1 FUNctions

1.1 What would Python display?

```python
>>> from operator import add, mul
>>> def f(x, y):
...     print('h', y)
...     return add(add(x, y), 1)
>>> def g(x, y):
...     print('u', x)
...     return mul(mul(x, y), 2)
>>> f(2, g(2, f(500, 2)))
```

```
h 2
u 2
h 2012
2015
```

2 Types

2.1 What would Python display for each expression?

(a) 3 + 4

```
7
```

(b) '3' + 4

```
TypeError
```

(c) '3' + 4'

```
'3' + 4'
```

(d) '3' + '4'

```
'34'
```

2.2 Why does Python behave this way?
Each value in Python has a type. When we perform “addition”, Python looks at the type of each value to figure out what to do. When both values are numbers, then we compute the sum of the numbers. When both values are strings, then we join the strings. However, Python doesn’t know how to add numbers to strings so it says, TypeError.

3 Moar FUNctions

3.1 >>> print(add(6, 1), print(7))

7
7 None

3.2 >>> print(print("Welcome to"), print("CS 61A"))

Welcome to
CS 61A
None None

User-defined Functions

3.3 >>> def boom():
...     return 1 / 0
...

>>> boom

<function boom at ...>

>>> boom()

ZeroDivisionError

3.4 >>> def square(x):
...     return x * x
...

>>> def happy(num):
...     print(num)
...     return num
...

>>> def sad(num):
...     return num
...     return 1 / 0
...

>>> print(square(sad(5)), happy(3))

3
25 3
4 Beep Boop Bam

4.1 What would Python display?

```python
>>> x = 6
>>> def beep(x):
...     print(x)
...
>>> def boop(x):
...     y = x
...     x = 7
...     print(x)
...
>>> y = beep(6)
6
>>> boop(28)
7
>>> y + beep(8)
```

8

```
TypeError: unsupported operand type(s) for +: 'NoneType' and 'NoneType'
```

Note that neither function has a return statement, so they will both return None by default.