Hyphens

Index Approach

In [20]:
```
def hyphens(s):
    hyp=''
    alps=''
    for i in range(len(s)):
        if s[i]=="-":
            hyp=hyp+s[i]
        else:
            alps=alps+s[i]
    print(hyp+alps)
hyphens("h-e-l-l-o")
hyphens("hello-hi-sir-bye")
hyphens("h-e-l-l-ohello-hibye-sir---high--and-bye-love-my-self")
```

```
----hello
---hellohisirbye
---------------hellohellohibyesirhighandbyelovemyself
```

String approach

In [21]:
```
def hyphens(s):
    hyp=''
    alps=''
    for i in s:
        if i=="-":
            hyp=hyp+i
        else:
            alps=alps+i
    print(hyp+alps)
hyphens("h-e-l-l-o")
hyphens("hello-hi-sir-bye")
hyphens("h-e-l-l-ohello-hibye-sir---high--and-bye-love-my-self")
```

```
----hello
---hellohisirbye
---------------hellohellohibyesirhighandbyelovemyself
```

N Base Notation

In [22]:
```
def notation(n,num):
    remi=[]
    while num!=0:
        rem=num%n
        remi.append(rem)
        num=num//n
    result=''
    for i in remi:
        if i>9:
            c=chr(i+55)
            result+=c
        else:
            result+=str(i)
    return result[::-1]
```

In [23]:
```
notation(12,718)
```

Out[23]: '4BA'

In [24]:
```
notation(184,778)
```

Out[24]: '4a'

In [25]:
```
notation(6,500)
```

Out[25]: '2152'

Check Password

In [26]:
```
def checkPass(pwd):
    if len(pwd)<4:
        return 0
    if pwd[0].isdigit():
        return 0
    cap=0
    num=0
    for i in pwd:
        if i.isupper():
            cap=1
        elif i.isdigit():
            num=1
        elif i==" " or i=="/":
            return 0
    if cap==1 and num==1:
        return 1
    return 0
```

In [27]:
```
checkPass("Wewee45")
```

Out[27]: 1

In [28]:
```
checkPass("4Wewee45")
```

Out[28]: 0

In [29]:
```
checkPass("We wee45")
```

Out[29]: 0

In [30]:
```
checkPass("Wew/e e45")
```

Out[30]: 0

Diff of 2 sets of numbers from n to m

In [31]:
```
def summ(n,m):
    d=0
    nd=0
    for i in range(1,m+1):
        if i%n==0:
            d+=i
        else:
            nd+=i
    return abs(nd-d)
```

In [32]:
```
summ(4,20)
```

Out[32]: 90

In [33]:
```
summ(9,100)
```

Out[33]: 3862

Large and Small Num

In [34]:
```
arr=[3,2,1,7,5,4]
arr.sort()
n=len(arr)
e=arr[0:n:2]
o=arr[1:n:2]
print(e,o)
print(e[-2]+o[-2])
```

```
[1, 3, 5] [2, 4, 7]
7
```

In [35]:
```
def arr_sum(arr):
    arr.sort()
    n=len(arr)
    e=arr[0:n:2]
    o=arr[1:n:2]
    return e[-2]+o[1]
```

In [36]:
```
arr_sum([4,1,3,6,2,11,5,5])
```

Out[36]: 9

Product of Smallest Pair

In [52]:
```
def small_pair(arr,summ):
    if len(arr)<2:
        return -1
    arr.sort()
    if arr[0]+arr[1]<summ:
        return arr[0]*arr[1]
    else:
        return 0
```

In [53]:
```
small_pair([4,1,3,7,8],3)
```

Out[53]: 0

In [59]:
```
small_pair([8,4,3,6],8)
```

Out[59]: 12

In [58]:
```
small_pair([2,6,1,7,8],4)
```

Out[58]: 2

In [56]:
```
small_pair([1],5)
```

Out[56]: -1

Absolute Difference

In [68]:
```
def absdiff(arr,num,diff):
    c=0
    for i in arr:
        if abs(num-i)<=diff:
            c+=1
    if c>0:
        return c
    else:
        return -1
```

In [69]:
```
absdiff([12,3,14,56,77,13],13,2)
```

Out[69]: 3

In [70]:
```
absdiff([9,7,13,21,5],8,2)
```

Out[70]: 2

Valid Parentheses

In [93]:
```
def validpa(p):
    dict={'(':')','{':'}','[':']'}
    stack=[]
    for i in p:
        if i in dict:
            stack.append(dict[i])
        else:
            if stack[-1]==i:
                stack.pop()
    if len(stack)==0:
        return "valid"
    else:
        return "Not valid"
validpa('[({})]')
```

Out[93]: 'valid'

In [96]:
```
def validpa(p):
    dict={'(':')','{':'}','[':']'}
    stack=[]
    for i in p:
        if i in dict:
            stack.append(dict[i])
        else:
            if stack[-1]==i:
                stack.pop()
    if len(stack)==0:
        return "valid"
    else:
        return "Not valid"
validpa('[(})]')
```

Out[96]: 'valid'

In [99]:
```
def validpa(p):
    dict = {'(': ')', '{': '}', '[': ']'}
    stack = []

    for i in p:
        if i in dict:
            stack.append(dict[i])
        else:
            if stack[-1] != i:
                return "Not valid"
            stack.pop()

    return "valid" if not stack else "Not valid"
validpa('[(})]')
```

Out[99]: 'Not valid'

In [ ]:
```

```