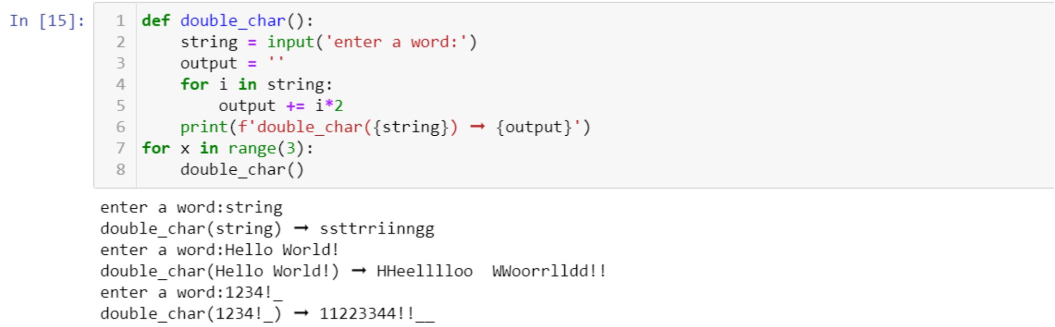
1. Create a function that takes a string and returns a string in which each character is repeated once.

Examples:
double_char("String") ➞ "SSttrriinngg"
double_char("Hello World!") ➞ "HHeelllloo WWoorrlldd!!"
doublechar("1234!_") ➞ "11223344!!__"

Ans:

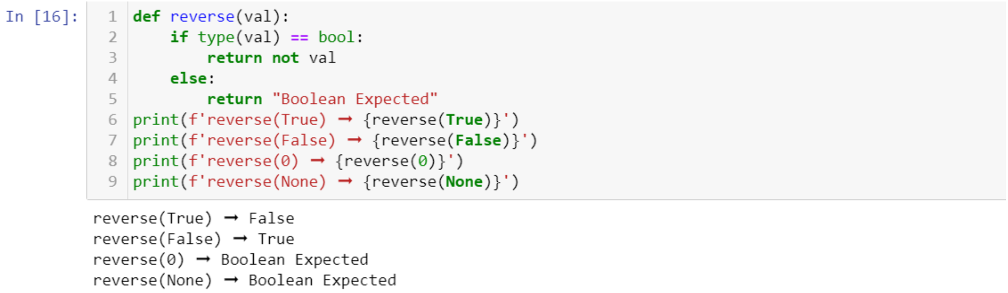


```
In [15]:
def double_char():
    string = input('enter a word:')
    output = ''
    for i in string:
        output += i*2
    print(f'double_char({string}) ➞ {output}')
for x in range(3):
    double_char()
```

```
enter a word:string
double_char(string) ➞ ssttrriinngg
enter a word:Hello World!
double_char(Hello World!) ➞ HHeelllloo  WWoorrlldd!!
enter a word:1234!_
double_char(1234!_) ➞ 11223344!!__
```

2. Create a function that reverses a boolean value and returns the string "boolean expected" if another variable type is given.

Examples:
reverse(True) ➞ False
reverse(False) ➞ True
reverse(0) ➞ "boolean expected"
reverse(None) ➞ "boolean expected"
Ans:



```
In [16]:
def reverse(val):
    if type(val) == bool:
        return not val
    else:
        return "Boolean Expected"
print(f'reverse(True) ➞ {reverse(True)}')
print(f'reverse(False) ➞ {reverse(False)}')
print(f'reverse(0) ➞ {reverse(0)}')
print(f'reverse(None) ➞ {reverse(None)}')
```

```
reverse(True) ➞ False
reverse(False) ➞ True
reverse(0) ➞ Boolean Expected
reverse(None) ➞ Boolean Expected
```

3. Create a function that returns the thickness (in meters) of a piece of paper after folding it n number of times. The paper starts off with a thickness of 0.5mm.

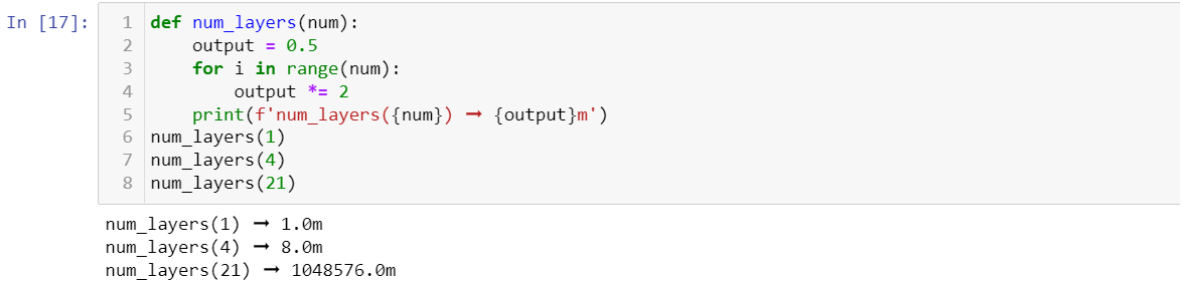
Examples:
`num_layers(1) ➞ "0.001m"

# Paper folded once is 1mm (equal to 0.001m)
num_layers(4) ➞ "0.008m"

# Paper folded 4 times is 8mm (equal to 0.008m)
num_layers(21) ➞ "1048.576m"

Paper folded 21 times is 1048576mm (equal to 1048.576m)`

Ans:



In [17]:

```
def num_layers(num):
    output = 0.5
    for i in range(num):
        output *= 2
    print(f'num_layers({num}) ➞ {output}m')
num_layers(1)
num_layers(4)
num_layers(21)
```

```
num_layers(1) ➞ 1.0m
num_layers(4) ➞ 8.0m
num_layers(21) ➞ 1048576.0m
```

4. Create a function that takes a single string as argument and returns an ordered list containing the indices of all capital letters in the string.

Examples:

index_of_caps("eDaBiT") ➞ [1, 3, 5]

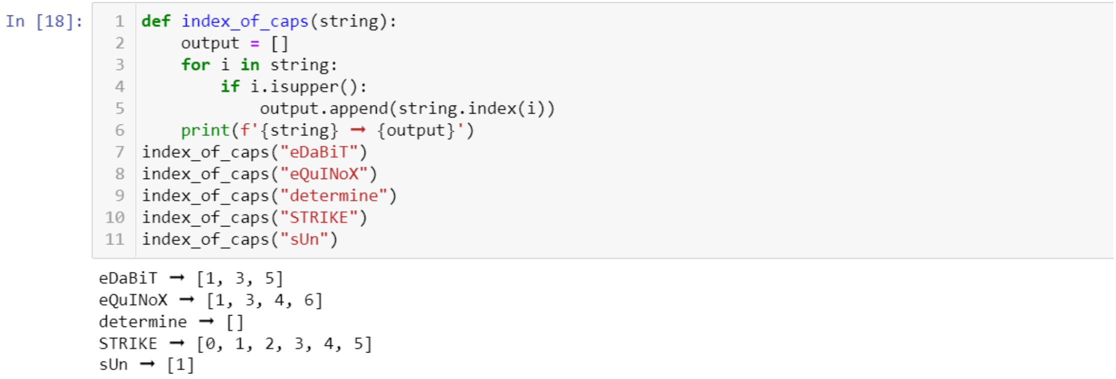
index_of_caps("eQuINoX") ➞ [1, 3, 4, 6]

index_of_caps("determine") ➞ []

index_of_caps("STRIKE") ➞ [0, 1, 2, 3, 4, 5]

index_of_caps("sUn") ➞ [1]

Ans:



In [18]:

```
def index_of_caps(string):
    output = []
    for i in string:
        if i.isupper():
            output.append(string.index(i))
    print(f'{string} ➞ {output}')
index_of_caps("eDaBiT")
index_of_caps("eQuINoX")
index_of_caps("determine")
index_of_caps("STRIKE")
index_of_caps("sUn")
```

```
eDaBiT ➞ [1, 3, 5]
eQuINoX ➞ [1, 3, 4, 6]
determine ➞ []
STRIKE ➞ [0, 1, 2, 3, 4, 5]
sUn ➞ [1]
```

5. Using list comprehensions, create a function that finds all even numbers from 1 to the given number.

Examples:

find_even_nums(8) ➞ [2, 4, 6, 8]

find_even_nums(4) ➞ [2, 4]

find_even_nums(2) ➞ [2]

Ans:

In [1]:

```python
def find_even_nums(n):
    output = [i for i in range(1,n+1) if i%2 == 0]
    print(f'find_even_nums({n}) ➞ {output}')
find_even_nums(8)
find_even_nums(4)
find_even_nums(2)
```

```
find_even_nums(8) ➞ [2, 4, 6, 8]
find_even_nums(4) ➞ [2, 4]
find_even_nums(2) ➞ [2]
```