Date 18 June 2019

Problem Set(Hckerearth)

- Play with Numbers
- Special Number
- Highest Reminder

Concepts

- Tuples
- Dictionaries
- Packages and Modules in Python

Special Number

In [10]:
```
# Function to determine if a number is special number or not
def isspecialNumber(n,p):
    if numberPrimeFactors(n) >= p:
        return True
    return False

# Function to check if number is prime
def isPrime(n):
    flag = 1
    if n == 2:
        return True
    for i in range(2,n//2+1):
        if n%i == 0:
            flag = 0
            return False
    if flag == 1:
        return True

# Function to determine number of prime factors for a given number
def numberPrimeFactors(n):
    if isPrime(n):
        return 1
    count = 0
    for i in range(2,n//2+1):
        if isPrime(i) and n%i == 0:
            count+=1
    return count
numberPrimeFactors(30)
isspecialNumber(30,2)
```

Out[10]:
```
True
```

In [8]:
```
def solution2():
    p = int(input())
    t = int(input())
    for i in range(0,t):
        n=int(input())
        if isspecialNumber(n,p):
            print("YES")
        else:
            print("NO")
solution2()
```

```
2
3
6
YES
3
NO
5
NO
```

Highest Reminder

write a program to find a nuatural number that is smaller than N highest reminder when devided by that number. If there is more than one such number , orunt the smallest number N highest = 0

x < N and n % X == highest

In [14]:
```
def highestReminder(n):
    hr = 0
    v = n
    for i in range(n-1,n//2,-1):
        r = n % i
        if r > hr:
            hr = r
            v = i
    print(v)
    return
highestReminder(5)
```

```
3
```

In [ ]:
```

```

Tuples

t1 = () li = [] Difference between Lists and Tuples lists are mutable - can be changed/Modified

- Used to access ,Modify,Add,Delete Data Tuples are immutable - Cannot be changed once initialised

- Used to acess data only
- All Slicing work

In [20]:
```
tl = (1,2,8,6,0)
tl[3] # Accessing 4th element
tl[len(tl)//2:] # Accessing all elements from middle to end
```

Out[20]:
```
(8, 6, 0)
```

In [21]:
```
type(tl)
```

Out[21]:
```
tuple
```

In [ ]:
```

```

Dictionaries

It works on the concept of set unique Data

keys, values key is the unique identifier for a value value is data that can be accessed with a key

In [35]:
```
dl = {"k1":"value1","k2":"value2"}
dl["k1"] # Accessing the value with key k1
dl.keys()# This is return list of all keys
dl.values()# returns list of all values
dl.items()# returns list of tuples of keys and values
# Adding the new value into the dictionaries with the help of key
dl["k3"] = "value3"
dl["k3"] = "value4"# Updating an element by using key
dl.pop("k3")#Removing an element
"k1" in dl
```

Out[35]:
```
True
```

Contacts Application

- Add Contact
- Search for contact
- List All Contacts
  - name1 : phone1
  - name2 : phone2
- Modify contact
- Remove contact
- Import Contacts

In [42]:
```
contacts = {}
def addContact(name,phone):
    # Verify that contact doesnot already exist
    if name not in contacts:
        contacts[name] = phone
        print("contact %s added" % name)
    else:
        print("Contact %s already exists" % name)
addContact("mastan","8500782761")
contacts
```

```
contact mastan added
```

Out[42]:
```
{'mastan': '8500782761'}
```

In [45]:
```
def searchContacts(name):
    if name in contacts:
        print(name,":",contacts[name])
    else:
        print("%s does not exist " % name)
    return
searchContacts("vali")
```

```
vali does not exist
```

In [52]:
```
def getAll():
    print(contacts)
getAll()
```

```
{'mastan': '9502304797'}
```

In [49]:
```
# Updating the contact
def updateContact(name,phone):
    if name in contacts:
        contacts[name] = phone
    else:
        print("%s does not exist " % name)
updateContact("mastan","9502304797")
```

In [56]:
```
# Removing the contact
def deleteContact(name):
    if name in contacts:
        contacts.pop(name)
    else:
        print("%s does not exist " % name)
deleteContact("mastan")
```

In [50]:
```
addContact("vali","8500782761")
```

```
{'mastan': '9502304797'}
```

In [53]:
```
# New Contacts is given as a dicitionary
# Merge new contacts with existing contacts
def importContacts(newContacts):
    contacts.update(newContacts)
    print(len(newContacts.keys())," added successfully")
    return
newContacts = {"chaitanya":8106410134,"sulthan":9989794454}
importContacts(newContacts)
```

```
2  added successfully
```

In [54]:
```
contacts
```

Out[54]:
```
{'mastan': '9502304797', 'chaitanya': 8106410134, 'sulthan': 9989794454}
```

In [57]:
```
getAll()
```

```
{'chaitanya': 8106410134, 'sulthan': 9989794454}
```

In [ ]:
```

```

Package and Modules

- Package

```
    - Package is collection of modules (Python file.py) and subpackage
s
```

- SubPackage
- Module

```
    - It is a single python file containing functions
```

- Package -> SubPackage ->Modules -> Functions

In [6]:
```
from math import floor as fl

fl(123.456)# to convert the floating values to integer values

#pi
```

Out[6]:
```
123
```

In [15]:
```
# Function to generate N random numbers in a given range
import random
def generateNRandomNumbers(n,lb,ub):
    for i in range(0,n):
        print(random.randint(0,100),end= " ")
generateNRandomNumbers(10,0,100)
#random.randint(0,100)
```

```
41 53 18 13 51 70 51 53 99 12
```

In [ ]:
```
```

In [7]:
```
from packages import numerical
numerical.isPrime(3)
numerical.numberPrimeFactors(96)
```

Out[7]:
```
2
```

In [8]:
```
import random
```

In [9]:
```
dir(random)
```

Out[9]:
```
['BPF',
 'LOG4',
 'NV_MAGICCONST',
 'RECIP_BPF',
 'Random',
 'SG_MAGICCONST',
 'SystemRandom',
 'TWOPI',
 '_BuiltinMethodType',
 '_MethodType',
 '_Sequence',
 '_Set',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_acos',
 '_bisect',
 '_ceil',
 '_cos',
 '_e',
 '_exp',
 '_inst',
 '_itertools',
 '_log',
 '_os',
 '_pi',
 '_random',
 '_sha512',
 '_sin',
 '_sqrt',
 '_test',
 '_test_generator',
 '_urandom',
 '_warn',
 'betavariate',
 'choice',
 'choices',
 'expovariate',
 'gammavariate',
 'gauss',
 'getrandbits',
 'getstate',
 'lognormvariate',
 'normalvariate',
 'paretovariate',
 'randint',
 'random',
 'randrange',
```

```
 'sample',
 'seed',
 'setstate',
 'shuffle',
 'triangular',
 'uniform',
 'vonmisesvariate',
 'weibullvariate']
```

In [ ]: