Assignment 3

Exercise 1

```
"Exercise 1"
def recursive_max(vals: list):
    if len(vals) == 1:
        return vals[0]
    else:
        smallest = min(vals[0], vals[1])
        vals.remove(smallest)
        return recursive_max(vals)
```

Exercise 2

```
"Exercise 2"
def recursive_fibonacci(val: int):
    if val < 2:
        return val
    else:
        return recursive_fibonacci(val-2) + recursive_fibonacci(val-1)
```

The recursive fibonacci function suffers from having to perform an increasing amount of redundant calculations as the value passed in increases. when we call recursive_fibonacci(val-2), the second time recursive_fibonacci(val-1) is called, the operation being done is the same

Iterative fibonacci simply performs calculations on the input and doesn't suffer from this redundancy.

Exercise 3

```
"Exercise 3"
def recursive_sum(n: int, subtract: int = 0):
    if n<1:
        return 0
```

```
    else:
        return n + recursive_sum(n-subtract, subtract+2)
```

Exercise4

```
"Exercise 4"
def recursive_power(a: int, pow: int):
    if pow == 0:
        return 1
    else:
        return a * recursive_power(a, pow-1)
```

Function Calls

```
clt.header("Maxmimum")
print(recursive_max([1,55,23429,123,523,42]))

clt.header("Fibonacci")
print(recursive_fibonacci(36))
print(recursive_fibonacci(10))


clt.header("Sum")
print(recursive_sum(55))

clt.header("power")
print(recursive_power(2,5))
```

Output

```
Maxmimum:
23429

Fibonacci:
14930352
55

Sum:
```

```
328

Power:
32
```