Funkce k implementaci

brackets_depth():

Zjistí hloubku listů závorek

```
          ({<[]>}[<>])
1...          ()
             / \
2...        {}  []
            |   |
3...        <>  <>
            |
4...        []
```

Validate():

Zjistěte, zda vstupní řetězec obsahuje správný pár otevřených a uzavřených závorek společně s uvozovkami a escape sekvencemi.

```
("as"[<{}>]'df')
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 0
Result = []
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 1
Result = []
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 2
Result = []
```

```
brackets_depth() # +=1 ; -=1

({<[]>}[<>])

Counter = 3
Result = []
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 4
Result = []
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 3
Result = [4]
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 2
Result = [4]
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 1
Result = [4]
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 2
Result = [4]
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 3
Result = [4]
```

brackets_depth() # +=1 ; -=1

```
({<[]>}[<>])

Counter = 2
Result = [4,3]
```

```
brackets_depth() # +=1 ; -=1

({<[]>}[<>])

Counter = 1
Result = [4,3]
```

brackets_depth() # +=1 ; -=1

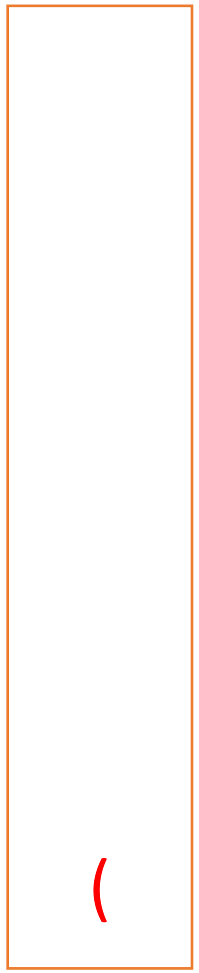
```
({<[]>}[<>])

Counter = 0
Result = [4,3]
```

Validate() # use a stack

```
("as"[<{}>]'df')
```

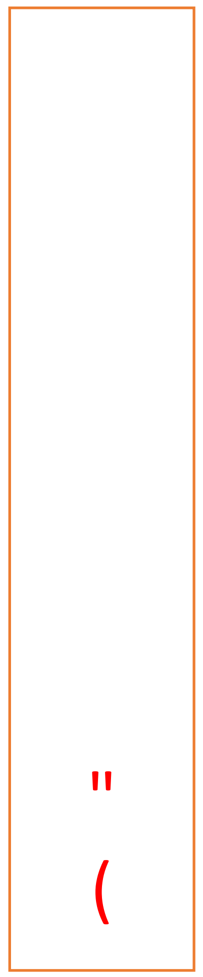
`Validate() # use a stack`

```
("as"[<{}>]'df')
```



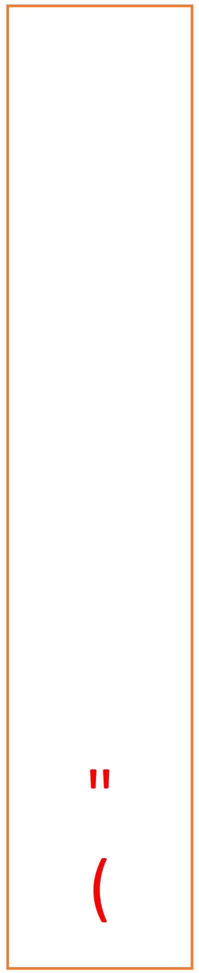
Validate() # use a stack

```
("as"[<{}>]'df')
```



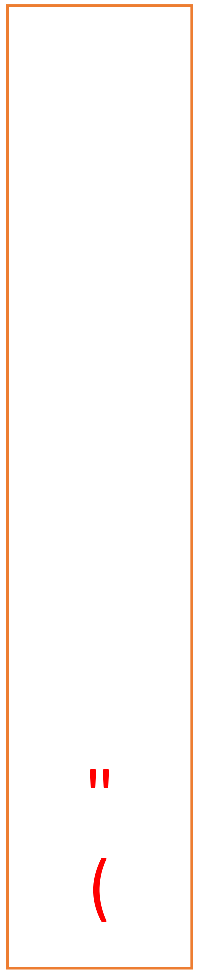
Validate() # use a stack

```
("as"[<{}>]'df')
```



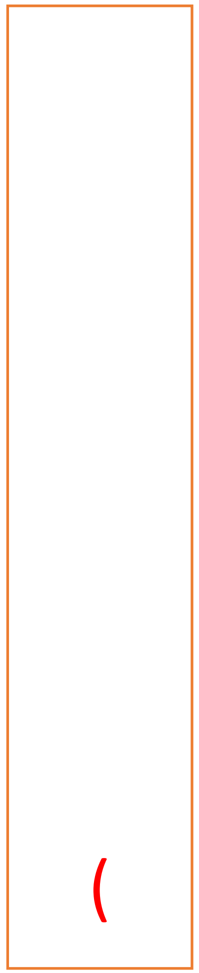
Validate() # use a stack

("as"[<{}>]'df')



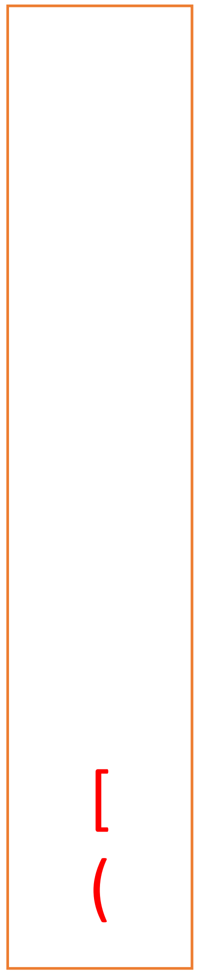
Validate() # use a stack

```
("as"[<{}>]'df')
```



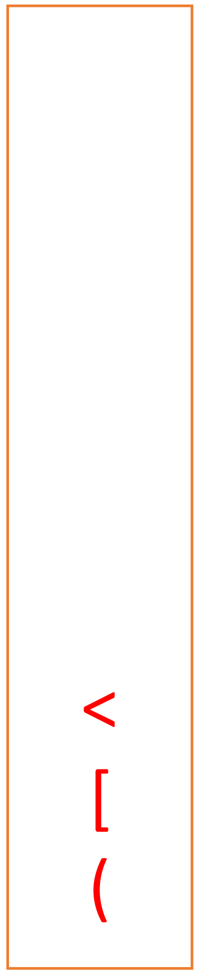
Validate() # use a stack

```
("as"[<{}>]'df')
```



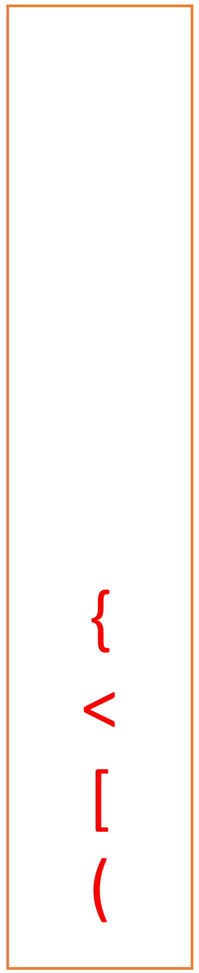
Validate() # use a stack

```
("as"[<{}>]'df')
```



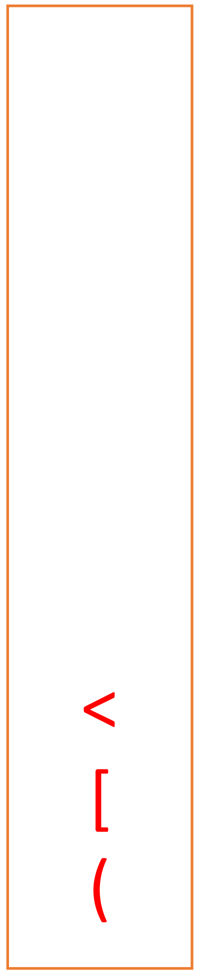
Validate() # use a stack

```
("as"[<{}>]'df')
```



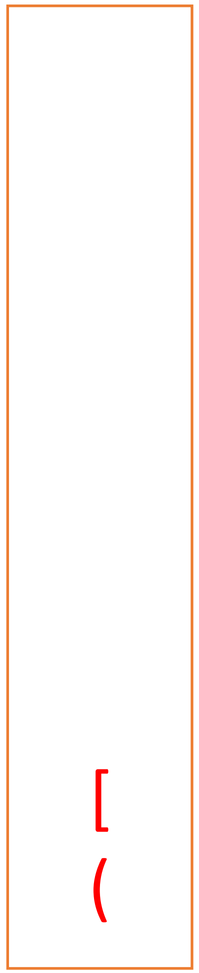
Validate() # use a stack

```
("as"[<{}>]'df')
```



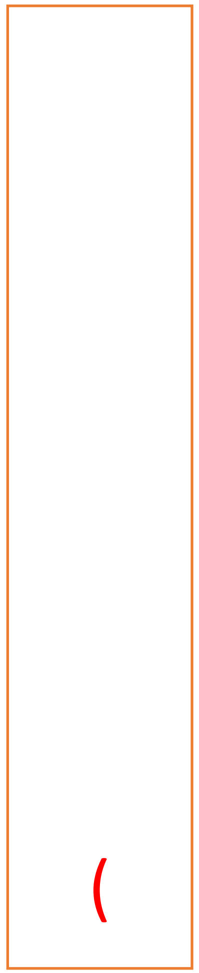
Validate() # use a stack

```
("as"[<{}>]'df')
```



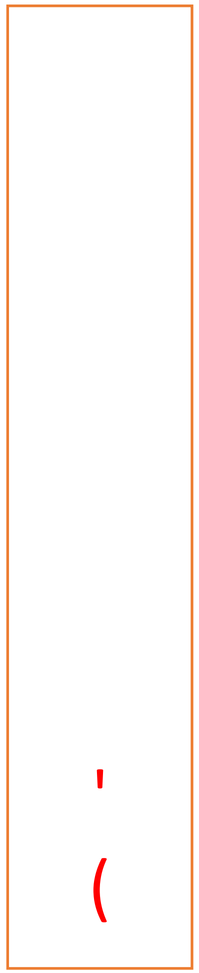
Validate() # use a stack

```
("as"[<{}>]'df')
```



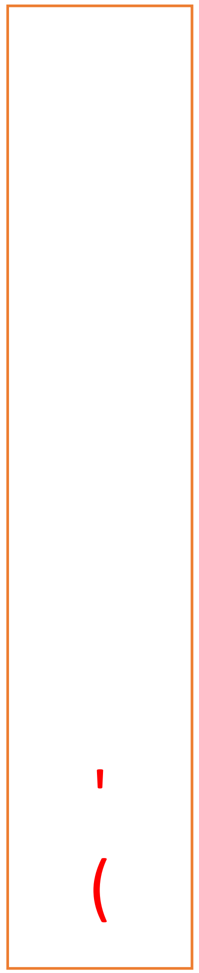
Validate() # use a stack

```
("as"[<{}>]'df')
```



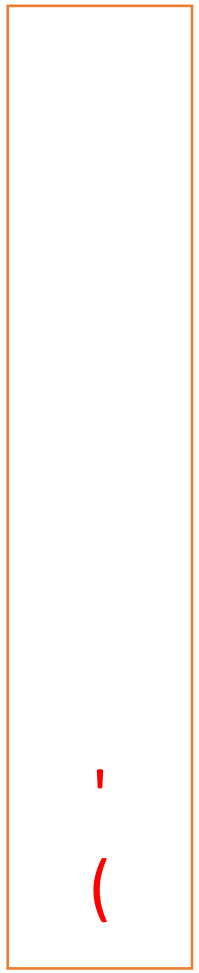
`Validate() # use a stack`

```
("as"[<{}>]'df')
```



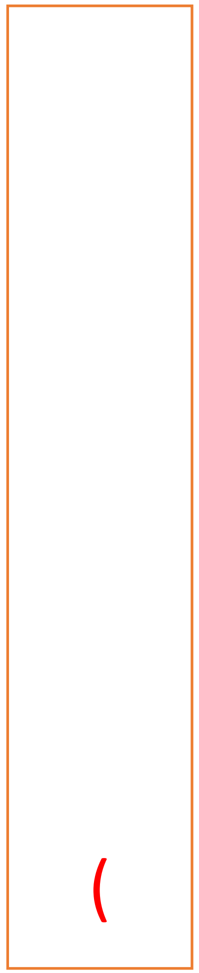
```
Validate() # use a stack
```

```
("as"[<{}>]'df')
```



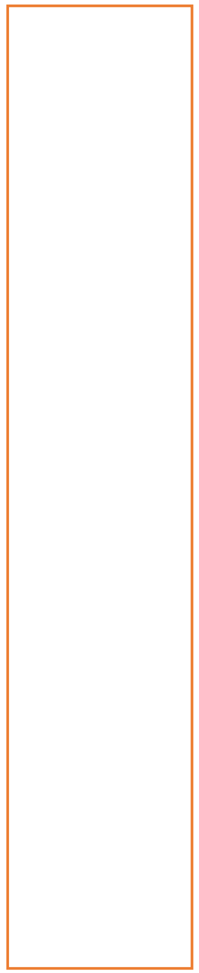
Validate() # use a stack

```
("as"[<{}>]'df')
```



```
Validate() # use a stack
```

```
("as"[<{}>]'df')
```



Validate() # use a stack

```
("as"[<{}>]'df')

return False když:
      narazíte na nesprávnou závorku
      zásobník není na konci prázdný
      není na konci ukončen řetězec
```

Testy

brackets_depth():

test_brackets_depth

Validate():

test_validate_round_brackets

test_validate_brackets

test_validate_quotes

test_validate_mix_quotes

test_validate_escape_quotes

Testy

brackets_depth():

test_brackets_depth

Validate():

test_validate_round_brackets

test_validate_brackets

test_validate_quotes

test_validate_mix_quotes

test_validate_escape_quotes

Enum

```
>>> from enum import Enum, auto
>>> class Color(Enum):
...     RED = auto()
...     BLUE = auto()
...     GREEN = auto()
>>> Color.RED is Color.RED
True
>>> Color.RED is Color.BLUE
False
>>> Color.RED is not Color.BLUE
True
```