

Assignment 19 Solutions

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

Examples:

`double_char("String")` → "SSttrriinnngg"

`double_char("Hello World!")` → "HHeellllloo WWoorrlldd!!"

`doublechar("1234! ")` → "11223344!!__"

In [2]:

```
def double_char(in_string):
    out_string = ''
    for ele in in_string:
        out_string += ele*2
    return out_string

print(f'→ {double_char("String")}')
print(f'→ {double_char("Hello World")}')
print(f'→ {double_char("1234! _")}')

→ SSttrriinnngg
→ HHeellllloo WWoorrlldd
→ 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"

In [3]:

```
def reverse(in_bool):
    if type(in_bool) == bool:
        return not in_bool
    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)
```

```
In [4]: def num_layers(in_num):
        out_num = 0.5
        for ele in range(in_num):
            out_num *=2
        print(f'Output → {out_num/1000}m')

        num_layers(1)
        num_layers(4)
        num_layers(21)
```

```
Output → 0.001m
Output → 0.008m
Output → 1048.576m
```

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]
```

```
In [5]: def index_of_caps(in_string):
        out_string = []
        for ele in in_string:
            if ele.isupper():
                out_string.append(in_string.index(ele))
        print(f'{in_string} → {out_string}')

        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]
```

```
In [6]: def find_even_nums(in_num):
        out_list = [i for i in range(1,in_num+1) if i%2 == 0]
        print(f'Output → {out_list}')

        find_even_nums(8)
        find_even_nums(4)
        find_even_nums(2)
```

```
Output → [2, 4, 6, 8]
Output → [2, 4]
Output → [2]
```

In []:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js