Powerful numbers

General information

Premature abortion of loops

In Python you can use the statements `break` and `continue` to abort a loop before it has come to completion. In general, however, these statements are considered bad programming style.

One situation where you may want a premature abortion of a loop occurs when you want to find a solution by trying all possible cases, and stop as soon as one solution has been found. Instead of using `break` or `continue` in this case, it is better to use an additional Boolean variable that indicates whether or not the solution has already been found.

```
>>> found = False
>>> while not found:
...     if (solution found): #solution found represents a condition
...         found = True
...
```

As soon as the solution has been found (represented here by the fact that the condition *solution found* evaluates to `True`), the variable `found` is assigned the value `True`. As a result, the `while`-loop ends the next time the `while`-conditions is evaluated after the current iteration.

Cowsay

General information

Escaping literal backslashes

In Python a backslash is used inside a string to escape the next character in the string, so that this character loses its special meaning (for example to put a double quote inside a string that is enclosed itself in between a pair of double quotes) or to give a special meaning to the next character (such as the end of line that is represented by the character `'\n'` or a tab that is represented by the character `'\t'`).

Because this gives a special meaning to a backslash inside a string (used to escape characters), a literal backslash inside a string must be represented by two successive backslashes (`'\\'`), where the first backslash serves as the escape symbol and the second backslash is the character that is given its literal meaning by escaping it.

Remove leading and/or trailing whitespace

In Python you can use the string method `strip` to remove leading and trailing whitespace (spaces, tabs and newlines). In case you only want to remove leading whitespace, you can use the string method `lstrip`. In case you only want to remove trailing whitespace, you can use the string method `rstrip`. You can also pass an argument to these string methods, that indicates which leading and/or trailing characters have to be removed. For more details about these string methods, we refer to The Python Standard Library.

```
>>> text = '  This is a text  '
>>> text.strip()
'This is a text'
>>> text.lstrip()
'This is a text  '
```

```
>>> text.rstrip()
'  This is a text'
```

Remarks

String repetition

To repeat a given string a fixed number of times, you can multiply that string with an integer. As with the multiplication of numbers, this uses the `*` operator. The order of the string and the integer does not matter in the multiplication

```
>>> 'a' * 3
'aaa'
>>> 5 * 'ab'
'abababababab'
```

This is very handy in case the number of repetitions of the string is not known beforehand, but for example can be retrieved from a variable.

```
>>> repetitions = 4
>>> repetitions * 'abc'
'abcabcabcabc'
```

Wepe speapeak p

Specific information

Een mogelijke strategie die je voor deze opgave kunt gebruiken is de volgende. Je itereert over de posities van de regel en je werkt met 2 variabelen. De eerste variabele zal de uiteindelijke gedecodeerde tekst bevatten en in de tweede variabele houd je de klinkergroep bij. Wanneer je dan een `p` tegenkomt en je hebt een *niet lege klinkergroep*, kun je die klinkergoep verwerken en enkele posities verder springen.

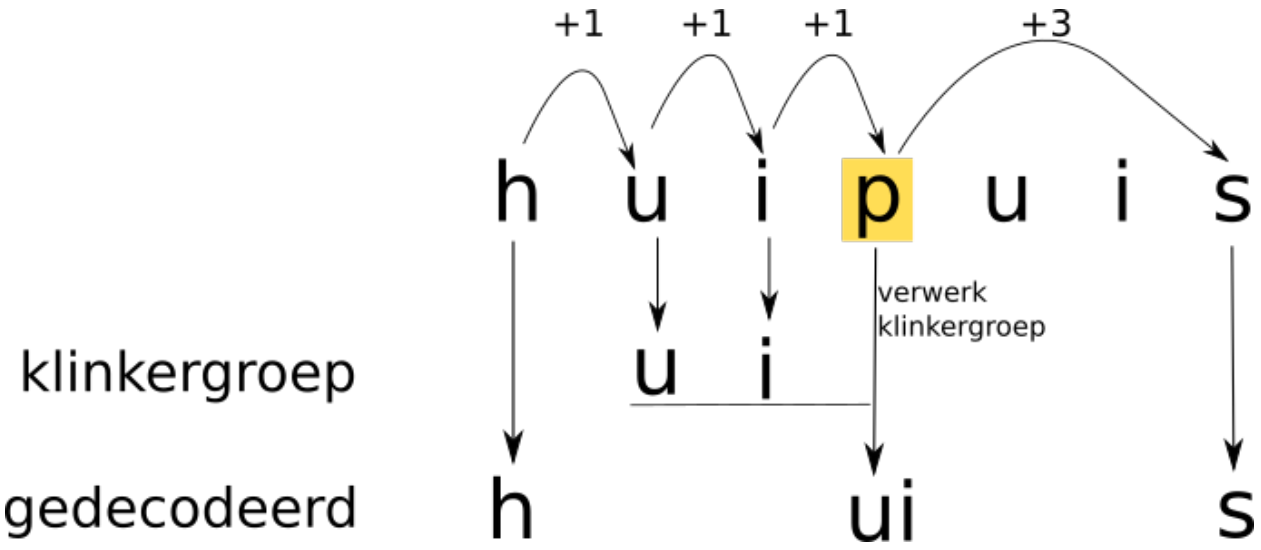

Figure 1: Wepe sprepekepen p