Python Lesson 3

Sending secret messages!

We're going to write a Python program which takes a secret message and encrypts it using a code or **cipher**. A cipher is an **algorithm** (a series of steps) for transforming a message to hide its meaning.

We'll use a simple **substitution cipher**, which means that each letter in the alphabet is swapped for a different letter in the alphabet. The code we'll use was apparently first used by the Roman emperor Julius Caesar to communicate to his generals, so it is called a **Caesar Cipher**.

Secret codes are no good without the intended recipient being able to decrypt and read the message, so we'll write another program do that.

How Computers Work: Text and Character Sets

Computers store text strings as lists of characters (letters, punctuation, numbers, etc.), and each character is stored as a number. A set of rules describing what characters are available and how to convert between characters and numbers is called a **character set**. Without an agreed character set, sending messages between different types of computer would get very confusing! The number for the letter 'C' on one type of computer might appear as '{' on a different computer so messages sent between computers would be mangled!

Almost all computers in countries which use the same alphabet as English (the Latin alphabet) use a **character set** derived from **ASCII,** the "*American Standard Code for Information Interchange*". ASCII doesn't work for languages with different alphabets like Greek, Russian, Arabic or Japanese.[1]

This means that our code will just work with the Latin alphabet.

Every symbol in the ASCII character set is assigned a number between 0 and 127 (7 binary digits). For example, space is 32, zero is 48, A is 65 and Z is 90. We can easily print character for each symbol with Python! The `chr(number)` Python function returns the character for an ASCII code.

```
for code in range(32, 128):
    print(chr(code), code, sep="=", end="\t")
    if code % 8 == 7:
        print()
```

1 These days, with computers sharing data on the internet all around the world in countries with different languages, we need a shared character set that can support more than Latin. The **Unicode** standard extends ASCII by adding extra symbols for most other languages (**齒页黾**) and stranger things like ☺♥♬➽❀. ASCII has 128 characters but the first 32 aren't printable and have special meanings (e.g. tabs, newlines, end of string marker); Unicode currently has more than a million characters and it's increasing! Fortunately, each ASCII character has the same number in Unicode.

Printable ASCII characters and their values:

```
 =32  !=33  "=34  #=35  $=36  %=37  &=38  '=39
(=40  )=41  *=42  +=43  ,=44  -=45  .=46  /=47
0=48  1=49  2=50  3=51  4=52  5=53  6=54  7=55
8=56  9=57  :=58  ;=59  <=60  ==61  >=62  ?=63
@=64  A=65  B=66  C=67  D=68  E=69  F=70  G=71
H=72  I=73  J=74  K=75  L=76  M=77  N=78  O=79
P=80  Q=81  R=82  S=83  T=84  U=85  V=86  W=87
X=88  Y=89  Z=90  [=91  \=92  ]=93  ^=94  _=95
`=96  a=97  b=98  c=99  d=100 e=101 f=102 g=103
h=104 i=105 j=106 k=107 l=108 m=109 n=110 o=111
p=112 q=113 r=114 s=115 t=116 u=117 v=118 w=119
x=120 y=121 z=122 {=123 |=124 }=125 ~=126 ¨=127
```

You can see that capital (upper case) letters are in the range 65 to 90 and small (lower case) letters are in the range 97 to 122.

Encrypting Messages

Create a new Python program file called `encrypt.py`. When finished the program should let you type in a message, then print the encrypted version of that message.

To encrypt a message for our cipher we will convert each character of the Latin alphabet (characters "A" to "Z" and "a" to "z") in the message to the equivalent ASCII number, do some simple arithmetic on the number, then convert back to a different symbol.

We need to make sure that the arithmetic is reversible so we can decrypt the message! For our Caesar cipher we will add 3 to the ASCII value to encrypt an subtract 3 to decrypt. We need to take some care at the end of the alphabet to make sure that each letter to becomes a different letter: A→D, B → E, C → F, … X → A, Y → B, Z → C.

There are many ways you can write this program, but here are some hints and suggestions:

- `ord(character)` is an in-built function that returns the ASCII value of a string containing a single letter or symbol.
- `chr(number)` is a function that does the opposite. It returns a string containing a single character from an ASCII value.
- Try writing a function `def encrypt_character(char):` which takes a single character as an argument, calculates the encrypted version of that character, and returns it.
- In our cipher we will only encrypt letters of the alphabet and we will leave other characters (spaces, punctuation, numbers) unchanged. Your function will need to use an `if` statement to check what type of character it has been given. This line will check if the character is a letter between A and W.

  ```
  if char.upper() >= 'A' and char.upper() <= 'W':
  ```

- To encrypt a letter between 'A' and 'W', take the ASCII value of the letter and add 3 to it.
- For the other letters in the alphabet ('X' to 'Z') subtract 23, so that 'X', 'Y', and 'Z' become 'A', 'B', and 'C'.
- The line `message=input("Type your secret message: ")` will use the `input()` function to ask you to type a message and store the string value that you typed in a variable called `message`.
- Can you remember how to use a `for` statement to loop over every character in a string? We included an example of this in the previous lesson. This will allow your application to call your function for each letter. You can use string concatenation (+) to build up the encrypted message before you print it.

Decrypting Messages

The program to decrypt messages will be very similar to the program to encrypt messages. Save a copy of the program that encrypts messages with the new name `decrypt.py`.

The main change that we'll need to make is to reverse the arithmetic: subtract 3 for letters in the range D-Z and add 23 for other letters.

Can you work out what to do?

When you think you've finished, try typing in (or copy & paste!) the result of a message that you encrypted with `encrypt.py` to test if `decrypt.py` gets you back the original secret message.

Can your program correctly decrypt this message?

```
L nqrz d ghdg sduurw zkhq L vhh rqh, dqg L'p orrnlqj dw rqh uljkw qrz.
```

How Computers Work: Security and Data Encryption

This cipher is just a bit of fun. Unfortunately, it's the code is just too easy to crack to be used for sending any important secrets. You could do it in a few minutes with a pencil and some paper. Someone should warn Julius Caesar!

It's very hard to invent your own unbreakable cipher for storing and sending secrets, and there's a lot of advanced (but interesting) maths involved. If one day you decide that your programs need strong encryption for any reason, it's best to either:

1) Use a Python module such as `simplecrypt` which has been written by experts. This provides the same types of cipher used by banks, governments and spies and it is really hard to crack.
2) Study computer science or maths at university!

Solutions

Try not to peek at these unless you get stuck!

One possible solution for **encrypt.py**.

```
def encrypt_character(char):
    value=ord(char)
    upper=char.upper()
    if upper >= 'A' and upper <= 'W':
        value=value+3
    elif upper >= 'X' and upper <= 'Z':
        value=value-23
    return chr(value)

message=input("Type your secret message: ")
encrypted_message=''
for char in message:
    encrypted_message = encrypted_message + encrypt_character(char)
print(encrypted_message)
```

One possible solution for **decrypt.py.**

```
def decrypt_character(char):
    value=ord(char)
    upper=char.upper()
    if upper >= 'D' and upper <= 'Z':
        value=value-3
    elif upper >= 'A' and upper <= 'C':
        value=value+23
    return chr(value)

encrypted_message=input("Type your encrypted message: ")
message=''
for char in encrypted_message:
    message = message + decrypt_character(char)
print(message)
```

There are many other ways you can write these programs so you don't need to worry if you've done it another way.