Lesson 1

Computer programs are about processing numbers, strings, images, sounds, etc. To show what can Python do, we'll begin with strings. a collection of strings make up texts... And so, we begin...

We can read a text, if we understand the language, in which it is written. Because we understand words, which make up the text. And we understand letters, out of which the text is made.

For example, we can read a text like this:

> IF you happen to have read another book about Christopher Robin, you may remember that he once had a swan (or the swan had Christopher Robin, I don't know which) and that he used to call this swan Pooh. That was a long time ago, and when we said good-bye, we took the name with us, as we didn't think the swan would want it any more. Well, when Edward Bear said that he would like an exciting name all to himself, Christopher Robin said at once, without stopping to think, that he was Winnie-the-Pooh. And he was. So, as I have explained the Pooh part, I will now explain the rest of it.

We can even name the book, this piece of text comes from...

Right! The book is *Winnie-The-Pooh and All, All, All* by **Alan Alexander Miln**

But what if the letters in a text are shuffled in a certain way? Scientists say, that very little will change for a reader.

Here is an example...

> Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

Are we still able to read it?

This is because we pick up not **single letters**, but recognize a **word as a whole**.

Let's have a closer look!

We are going to write a Python program that takes a text and mixes up letters. And we are going to see how hard or easy it will be for us to understand the whole text...

Here we begin our program...

First of all, we need a variable that will hold the piece of text:

In [4]:
```
text = '''If you happen to have read another book about Christopher Robin, y
ou may remember that he once had a swan (or the swan had Christopher Robin,
I don't know which) and that he used to call this swan Pooh. That was a long
 time ago, and when we said good-bye, we took the name with us, as we didn't
 think the swan would want it any more. Well, when Edward Bear said that he
would like an exciting name all to himself, Christopher Robin said at once,
without stopping to think, that he was Winnie-the-Pooh. And he was. So, as I
 have explained the Pooh part, I will now explain the rest of it.'''
```

So far, we can't see any result. But if we print the text, we will see it...

In [2]:
```
print(text)
```

```
If you happen to have read another book about Christopher Robin, you may rem
ember that he once had a swan (or the swan had Christopher Robin, I don't kn
ow which) and that he used to call this swan Pooh. That was a long time ago,
 and when we said good-bye, we took the name with us, as we didn't think the
 swan would want it any more. Well, when Edward Bear said that he would like
 an exciting name all to himself, Christopher Robin said at once, without st
opping to think, that he was Winnie-the-Pooh. And he was. So, as I have expl
ained the Pooh part, I will now explain the rest of it.
```

But in our program we will need not the whole text at once, but particular words one by one. How do we split the test?

In [5]:
```
textwords = text.split(' ')
print(textwords)
```

```
['If', 'you', 'happen', 'to', 'have', 'read', 'another', 'book', 'about', 'C
hristopher', 'Robin,', 'you', 'may', 'remember', 'that', 'he', 'once', 'had'
, 'a', 'swan', '(or', 'the', 'swan', 'had', 'Christopher', 'Robin,', 'I', "d
on't", 'know', 'which)', 'and', 'that', 'he', 'used', 'to', 'call', 'this',
'swan', 'Pooh.', 'That', 'was', 'a', 'long', 'time', 'ago,', 'and', 'when',
'we', 'said', 'good-bye,', 'we', 'took', 'the', 'name', 'with', 'us,', 'as',
 'we', "didn't", 'think', 'the', 'swan', 'would', 'want', 'it', 'any', 'more
.', 'Well,', 'when', 'Edward', 'Bear', 'said', 'that', 'he', 'would', 'like'
, 'an', 'exciting', 'name', 'all', 'to', 'himself,', 'Christopher', 'Robin',
 'said', 'at', 'once,', 'without', 'stopping', 'to', 'think,', 'that', 'he',
 'was', 'Winnie-the-Pooh.', 'And', 'he', 'was.', 'So,', 'as', 'I', 'have', '
explained', 'the', 'Pooh', 'part,', 'I', 'will', 'now', 'explain', 'the', 'r
est', 'of', 'it.']
```

We got a list of words, which make up the text. We can see as well, that some punctuation marks become parts of the words. We don't need them in our program. How do we remove them?

In [9]:
```
text = text.replace('.', '')
text = text.replace(',', '')
text = text.replace('-', ' ')
text = text.replace('(', '')
text = text.replace(')', '')
textwords = text.split()
print (textwords)
```

```
['If', 'you', 'happen', 'to', 'have', 'read', 'another', 'book', 'about', 'C
hristopher', 'Robin', 'you', 'may', 'remember', 'that', 'he', 'once', 'had',
 'a', 'swan', 'or', 'the', 'swan', 'had', 'Christopher', 'Robin', 'I', "don'
t", 'know', 'which', 'and', 'that', 'he', 'used', 'to', 'call', 'this', 'swa
n', 'Pooh', 'That', 'was', 'a', 'long', 'time', 'ago', 'and', 'when', 'we',
'said', 'good', 'bye', 'we', 'took', 'the', 'name', 'with', 'us', 'as', 'we'
, "didn't", 'think', 'the', 'swan', 'would', 'want', 'it', 'any', 'more', 'W
ell', 'when', 'Edward', 'Bear', 'said', 'that', 'he', 'would', 'like', 'an',
 'exciting', 'name', 'all', 'to', 'himself', 'Christopher', 'Robin', 'said',
 'at', 'once', 'without', 'stopping', 'to', 'think', 'that', 'he', 'was', 'W
innie', 'the', 'Pooh', 'And', 'he', 'was', 'So', 'as', 'I', 'have', 'explain
ed', 'the', 'Pooh', 'part', 'I', 'will', 'now', 'explain', 'the', 'rest', 'o
f', 'it']
```

Now we need a piece of Python code, that will take a word, separate the first and the last letters (to keep them in their places), but will change places of the letters in the middle. For example, we can revert the order of the middle part. Like this: E XPLAI N --> E IALPX N

To do this, slices in Python are very helpful. Here are some examples:

In [6]:
```
a = [1, 2, 3, 4, 5]
#a[start:stop:step]
print( a[::-1])
print(a[1:4:2])
print(a[-1:0:-1])
```

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

In [8]:
```
for word in textwords:
    fl = word[0]
    ll = word[-1:]
    m = word[-2:0:-1]
    nw = fl + m + ll
    print (nw, end=' ')
```

```
If you heppan to hvae raed aehtonr book auobt Cehpotsirhr Ribon you may rebm
emer taht he ocne had aa sawn or the sawn had Cehpotsirhr Ribon II d'not kon
w wcihh and taht he uesd to clal tihs sawn Pooh Taht was aa lnog tmie ago an
d wehn we siad gybdooe we took the nmae wtih us as we d'ndit tnihk the sawn
wluod wnat it any mroe Wlel wehn Erawdd Baer siad taht he wluod lkie an enit
icxg nmae all to hlesmif Cehpotsirhr Ribon siad at ocne wuohtit snippotg to
tnihk taht he was WooPehteinnih And he was So as II hvae eenialpxd the Pooh
prat II wlil now eialpxn the rset of it
```

We got it! But the resulting text is very hard to read! That means, reverting the middle is too big change of a word. We need another program. That will do smaller changes. For example, change just two neighbour letters in the middle. Doing this, we have to remember, that words that are 3 letters long and shorter can't be changed in that way. And 4-letter words can be changes in a single way.

In [9]:
```
for word in textwords:
    if len(word) >3:
        fl = word[0]
        sl = word[1]
        tl = word[2]
        rest = word[3:]
        nw = fl + tl + sl + rest
    else:
        nw = word
    print (nw, end = ' ')
```

```
If you hpapen to hvae raed aonther book aobut Crhistopher Rboin you may rmee
mber taht he ocne had a sawn or the sawn had Crhistopher Rboin I dno't konw
wihch and taht he uesd to clal tihs sawn Pooh Taht was a lnog tmie ago and w
ehn we siad goodbye we took the nmae wtih us as we ddin't tihnk the sawn wuo
ld wnat it any mroe Wlel wehn Ewdard Baer siad taht he wuold lkie an ecxitin
g nmae all to hmiself Crhistopher Rboin siad at ocne wtihout sotpping to tih
nk taht he was WniniethePooh And he was So as I hvae epxlained the Pooh prat
 I wlil now epxlain the rset of it
```

Now it became too simple. We need to rewrite our program again. Now we want to replace two random letters in the middle. To do this, we need to import the 'random' module in Python.

In [10]:
```
import random
for word in textwords:
    if len(word) >4:
        fl = word[0]
        ll = word[-1:]
        ml = word[1:-1]
        i = random.randint(0, len (ml) - 2)
        mllst = list(ml)
        mllst[i], mllst[i+1] = mllst[i+1], mllst[i]
        ml = "".join(mllst)
        nw = fl + ml + ll
    else:
        nw = word
    print (nw , end = ' ')
```

```
If you hapepn to have read anotehr book aobut Chrsitopher Roibn you may reme
bmer that he once had a swan or the swan had Christpoher Roibn I dno't know
whcih and that he used to call this swan Pooh That was a long time ago and w
hen we said good bye we took the name with us as we dind't thnik the swan wu
old want it any more Well when Ewdard Bear said that he wolud like an exicti
ng name all to hismelf Chirstopher Rboin said at once withuot stoppnig to ti
hnk that he was Wninie the Pooh And he was So as I have explaiend the Pooh p
art I will now explian the rest of it
```

Let us make our code more efficient for future use. Let us make a function, that takes a word and returns the changes one.

In [12]:
```
def shuffle(word):
    if len(word) >4:
        fl = word[0]
        ll = word[-1:]
        ml = word[1:-1]
        i = random.randint(0, len (ml) - 2)
        mllst = list(ml)
        mllst[i], mllst[i+1] = mllst[i+1], mllst[i]
        ml = "".join(mllst)
        nw = fl + ml + ll
    else:
        nw = word
    return nw
```

Now we need the main program, that will call our function in a cycle and process the whole list.

In [14]:
```
import random
for t in textwords:
    print (shuffle(t),end = ' ')
```

```
If you hapepn to have read anotehr book abuot Chirstopher Rboin you may reme
mebr that he once had a swan or the swan had Chritsopher Roibn I do'nt know
wihch and that he used to call this swan Pooh That was a long time ago and w
hen we said good bye we took the name with us as we ddin't thnik the swan wo
lud want it any more Well when Ewdard Bear said that he wuold like an exicti
ng name all to himslef Christpoher Rboin said at once wihtout stopipng to th
nik that he was Winnie the Pooh And he was So as I have epxlained the Pooh p
art I will now expalin the rest of it
```

Finally, we need a code that will collect isolated words together into a single text and will print it.

In [11]:
```
#Your code here... Put the text together again.
```

It works!

We see that a text with some letters replaced is still readable. But the more letters we replace, and the more distant are the letters from each other, the harder is to understand the result.

And now - a different task!

How do we make a text **totally unreadable**? That is, how do we **encode** it... Of course we want the text to be still readable for some friends. Who know how to **decode it**...

The easiest way is to replace letters with different ones. Later we will put the letters back to their places to decode the text

We are going to use a Python dictionary:

In [12]:
```
encodic = {'a':'z', 'b':'y', 'c':'x'}
```

Because our dictionary is not complete (it does't contain the whole alphabet), we need to provide that letters, that are not in the dictionary will stay in their places

Итак, напишем всю программу целиком: она берет текст, обходит букву за буквой и заменяет буквы в соответствии со словарем

In [13]:
```
#Your code here... The entire encoding program!
```

It works! And finally we write a program that makes the reverse transformation. It will decode the encoded text by returning the letters into their places.

In [14]:
```
decodic = {'z':'a', 'y':'b', 'x':'c'}
```

In [15]:
```
#Your code here... The entire decoding program!
```

Let us now make the program better. Let us replace letters with digits and symbols

In [ ]:
```
#Your code here...
```

That is it!

Thank you!

In [ ]:
```

```