Session 3 Handout

1. Defining and Calling your Own Functions

```
[1]: def calculateWage(hours,base=10,bonus=.5):
         ''' Calculates weekly wage '''
         if hours<=40:
             pay=hours*base
         else:
             pay=hours*base+(hours-40)*base*bonus
         return pay

     help(calculateWage)
     print('Pay for 42 hours with default base and bonus:',calculateWage(42))
     print('Pay for 42 hours with base 12/hour and default bonus:',calculateWage(42,12))
     print('Pay for 42 hours with base 12/hour and bonus 60%:', calculateWage(42,12,.6))
     print('Pay for 42 hours with default base and bonus 50%:',calculateWage(42,bonus=0.6))
```

```
Help on function calculateWage in module __main__:

calculateWage(hours, base=10, bonus=0.5)
    Calculates weekly wage

Pay for 42 hours with default base and bonus: 430.0
Pay for 42 hours with base 12/hour and default bonus: 516.0
Pay for 42 hours with base 12/hour and bonus 60%: 518.4
Pay for 42 hours with default base and bonus 50%: 432.0
```

Q1: (Modification of case 2 from last session) Write a function named `orderQuantity` that takes two input arguments, `inventory` and `basestock`. If `inventory` is at least equal to `basestock`, then return 0. Otherwise, return the difference between `basestock` and `inventory`. Set the default value for `inventory` to be 0 and for `basestock` to be 100. Include an appropriate docstring to explain what the function does.

```
[3]: # Code to test your function
     help(orderQuantity)
     print(orderQuantity())
     print(orderQuantity(25))
     print(orderQuantity(51,50))
     print(orderQuantity(basestock=200))
     print(orderQuantity(inventory=80))
```

```
Help on function orderQuantity in module __main__:

orderQuantity(inventory=0, basestock=100)
    Calculates order quantity given inventory level and basestock level

100
75
0
200
20
```

Q2: Walk through the code to explain each line of the above output.

Packaging functions within a module

Open Spyder and create a new Python script, and copy paste the two functions `calculateWage` and `orderQuantity` into the script. Save the script into the same folder as this notebook, as `session3.py`. If everything is correct, you should be able to run the following.

```
[4]: import session3
     print(session3.calculateWage(40))
     print(session3.orderQuantity(30))
     help(session3.orderQuantity)
```

```
400
70
Help on function orderQuantity in module session3:

orderQuantity(inventory, basestock=100)
    Calculates order quantity given inventory level and basestock level
```

```
[ ]: print('Module contains the following variables and functions:', dir(session3))
```

```
[ ]: help(session3)
```

2. Exploring Existing Functions

```
[7]: help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

```
[ ]: print(dir(__builtins__))
```

Q3: Run the above line and out of the items in all lowercase, choose five that look interesting to you, and use `type` and `help` and trial and error to find out what each of these built-in objects are and what you can do with them. Explain to your neighbor.

Q4: Import the `math` module and print the list of variables and functions within this module using `dir`. Choose five functions from this list and use `help` and trial and error to figure out how to use them. Explain to your neighbor.

Q5: Use `dir` on the string object `"Hi"`. Choose five functions from this list and use `help` and trial and error to figure out how to use these functions built in to every string object. Explain to your neighbor.

Case 6a. Mortgage Calculator I

Write a function `numberMonths` in module `session3` that calculates how many months it would take to pay off a mortgage given the monthly payment. The function has four input arguments: `total`, `monthly`, `interest`, and `downpay`. Let the default values for `interest` be 0.0425 and for `downpay` be 0. Label the four arguments T, M, I, D respectively. The number of months needed N is given by the formula

$$N = ceil\left(\frac{-\log(1-\frac{i(T-D)}{M})}{\log(1+i)}\right),$$

where $i = I/12$ is the monthly interest rate and *ceil* is the `math.ceil` function. (Note, after modifying the `session3.py`, you will have to restart the kernel using the toolbar above to reload the latest version.)

```
[20]: import session3 as s3
      s3.numberMonths(500000,4000)/12
```

```
13.833333333333334
```

```
[21]: s3.numberMonths(500000,4000,interest=0.05)/12
```

```
14.75
```

Case 6b. Mortgage Calculator II

Write a function `monthlyPayment` in module `session3` that calculates the monthly payment needed to pay off a mortgage in a given number of months. The function has four input arguments: `total`, `months`, `interest`, and `downpay`. Let the default values for `interest` be 0.0425 and for `downpay` be 0. Label the four arguments T, N, I, D respectively. The monthly payment M is given by the formula

$$M = \frac{(1+i)^N}{(1+i)^N - 1} i(T-D),$$

where $i = I/12$ is the monthly interest rate. Round the answer to two decimal places using the `round` function.

```
[22]: s3.monthlyPayment(500000,12*30)
```

```
2459.7
```

```
[23]: s3.monthlyPayment(500000,12*30,interest=0.05)
```

```
2684.11
```