In [8]:
```
# Usage:
#      % python drop_char.py
#
# Himanshu Mohan, Nov 11, 2019
from typing import List

def drop_char(word: str) -> List[str]:
    "Return all lower case strings obtained by dropping one char"
    new_word=word.lower()
    new_list=[ new_word[0:i]+new_word[i+1:] for i in range(len(new_word
))]

    return(new_list)

drop_char('Total')
```

Out[8]:
```
['otal', 'ttal', 'toal', 'totl', 'tota']
```

In [4]:
```
# anagram.py
#
# a function that decides if two strings are anagrams
# Usage:
#      % python anagram.py
#
# Himanshu Mohan, Nov 11, 2019

def is_anagram(word1: str, word2: str) -> bool:
    "Is word2 an anagram of word1?"

    try:
        if word1.lower()==word2.lower():
            raise AssertionError

    except AssertionError:
            print("Did not expect", word1, "and", word2, "to be anagram
s")
            return False
    # Get lengths of both strings
    n1 = len(word1)
    n2 = len(word2)

    # If lenght of both strings is not same, then
    # they cannot be anagram
    if n1 != n2:
        return False

    # Sort both strings
    word1_list = sorted(word1.lower())
    #print(sorted(word1.lower()))
    word2_list = sorted(word2.lower())
    #print(sorted(word2.lower()))

    # Compare sorted strings
    for i in range(0, n1):
        if word1_list[i] != word2_list[i]:
            return 0

    return True

print(is_anagram('silent','Listen'))
```

```
True
```

In [5]:
```
# merge.py
#
# a function that takes two strings representing sorted lists,
# and returns a sorted list of the merged values
# Usage:
#       % python merge.py
#
# Himanshu Mohan, Nov 11, 2019

from typing import List
import re

def merge(s1: str, s2: str) -> List:
    """Take two strings representing sorted lists
       Return a a list holding the merged values"""

    invalid = re.compile('[^0-9]')
    ls1 = list(s1+s2)

    new_list=[]
    cleaned_ls1 = [int(i) for i in ls1 if not invalid.search(i)]

    n = len(cleaned_ls1)
    for i in range(n):
        for j in range(1,n):
            if cleaned_ls1[j-1] > cleaned_ls1[j]:
                (cleaned_ls1[j-1], cleaned_ls1[j]) = (cleaned_ls1[j], cl
eaned_ls1[j-1])

    return (cleaned_ls1)


print(merge('[4, 3, 1]', '[4, 2, 6]') )
```

```
[1, 2, 3, 4, 4, 6]
```

In [6]:
```
# Person.py
#
# Person is a class that defines a citizen with a name.
# Students and Employees are subclasses of Persons.
# Usage:
#      % python Person.py
#
# Himanshu Mohan, Nov 12, 2019

class Person:
    "People have a first and last name"

    def __init__(self, first, last):
        self.firstname = first
        self.lastname = last

    def __str__(self):
        return self.firstname + " " + self.lastname

    def __eq__(self, other):
        #return (self.firstname == other.firstname) and (self.lastname =
= other.lastname)
        return ((self.firstname).lower() == (other.firstname).lower())an
d((self.lastname).lower() == (other.lastname).lower())
    def is_employed(self):
        return False


class Student(Person):
    "Person who is a student"

    def __init__(self, first, last, school, id):
        # Call Superclass to set common information
        super().__init__(first, last)
        self.school = school
        self.id = id

    def __str__(self):
        # Call Superclass to dispaly common information
        return super().__str__() + ", " + str(self.id) + ' at ' + self.s
chool

    def __eq__(self, other):
        return super().__eq__(other) and (self.id == other.id) and (self
.school == other.school)

    def is_employed(self):
        return False

class Employee(Person):
    "Person who is employed"
    def __init__(self, first, last, company, id):
        #pass
        # Call Superclass to set common information
        super().__init__(first, last)
        self.company = company
```

```
        self.id = id

    def __str__(self):
        # Call Superclass to dispaly common information
        return super().__str__() + ", " + str(self.id) + ' at ' + self.c
ompany

    def __eq__(self, other):
        return super().__eq__(other) and (self.id == other.id) and (self
.company == other.company)

    def is_employed(self):
        return True
```

In [7]:
```
p1=Person ('John', 'Jacob')
p2=Person('john', 'jacoB')
print(p1==p2)
```

```
True
```

In [ ]:
```

```