BEGINNERS' PYTHON - FUNCTIONS

LEWIS GAUL

ST EDMUND HALL

4TH YEAR MATHEMATICS

LEWIS.GAUL@SEH.OX.AC.UK

LIST COMPREHENSIONS

- Simple way to construct lists using one or more conditions.
- Useful for 'flattening' lists.
- e.g. `[num for num in range(10) if num**2 < 50],`

```
[i**0.5 for i in range(6)],

[i for row in [[1, 2], [3, 4]] for i in row]
```

INTRODUCTION TO FUNCTIONS

- Called using parentheses, often with arguments inside
- Can be defined to take a fixed number of arguments
- Arguments can be any data type (any object)
- Arguments can be made optional by giving default values
- Can alternatively be defined to take any number of arguments

SOME USEFUL FUNCTIONS

```
len(list/str)
sorted(list)
reversed(list)
sum(list)
max/min(list)
range([start],stop,[step])
round(float)
int(str/float) e.g. int('3')
```

Methods:

```
list.sort()
list.append(object)
list.pop(index)
list.extend(list)  list/
str.count(object) list/
str.index(object)
str.join([list of strings])
```

DEFINING YOUR OWN FUNCTIONS

Try calling the functions below,
e.g. `my_func(2, 5)`
e.g. `cube(1.2)`

```
def [function name]([arguments]):
    [do this]
    return [something] (optional)

[name] = lambda [args]: [do this]
```

```
def my_func(x, y):
    result = x + y // x
    return result

cube = lambda x: x**3
```

FUNCTION ARGUMENTS

```
def get_max(arg1, arg2=0, arg3=1):
  print(arg1, arg2, arg3)

func(3); func([]); func(arg1=‘hello’)
func(5, 6);
func(-1, arg2=0); func(0.5, arg3=-3)
func(3, 2, 1); func(1, arg3=False, arg2=0)
```

CHALLENGE 3

- Go to github.com/LewisGaul/python-tutorial, download `challenge3.py`
- Work out how the code works (try adding in some print statements)
- Write comments with '#' to explain how it works
- Can you improve the code?
- When you understand it all have a go at the challenge
- Try to use sensible variable names
- Avoid using too many indented layers or repeating code