# ASCII standard

How does a computer, which can only work with binary numbers, handle letters (characters)? It will probably not surprise you that this is done by representing each of these characters as a binary number. The first developers of computer systems came up with several ways to represent characters as binary numbers.

But first of all, what is a character? Characters are entities that you print on a sheet, which usually consist of letters (a, b, c, ...), numbers (0, 1, 2, ...), and punctuation marks (period, comma, semicolon, ...). However, there are characters that cannot be printed, such as a *carriage return* or *tab*, along with other characters that go back to the time of the very first computer systems such as *form feed*.

The first standardized way to present characters in a computer as binary numbers was the *American Standard Code for Information Interchange*, or ASCII for short, developed in 1963. The ASCII table shows how each of the natural numbers from 0 to 255 can be converted to its corresponding character, and vice versa. For example, the character "A" is represented by the number 65, while the number 97 is the character "a".

Luckily, you do not have to know this table by heart in order to work with the ASCII representation of characters. In Python, you can use the following two built-in functions:

- chr(*n*) prints the character that corresponds with the number *n* in the ASCII table. In that way chr(97) returns the string "a".
- ord(*c*) prints the ASCII value of the character *c* . For example: ord("A") returns the value 65 .

## Input

No input.

## Output

Write out a portion of the ASCII table by writing a line for each number from 33 to 126, showing the number, followed by a tab and the character that corresponds to this number in the ASCII table. This part of the ASCII table only contains printable characters.

## Example

**Output:**

```
33	!
34	"
35	#
...
124	|
125	}
126	~
```

Hoe gaat een computer die enkel kan werken met binaire getallen om met letters (karakters)? Het zal je wellicht niet verwonderen dat dit gebeurt door elk van deze karakters voor te stellen als

een binair getal. De eerste ontwikkelaars van computersystemen bedachten daarvoor verschillende manieren om karakters voor te stellen door binaire getallen.

Maar eerst en vooral, wat is een karakter? Karakters zijn die entiteiten die je afgedrukt ziet op een blad en die meestal bestaan uit letters (a, b, c, …), cijfers (0, 1, 2, …), en leestekens (punt, komma, puntkomma, …). Er bestaan echter ook karakters die niet kunnen afgedrukt worden, zoals een *carriage return* of een *tab*, naast andere karakters die teruggaan op de tijd van de allereerste computersystemen zoals *form feed*.

De eerste gestandaardiseerde manier om karakters als binaire getallen voor te stellen in een computersysteem was de *American Standard Code for Information Interchange* of kortweg ASCII, ontwikkeld in 1963. De ASCII tabel geeft aan hoe elk van de natuurlijke getallen 0 tot en met 255 kan omgezet worden naar zijn corresponderend karakter, en omgekeerd. Zo wordt bijvoorbeeld het karakter "A" voorgesteld door het getal 65, terwijl het getal 97 het karakter "a" voorstelt.

Gelukkig moet je deze tabel niet van buiten kennen om te kunnen werken met de ASCII-voorstelling van karakters. In Python kan je hiervoor gebruik maken van de volgende twee ingebouwde functies:

- chr(*n*) geeft het karakter terug dat correspondeert met het getal *n* in de ASCII tabel. Zo geeft chr(97) bijvoorbeeld de string "a" terug.
- ord(*c*) geeft de ASCII-waarde van het karakter *c* terug. Zo geeft ord("A") de waarde 65 terug.

## Invoer

Geen invoer.

## Uitvoer

Schrijf een deel van de ASCII-tabel uit, door voor elk getal vanaf 33 tot en met 126 een regel uit te schrijven met daarop het getal, gevolgd door en tab en het karakter dat correspondeert met dit getal in de ASCII-tabel. In dit deel van de ASCII-tabel staan enkel afdrukbare karakters.

## Voorbeeld

### Uitvoer:

```
33	!
34	"
35	#
...
124	|
125	}
126	~
```