$9,500'}}
```

In [86]:
```
# Acess the value of key 'history'

sampleDict = {
   "class":{
      "student":{
         "name":"Mike",
         "marks":{
            "physics":70,
            "history":80
         }
      }
   }
}

sampleDict['class']['student']['marks']['history']

```
executed in 9ms, finished 21:23:38 2021-01-12

Out[86]:
```
80
```

In [59]:
```
# Initialize dictionary with default values. Inputs are:-
# employees = ['Kelly', 'Emma', 'John']
# defaults = {"designation": 'Application Developer', "salary": 8000}

#Expected output:- {'Kelly': {'designation': 'Application Developer', 'salar
#                    'Emma': {'designation': 'Application Developer', 'salary
#                    'John': {'designation': 'Application Developer', 'salary

employees = ['Kelly', 'Emma', 'John']
defaults = {"designation": 'Application Developer', "salary": 8000}
my_dict={}

for i in employees:
    my_dict.update(dict({i:defaults}))
my_dict
```
executed in 16ms, finished 19:39:42 2021-01-13

Out[59]:
```
{'Kelly': {'designation': 'Application Developer', 'salary': 8000},
 'Emma': {'designation': 'Application Developer', 'salary': 8000},
 'John': {'designation': 'Application Developer', 'salary': 8000}}
```

In [55]:
```python
# 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 o
# Write a function that returns the mRNA transcript given the sequence of a 

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

nucleotide_bases=['A', 'T', 'C', 'G']

mRNA_bases=['U', 'A', 'G', 'C']
DNA_strand=''

RNA_to_mRNA=dict(zip(nucleotide_bases,mRNA_bases))

def RNA_to_mRNA_fn(DNA_strand):
    if DNA_strand in RNA_to_mRNA.keys():
        return RNA_to_mRNA[DNA_strand]
    else:
        print("DNA strand entered is invalid")

DNA_strand=str(input('Enter the DNA strand to get its mRNA transcript: '))

RNA_to_mRNA_fn(DNA_strand)

```
executed in 4.61s, finished 12:43:11 2021-01-13

```
Enter the DNA strand to get its mRNA transcript: T
```

Out[55]:
```
'A'
```

In [30]:
```python
# Write a function which takes a word as input and returns a dictionary with
# In - count_letter('google.com')
# Out - {'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}

lst=[]
my_lst=[]
my_dict={}
word=''

def count_letter(word="google.com"):
    my_lst=list(set(word))

    for i in my_lst:
        lst.append(word.count(i))

    return dict(zip(list(set(word)),lst))

count_letter()
```
executed in 14ms, finished 12:05:19 2021-01-13

Out[30]:
```
{'g': 2, 'e': 1, 'l': 1, 'm': 1, 'c': 1, '.': 1, 'o': 3}
```

In [9]:

```
# A DNA strand consisting of the 4 nucleotide bases is usually represented w
# Write a function that computes the base composition of a given DNA sequenc

# In - baseComposition("CTATCGGCACCCTTTCAGCA")
# Out - {'A': 4, 'C': 8, 'T': 5,  'G': 3 }

# In - baseComposition("AGT")
# Out - {'A': 1, 'C': 0, 'T': 1,  'G': 1 }

nucleotide_bases=['A', 'T', 'C', 'G']
Out={}
Out_sorted={}
lst2=[]

def baseComposition(DNA_sequence):
    my_lst=list(set(DNA_sequence))

    for i in my_lst:
        Out.update({i: DNA_sequence.count(i)})
    return Out


DNA_sequence=str(input('Enter the DNA sequence: '))

Out_unsorted=baseComposition(DNA_sequence)

lst2=['A','C','T','G']

for j in lst2:
    if j in Out_unsorted.keys():
        Out_sorted.update({j: Out_unsorted[j]})
    else:
        Out_sorted.update({j: 0})

print(Out_sorted)

```

executed in 3.15s, finished 15:58:22 2021-01-13

```
Enter the DNA sequence: AGT
{'A': 1, 'C': 0, 'T': 1, 'G': 1}
```

In [20]:

```
# [MCQ] Suppose "d" is an empty dictionary, which statement does not assign
# 1. d = {"Name": "Tom" }
# 2. d["Name"] = "Tom"
# 3. d.update({"Name": "Tom" })
# 4. d.setdefault("Name", "Tom")
# 5. None of the above.
#5 is the answer
```

executed in 5ms, finished 11:28:05 2021-01-13

In [6]:
```
# [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.

 # Answer is 2. d.values()
```
executed in 4ms, finished 20:03:42 2021-01-12

In [23]:
```
# [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

# Answer (3.) Both lists and dictionaries can't work as a list for a diction
# are immutable.

```
executed in 5ms, finished 11:34:57 2021-01-13

In [42]:
```
# Write a function reverseLookup(dictionary, value) that takes in a dictiona
# and a value as arguments and returns a sorted list of all keys that contai
# The function will return an empty list if no match is found.

# In - reverseLookup({'a':1, 'b':2, 'c':2}, 1)
# Out - ['a']
# In - reverseLookup({'a':1, 'b':2, 'c':2}, 2)
# Out - ['b', 'c']
# In - reverseLookup({'a':1, 'b':2, 'c':2}, 3)
# Out - []

user_value=eval(input('Enter a value you wish to read the key of: '))
my_dict={'a':1, 'b':2, 'c':2}
my_list=[]

def reverseLookup(my_dict , user_value):
    return [k for k in my_dict.keys() if my_dict[k]==user_value]

reverseLookup(my_dict,user_value)

```
executed in 1.91s, finished 16:57:31 2021-01-13

```
Enter a value you wish to read the key of: 2
```

Out[42]:
```
['b', 'c']
```

In [42]:
```
# Write a function invertDictionary(d) that takes in a dictionary as argumen
# In - invertDictionary({'a':1, 'b':2, 'c':3, 'd':2})
# Out - {1: ['a'], 2: ['b', 'd'], 3: ['c']}
# In - invertDictionary({'a':3, 'b':3, 'c':3})
# Out - {3: ['a', 'c', 'b']}
# In - invertDictionary({'a':2, 'b':1, 'c':2, 'd':1})
# Out - {1: ['b', 'd'], 2: ['a', 'c']}

lst=[]
Output=[]
Output_list=[]
my_dict={}

def invertDictionary(my_dict):
    a=list(set(list(my_dict.values())))
    for i in a:
        for k in my_dict.keys():
            if my_dict[k]==i:
                lst.append(k)
        Output.append(list(lst))
        lst.clear()
    return Output,a

Output_list,a=invertDictionary({'a':2, 'b':1, 'c':2, 'd':1})

print(dict(zip(a,Output_list)))

```
executed in 12ms, finished 00:00:19 2021-01-14

```
{1: ['b', 'd'], 2: ['a', 'c']}
```

In [57]:
```
# Write a function that converts a sparse vector into a dictionary as descri
# In - convertVector([1, 0, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 4])
# Out - {0: 1, 3: 2, 7: 3, 12: 4}
# In - convertVector([1, 0, 1 , 0, 2, 0, 1, 0, 0, 1, 0])
# Out - {0: 1, 2: 1, 4: 2, 6: 1, 9: 1}
# In - convertVector([0, 0, 0, 0, 0])
# Out - {}

my_dict={}

my_lst=[]

def convertVector(my_lst):
    j=0
    for i in my_lst:
        if i!=0:
            my_dict.update({j:i})
        j+=1

    return my_dict

convertVector([1, 0, 1 , 0, 2, 0, 1, 0, 0, 1, 0])
```
executed in 18ms, finished 01:02:25 2021-01-14

Out[57]:
```
{0: 1, 2: 1, 4: 2, 6: 1, 9: 1}
```

In [111]:
```
# Write a function that converts a dictionary back to its sparse vector repr
# In - convertDictionary({0: 1, 3: 2, 7: 3, 12: 4})
# Out - [1, 0, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 4]
# In - convertDictionary({0: 1, 2: 1, 4: 2, 6: 1, 9: 1})
# Out - [1, 0, 1, 0, 2, 0, 1, 0, 0, 1]
# In - convertDictionary({})
# Out - []

my_dict={}

my_lst=[]

def convertDictionary(my_dict):
    k=list(my_dict.keys())
    for i in range(0,k[-1]+1):
        if i not in my_dict.keys():
            my_lst.append(0)
        else:
            my_lst.append(my_dict[i])

    return my_lst

convertDictionary({0: 1, 2: 1, 4: 2, 6: 1, 9: 1})
```
executed in 13ms, finished 03:58:36 2021-01-14

Out[111]:
```
[1, 0, 1, 0, 2, 0, 1, 0, 0, 1]
```

In [ ]:
```
# Write a function that converts a dictionary back to its sparse vector repr
# In - convertDictionary({0: 1, 3: 2, 7: 3, 12: 4})
# Out - [1, 0, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 4]
# In - convertDictionary({0: 1, 2: 1, 4: 2, 6: 1, 9: 1})
# Out - [1, 0, 1, 0, 2, 0, 1, 0, 0, 1]
# In - convertDictionary({})
# Out - []

my_dict={}

my_lst=[]

def convertDictionary(my_dict):
    my_lst[k]=my_dict[k] for i in range(list(my_dict.keys())) if k


convertDictionary({0: 1, 3: 2, 7: 3, 12: 4})
```

In [13]:
```
# Given a Python dictionary, Change Brad’s salary to 8500
# sampleDict = {
#       'emp1': {'name': 'Jhon', 'salary': 7500},
#       'emp2': {'name': 'Emma', 'salary': 8000},
#       'emp3': {'name': 'Brad', 'salary': 6500}
# }

# Expected Output
# sampleDict = {
#       'emp1': {'name': 'Jhon', 'salary': 7500},
#       'emp2': {'name': 'Emma', 'salary': 8000},
#       'emp3': {'name': 'Brad', 'salary': 8500}
# }

sampleDict = {'emp1': {'name': 'Jhon', 'salary': 7500},
               'emp2': {'name': 'Emma', 'salary': 8000},
               'emp3': {'name': 'Brad', 'salary': 6500}}

sampleDict['emp3']['salary']=8500
sampleDict
```
executed in 13ms, finished 20:06:19 2021-01-12

Out[13]:
```
{'emp1': {'name': 'Jhon', 'salary': 7500},
 'emp2': {'name': 'Emma', 'salary': 8000},
 'emp3': {'name': 'Brad', 'salary': 8500}}
```

In [44]:
```python
# Get the key corresponding to the minimum value from the following dictiona
# sampleDict = {
#   'Physics': 82,
#   'Math': 65,
#   'history': 75
# }

# Expected Output
# Math

sampleDict = {
  'Physics': 82,
  'Math': 65,
  'history': 75
}

for i in sampleDict.keys():
    if sampleDict[i]==min(sampleDict.values()):
        print(i)

# [print(i) for i in sampleDict.keys() if sampleDict[i]==min(sampleDict.valu
```
executed in 8ms, finished 20:26:25 2021-01-12

```
Math
```

In [67]:
```python
# Rename key city to location in the following dictionary
# sampleDict = {
#   "name": "Kelly",
#   "age":25,
#   "salary": 8000,
#   "city": "New york"
# }

# Expected Output
# {
#   "name": "Kelly",
#   "age":25,
#   "salary": 8000,
#   "location": "New york"
# }

sampleDict = {
  "name": "Kelly",
  "age":25,
  "salary": 8000,
  "city": "New york"
}

sampleDict.pop('city')
sampleDict.update({"Location": "New york"})
print(sampleDict)
```
executed in 107ms, finished 19:48:05 2021-01-13

```
{'name': 'Kelly', 'age': 25, 'salary': 8000, 'Location': 'New york'}
```

In [77]:
```python
# Check if a value 200 exists in a dictionary
# sampleDict = {'a': 100, 'b': 200, 'c': 300}

# Expected Output: True

sampleDict = {'a': 100, 'b': 200, 'c': 300}
c=0

for i in sampleDict.values():
    if i==200:
        print(i==200)
        c+=1

if c==0:
    print("200 doesn't exist in the dictionary")

```
executed in 8ms, finished 20:56:51 2021-01-12

```
True
```

In [82]:
```python
# Delete set of keys from Python Dictionary
# sampleDict = {
#    "name": "Kelly",
#    "age":25,
#    "salary": 8000,
#    "city": "New york"

# }
# keysToRemove = ["name", "salary"]

# Expected Output:
# {'city': 'New york', 'age': 25}


sampleDict = {
  "name": "Kelly",
  "age":25,
  "salary": 8000,
  "city": "New york"
}

keysToRemove= ["name", "salary"]

for k in keysToRemove:
    sampleDict.pop(k)

sampleDict
```
executed in 9ms, finished 21:10:44 2021-01-12

Out[82]:
```
{'age': 25, 'city': 'New york'}
```