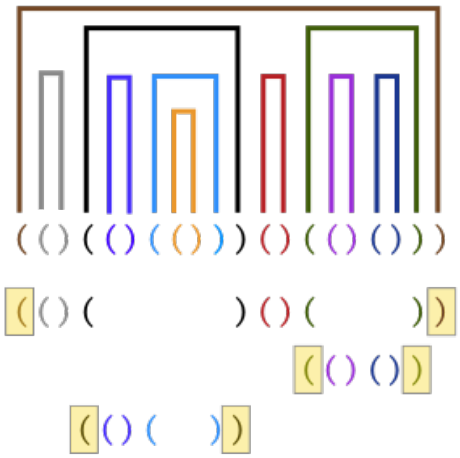
Balanced parentheses

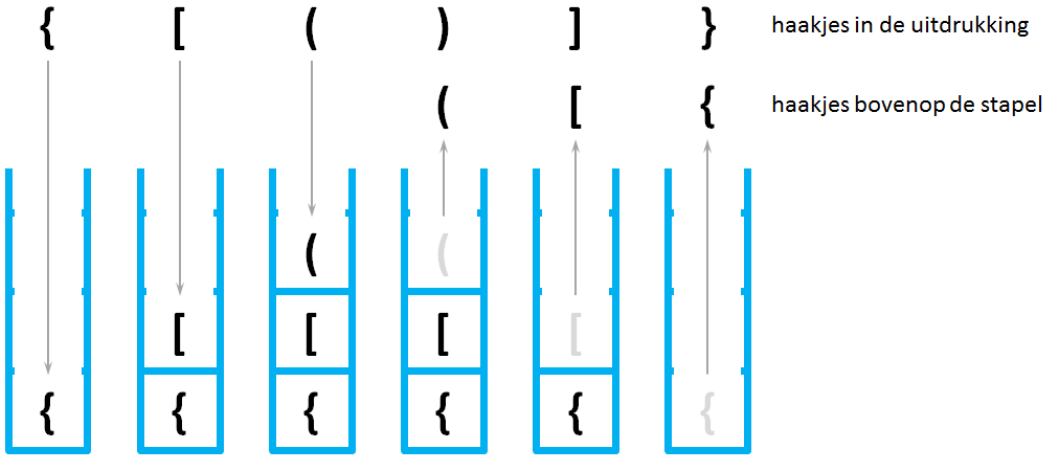
In Python all parentheses should be balanced. That means that each open parenthesis must be closed further down the source code and that each parenthesis that is closed, must have been open already earlier in the source code If the parentheses are not balanced, then this results in a compile error. For example, the illustration below shows which parentheses correspond with each other in the balanced text fragment (()(()(()))()(()())).



Example indicating which parentheses correspond with each other in the balanced expression(() (()(()))()(()())).

Assignment

Consider a given expression in which the following types of parentheses occur: parentheses (and), square brackets [and], and braces {and} and angle brackets <and>. Here we have given the first opening bracket, followed by the closing bracket for each type. In order to determine whether the parentheses in the expression are balanced, a stack can be used. Here, the symbols of the expression are gone through from left to right. Any opening parenthesis that is encountered, is placed on top of the stack. When a closing parenthesis is encountered, then the upper parenthesis is removed from the stack and compared with the closing parenthesis to see if both are of the same type. If that is not the case, then the parentheses are not balanced. Otherwise, the expression is further processed. If the entire expression has been processed, then the parentheses are only balanced when the stack is empty. Otherwise the expression is not balanced.



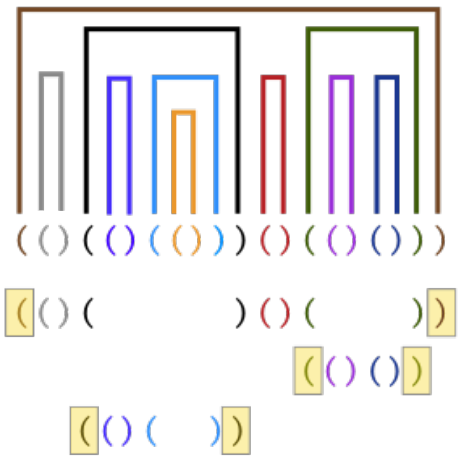
Example that shows how a stack can be used to check whether the brackets in the text fragment {[()]} are balanced.

Write a function balanced to which an expression must be passed as a string argument. The function must return a Boolean value indicating whether the parentheses in the given expression are balanced or not.

Example

```
>>> balanced('{[()]}')
True
>>> balanced('{[(]}')
False
>>> balanced('{[()}')
False
>>> balanced('{[()]')
False
```

In Python moeten alle haakjes gebalanceerd zijn. Dat betekent dat elk geopend haakje verderop in de broncode ook moet afgesloten worden en dat elk haakje dat gesloten wordt, eerder in de broncode reeds moet geopend zijn. Als de haakjes niet gebalanceerd zijn, dan resulteert dit in een compileerfout. Onderstaande illustratie geeft bijvoorbeeld aan welk haakjes met elkaar corresponderen in het gebalanceerde tekstfragment (()(()(()))()(()())).

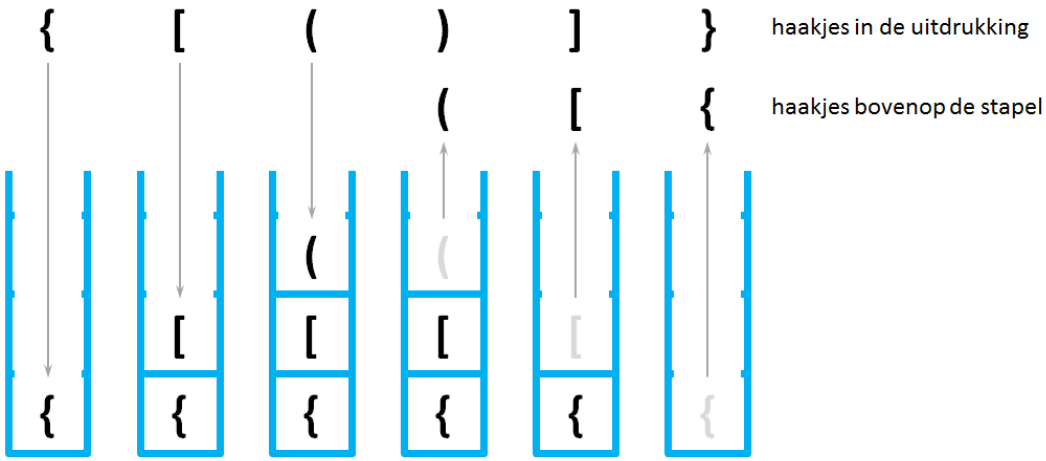


Voorbeeld dat aangeeft welke haakjes met elkaar corresponderen in de gebalanceerde uitdrukking (()(()(()))()(()())).

Opgave

Beschouw een gegeven uitdrukking waarin de volgende soorten haakjes voorkomen: ronde haakjes (en), vierkante haakjes [en], accolades { en } en scherpe haakjes < en >. Hierbij hebben we voor elke soort eerst het openend haakje gegeven, gevolgd door het afsluitend haakje. Om te bepalen of de haakjes in de uitdrukking gebalanceerd zijn, kan gebruik gemaakt worden van een stapel. Hierbij worden de symbolen van de uitdrukking van links naar rechts doorlopen. Elk openend haakje dat men hierbij tegenkomt, wordt bovenop de stapel geplaatst. Wanneer men een afsluitend haakje tegenkomt, dan wordt het bovenste haakje van de stapel gehaald en vergeleken met het afsluitende haakje om te zien of beide van dezelfde soort zijn. Indien dat niet het geval is, dan zijn de haakjes niet gebalanceerd. Anders wordt de uitdrukking verder verwerkt. Als de volledige uitdrukking verwerkt is, dan zijn de haakjes enkel gebalanceerd

als de stapel leeg is. Anders is de uitdrukking niet gebalanceerd.



Voorbeeld dat aangeeft hoe een stapel kan gebruikt worden om na te gaan dat de haakjes in het tekstfragment {[()]} gebalanceerd zijn.

Schrijf een functie gebalanceerd waaraan een uitdrukking als stringargument moet doorgegeven worden. De functie moet een Booleaanse waarde teruggeven, die aangeeft of de haakjes in de gegeven uitdrukking al dan niet gebalanceerd zijn.

Voorbeeld

```
>>> gebalanceerd('{[()]}')
True
>>> gebalanceerd('{[(]}')
False
>>> gebalanceerd('{[()}')
False
>>> gebalanceerd('{[()]')
False
```