```python
def parenthesisMatch(string):
    '''Creating 2 keys, open and closed parenthesis are used as keys
        each key has a value between 0-2. Fitting parenthesis of each
        dict share the same value e.g. '(' and ')' both have the value 0.
        This property will be used later, to check whether a popped item
        from the stack matches to the enqueuing one.'''

    open_par = dict(zip(('(', '[', '{'),range(3)))
    closed_par = dict(zip((')', ']', '}'),range(3)))
    stack = []
    for parenthesis in string:
            if parenthesis in open_par: #1. case: opening parenthesis
                stack.append(parenthesis) #stack it
            elif parenthesis in closed_par and not stack:  #2. case: closing parenthesis, empty stack
                return False
            elif parenthesis in closed_par and stack: #3. case: clsing parenthesis, stack is not empty
                #need to use '!=' instead of 'is not' in case of longer strings!
                if open_par[stack.pop()] is not closed_par[parenthesis]:
                    return False
    else:
        return True and not stack    #explaining negation: bool(x) = False, if x = {0, '', (), [], {}, None};

#Complexity: O(n) since it has to iterate through the whole string.
```

```python
parenthesisMatch('(TALK), (PYTHON()), (()T)O()((M(E)))')
```

```
True
```

```python
parenthesisMatch('(a([c{(d)}]))'), parenthesisMatch('{([(])}')
```

```
(True, False)
```

```python
parenthesisMatch('(wr(o(ng)ma)tching(')
```

```
False
```

```python
parenthesisMatch(')'), parenthesisMatch('('), parenthesisMatch('(()'), parenthesisMatch('[{')
```

```
(False, False, False, False)
```