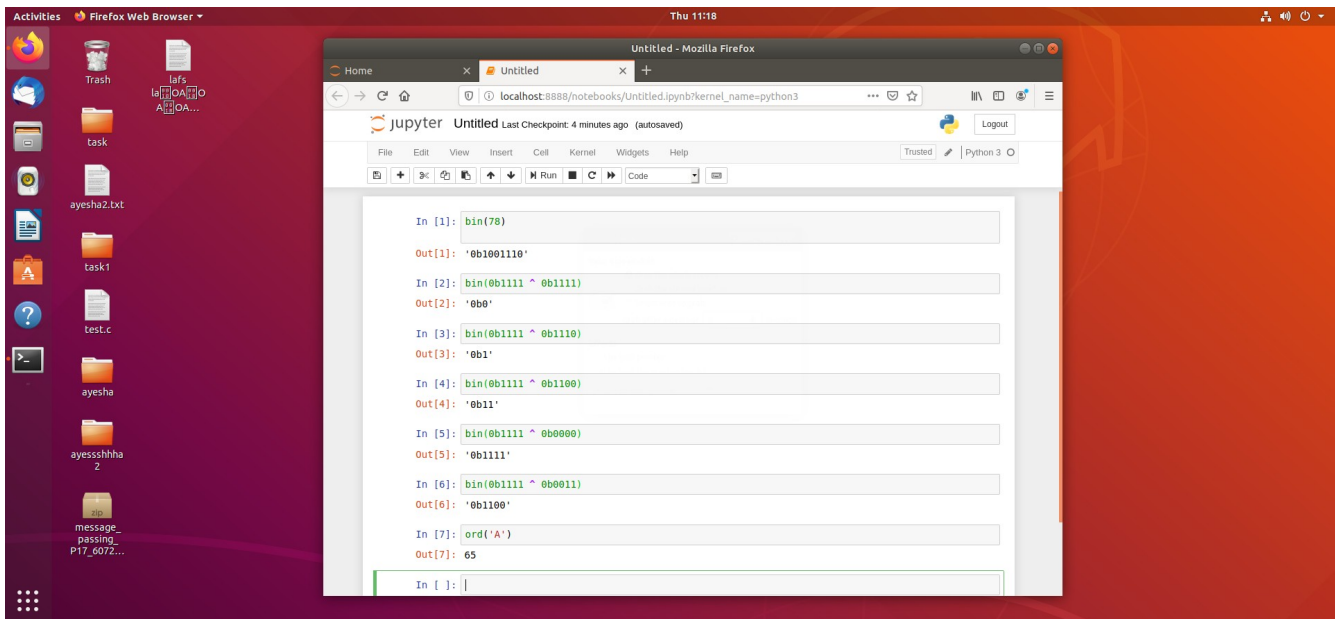


Name: Ayesha Aziz
Roll Number: P17-6072
Section: B

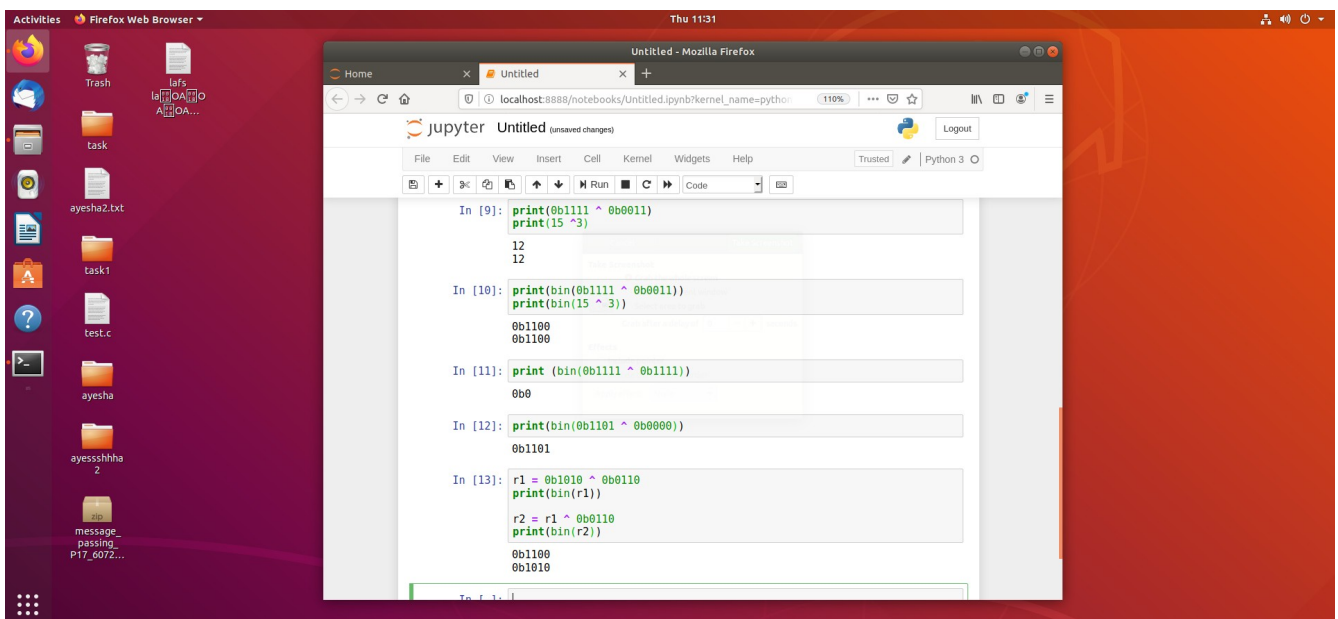
system administration Lab Task



if we want to convert any number into binary we use Bin which is used in the above screenshot converting 78 into binary.

For exclusive or I have used two binary numbers. This “^” is used for exclusive or. In the above example if we exclusive or two binary numbers it will return the binary number in string.

In the last Example I have used Unicode representation. This will return me the ascii value of A or any other special symbol or number can be used.



In the first example I have simply showed that if we take exclusive or of two binary number or any other number(having same binary number) than we get the same answer as shown above.

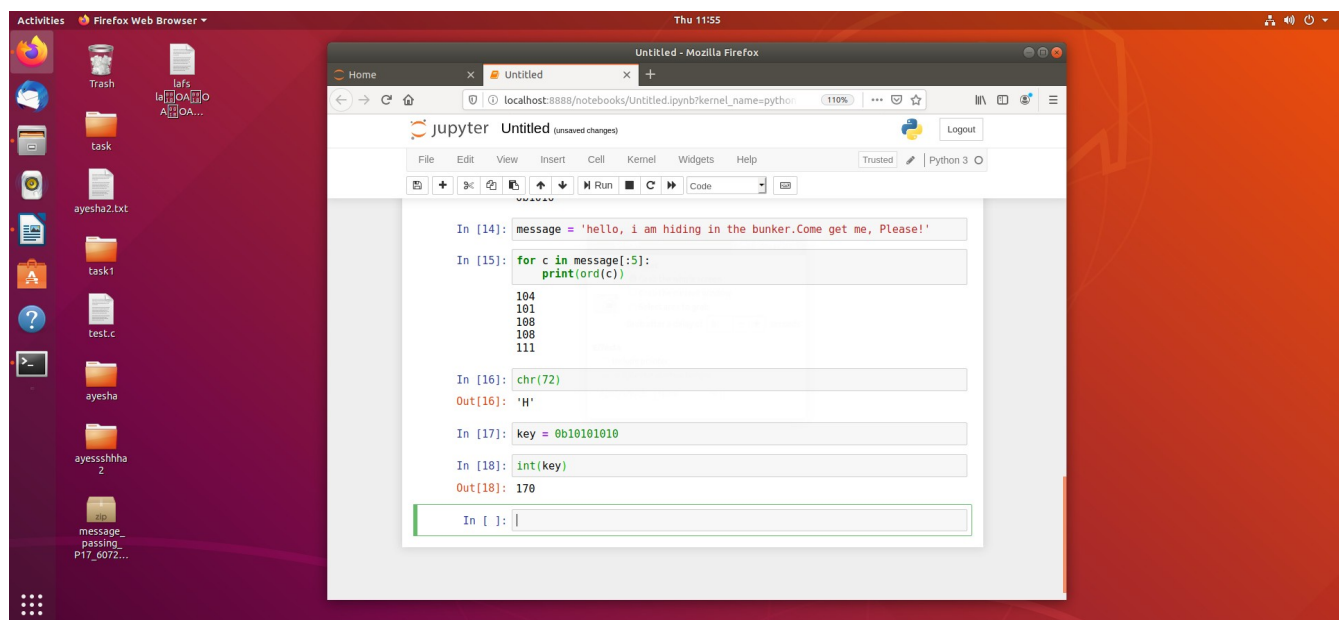
But if we use the Bin than it will give answer into binary.

If we exclusive or a number with itself we get Zero which can be seen above in the screenshot.

If we exclusive or a number with with 0 than we get some result rather than of Zero.

If we exclusive or something twice than we get the original number back which can be seen in the last example.

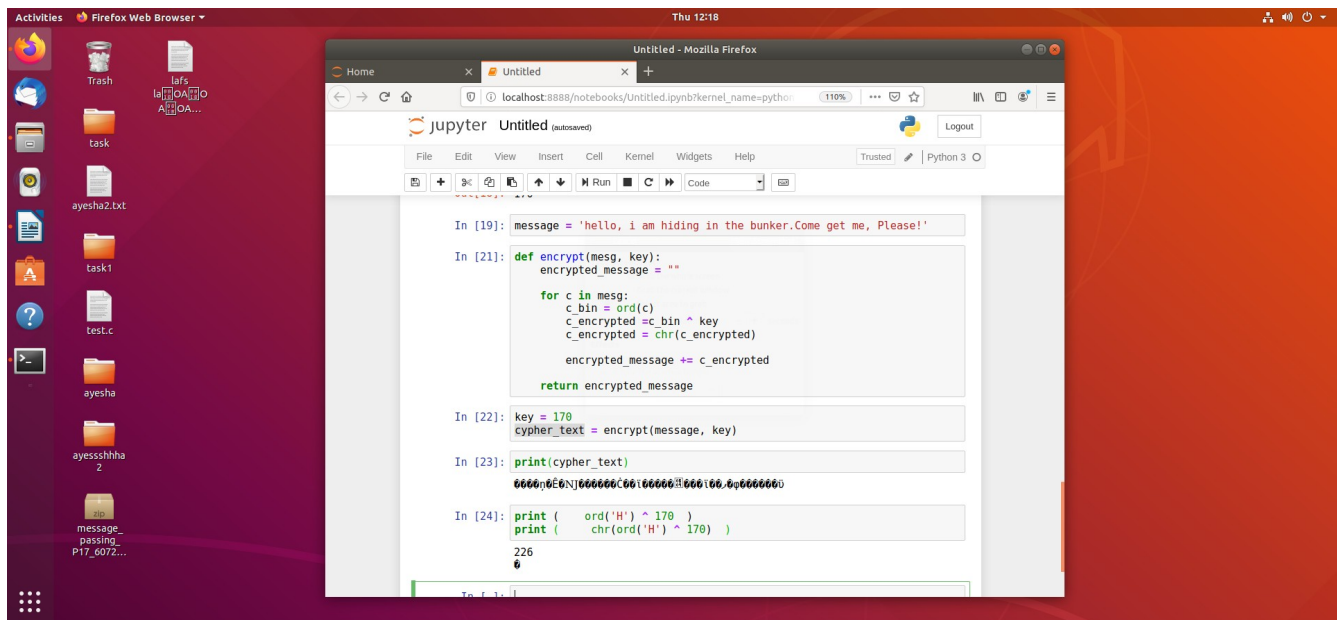
Building XOR-based Crypto



```
In [14]: message = 'hello, i am hiding in the bunker.Come get me, Please!'
In [15]: for c in message[:5]:
          print(ord(c))
104
101
108
108
111
In [16]: chr(72)
Out[16]: 'H'
In [17]: key = 0b10101010
In [18]: int(key)
Out[18]: 170
In [ ]: |
```

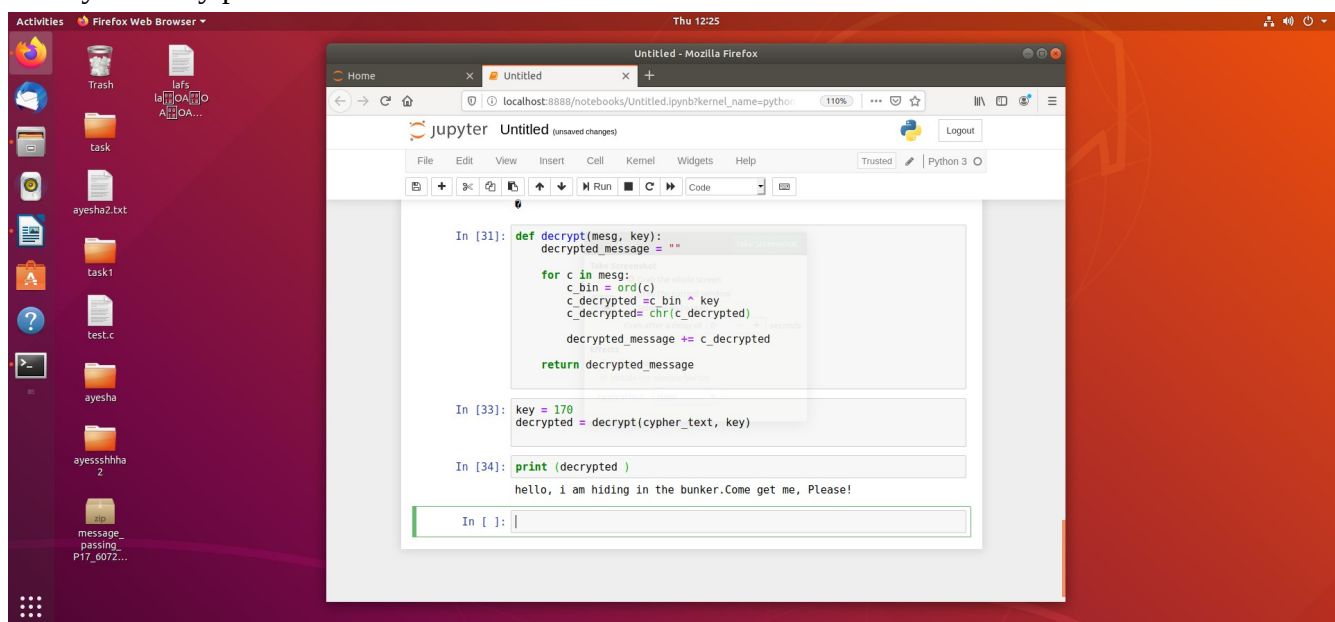
in the above first example we took one one character and exclusive or our key with character. Than you can see the result we have taken the character from start to 5.

in the second example I have simply converted 72 into character.



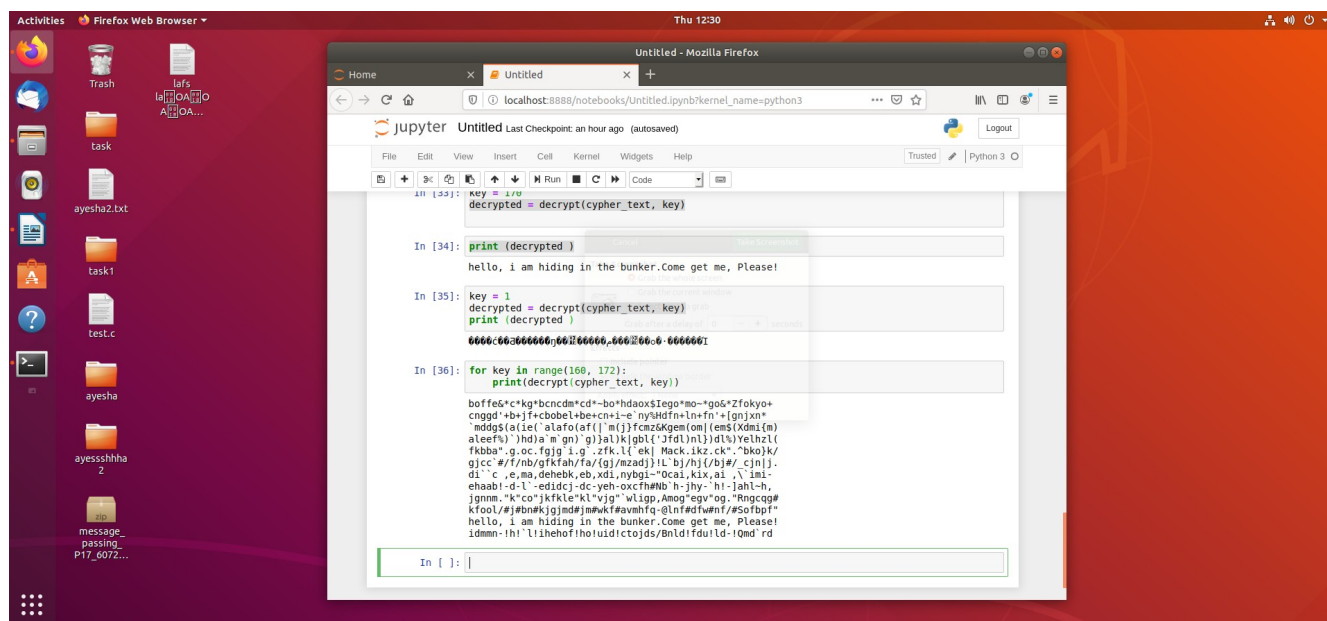
Encryption takes message from us and one key. It will simple take every character and make its **ord** and then it will exclusive or it with its key (working on decimal) . Then it will make its character representation and save it to c_encrypted. Then it will take encrypted character amd save it in encrypted_message.

When we take cypher_text then it will be printed into such a secret form that cannot be understood by any ordinary person.



Similarly when we decrypt the data it will also take one message and a key. The process will be repeated like Encryption. At the end we will get the original data.

Breaking the key



The screenshot shows a Jupyter Notebook interface in a Firefox browser. The notebook contains three code cells. The first cell defines a function `decrypt(cypher_text, key)`. The second cell prints the result of `decrypt(cypher_text, 1)`, which is a long string of garbage characters. The third cell prints the result of `decrypt(cypher_text, 172)`, which is the original message: "hello, i am hiding in the bunker.Come get me, Please!".

```
key = 172
decrypted = decrypt(cypher_text, key)

In [34]: print(decrypted)
hello, i am hiding in the bunker.Come get me, Please!

In [35]: key = 1
decrypted = decrypt(cypher_text, key)
print(decrypted)
boffe6*c*kg*bcncdm*cd*-bo*hdaox$Iego*mo-*go6*Zfokyo+
cnggd'+b+jf+cbobel+be+cn+i-e'ny$Hdfn+ln+fn'+[gn]xn*
'mdg$(a[ie' alefo[afil]'m(j)fcmz6kgem[om](es$Xdm[en]
aleef$)'hda[= gn] g)jal[k]gbl{fjdl[nl]}dl$)Yelhzl(
fkbb'a'.g.oc.fg]g'i.g'.zfk.l{ek] Mack.ikz.ck'.'bko]k/
gjcc'#/f/nb/gfkfah/fa/(g/mzad)}iL'bj/hj(/bj#/.c]n]j.
di'c'.e,ma,dehebk,eb,xdi,nybgi-'Ocsl,kix,ai '\im1-
ehaabl-d-l'-edidc]-dc-yeh-oxcfh#Nb'h-jhy-'hl-jahl-h,
jgnnm.'k'co'jkfkle'kl'v]g'w]l]gp,Amog'egv'og.'Rngcgg#
kfool/#]#bnk]g]md]m]mwkf#avmhfq-@lnfdwfnf/#Sofbpf"
hello, i am hiding in the bunker.Come get me, Please!
idmmn-lhi'lihehofiholuidictojds/Bnldlfdulld-lQmd'rd

In [ ]: |
```

in the above example we can see when we take key 1 it give use garbage values.

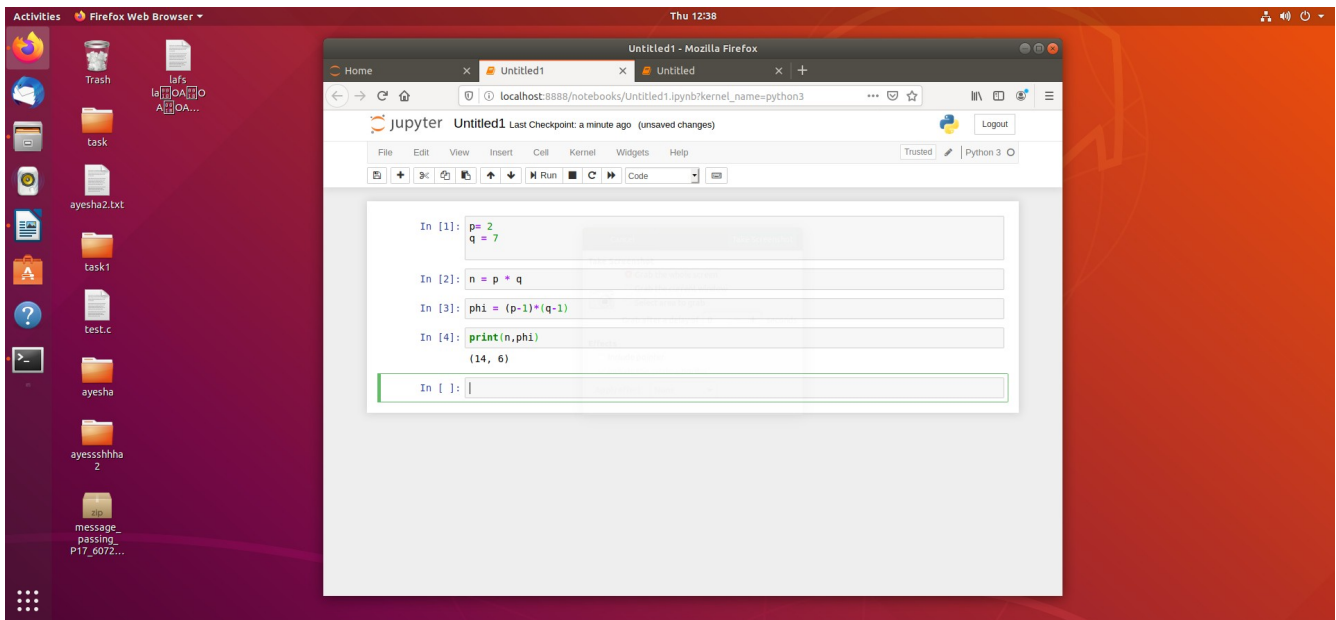
When we put this in loop it will give us meaningful data but after so many decryption.

Problem with this algorithm

The main problem for this algorithm was that what we will do when we have private key.

The solution is that we are going to share this private key with everyone but they will be still not able to decrypt this.

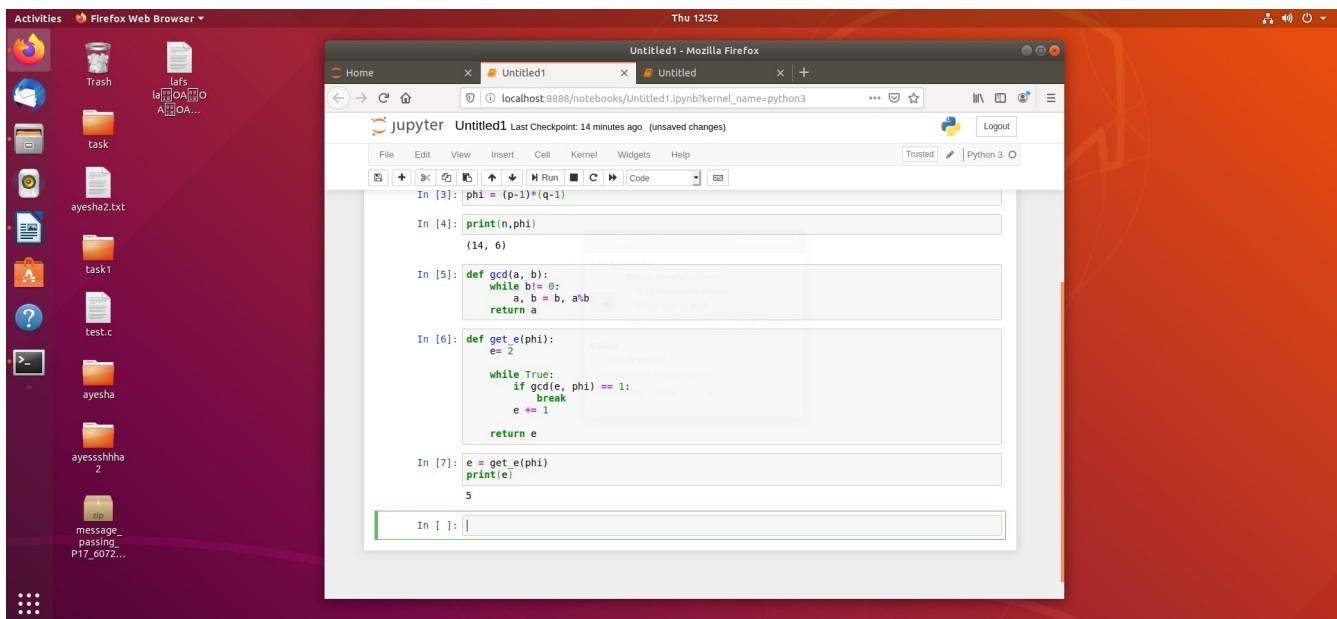
Public Key Crypto



in the above example we simply take two variables with name p and q.

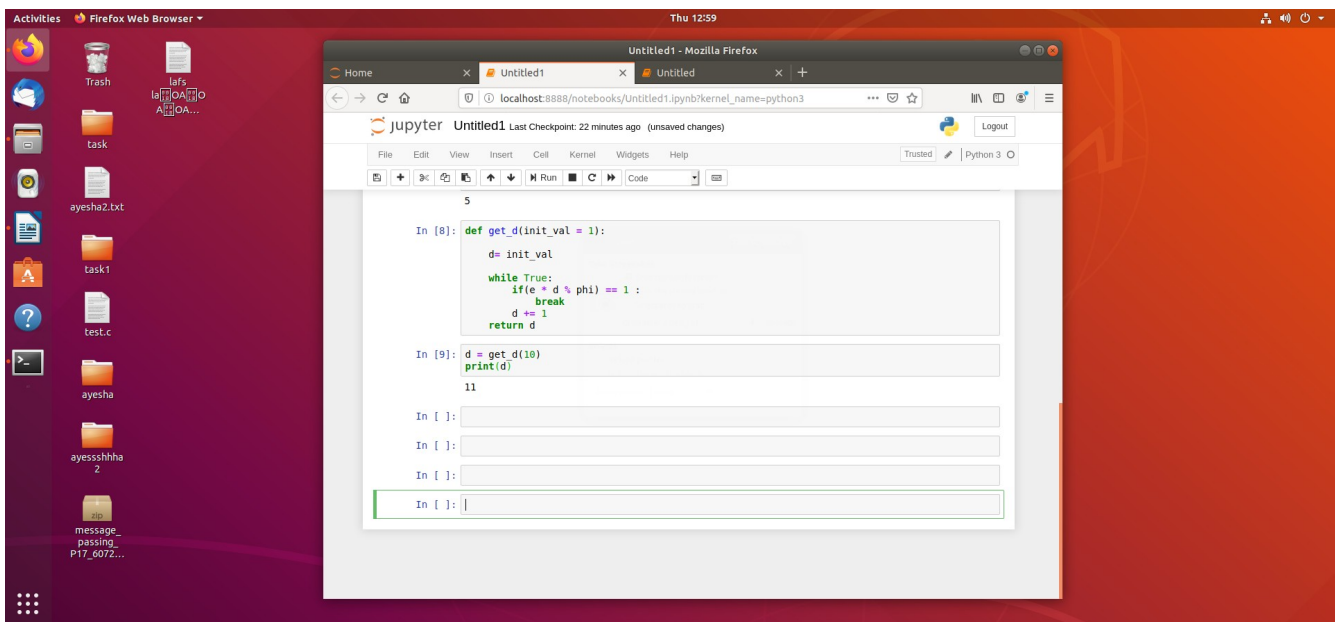
we applied two operations on it one by simple multiplying them and storing their answer in n. other operation is first subtracting 1 from both p and q and then storing their answer in phi.

Then we use print to show the difference in their answer.



Here we use GCD (greatest common divisor) . If the two numbers have GCD 1 than it means that, this is co prime.

In the second example it will take phi and e = 2 and it will simply run the loop until the GCD of e and phi is 1 (co prime).

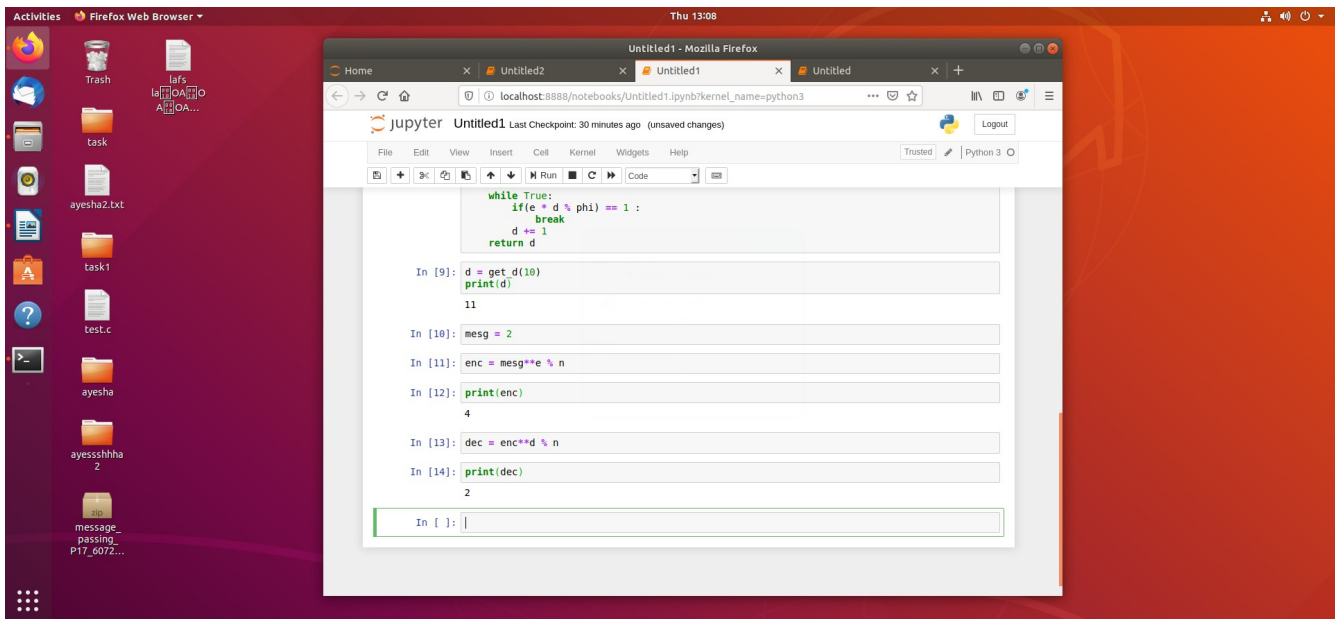


```
5
In [8]: def get_d(init_val = 1):
        d= init_val
        while True:
            if(e * d % phi) == 1 :
                break
            d += 1
        return d

In [9]: d = get_d(10)
        print(d)
11

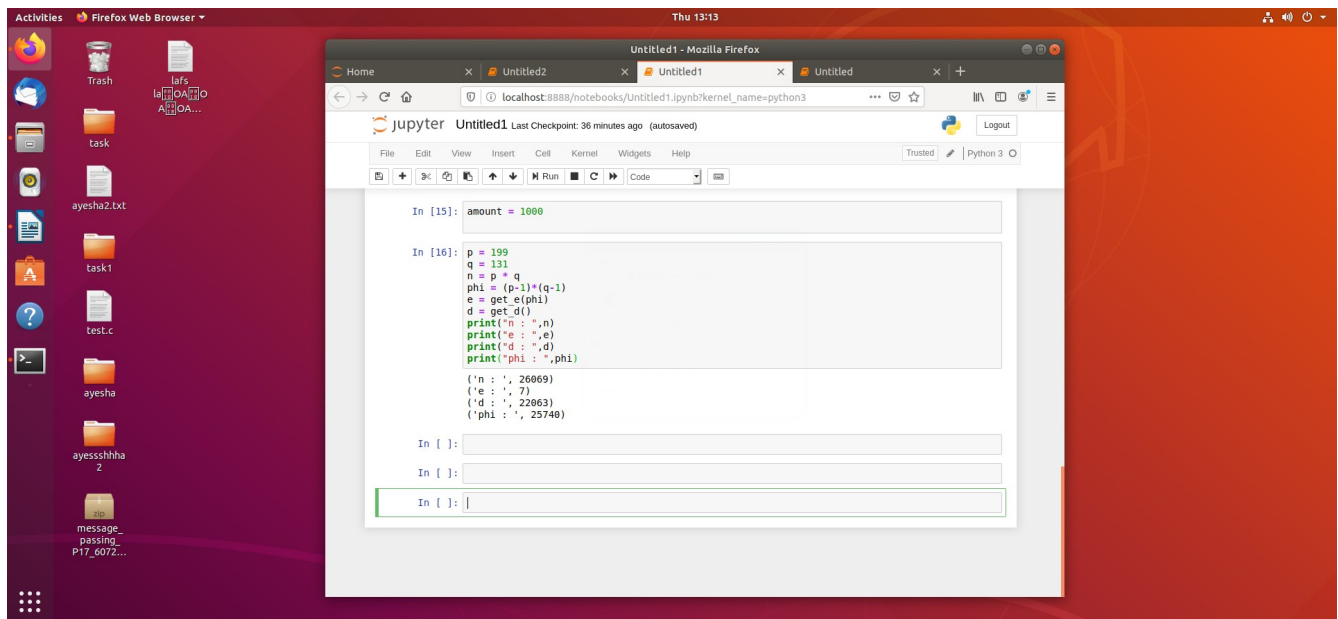
In [ ]: 
In [ ]: 
In [ ]: 
In [ ]: 
```

In the above example we start from initial value which is one and when the condition is satisfied that is when $e * d$ gives something and we take its % with phi it gives us 1.

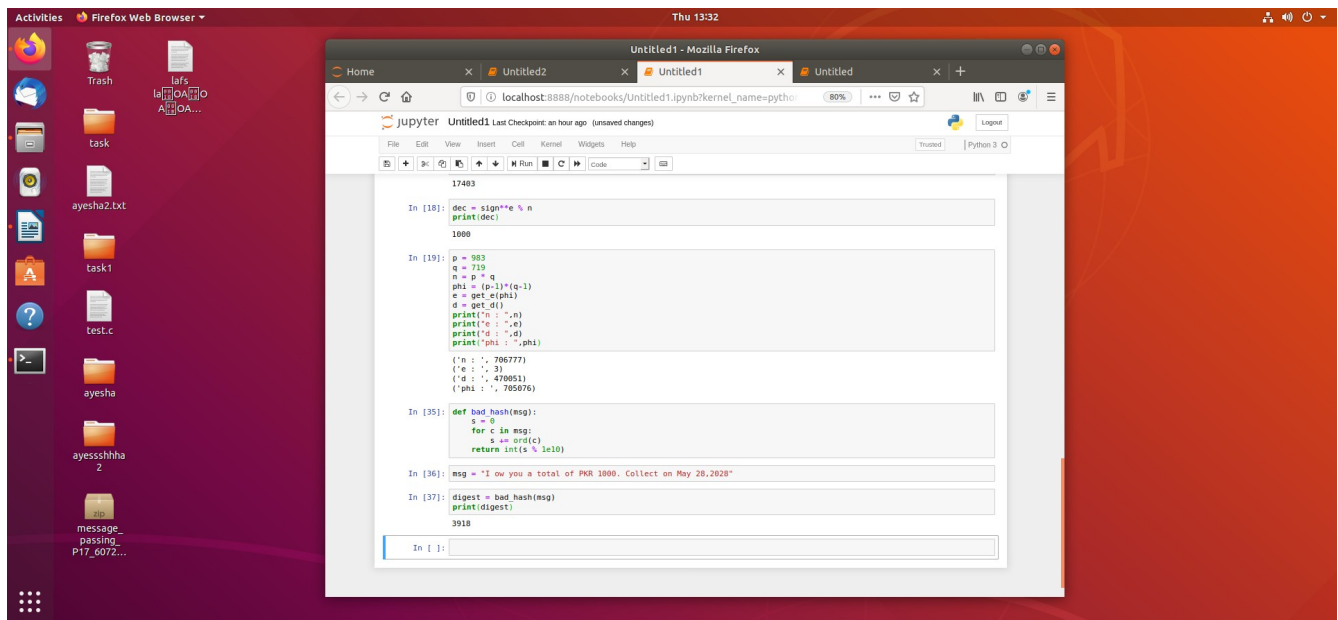
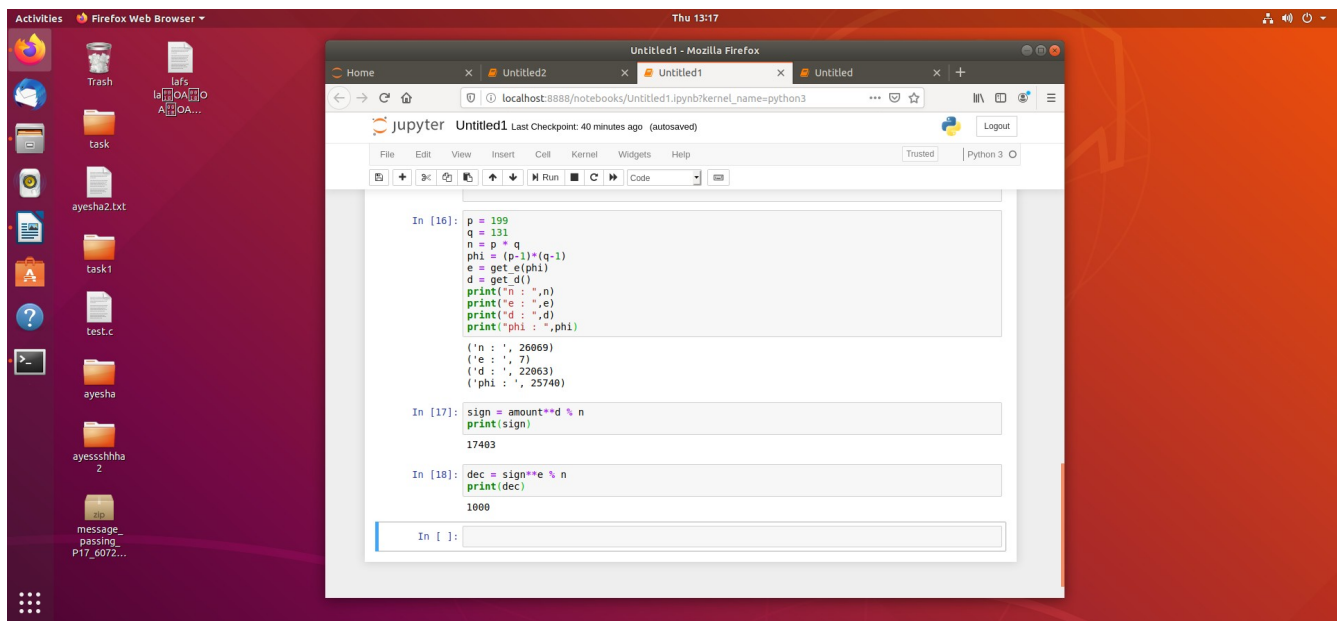


In the above example we are going to publish e and n . Now Alice will do $msg ** e$ and then take its $\%$ with n . It will give us the value. We have now published this where we want to.

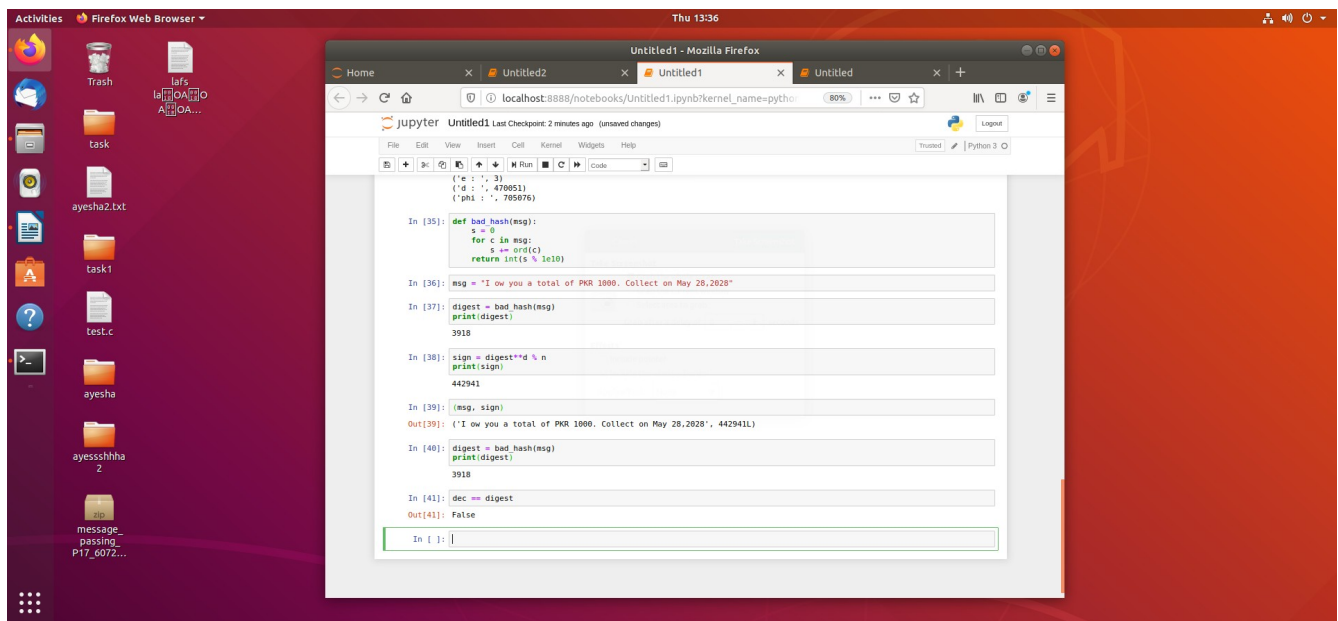
Now when we will decrypt the received data it will return us the original message.



We will first encrypt amount with d . After computing the following we will get some values as shown above.



Now we make a hash function which takes message in parameter. This will take one by one character and apply ord on it at the end it will give us hash value.



In the above example it will simple assign the value to sign after solving this.

In next example it will print the message along with sign value .

In next example it will print the digest value on the screen.