Session 3. Reading Documentation and Debugging

1. Reading Documentation

Method 1: Using `help` and `dir` within Python.

```
[1]: 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.

[2]: import math
     help(math.ceil)

Help on built-in function ceil in module math:

ceil(...)
    ceil(x)

    Return the ceiling of x as an Integral.
    This is the smallest integer >= x.

[3]: dir(math)

['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin',
 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf',
 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma',
 'gcd',  'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log',
 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt',
 'tan', 'tanh', 'tau', 'trunc']

[ ]: dir(__builtin__)
```

Method 2: Search on Google.
Examples of what to search:

- How to round up in Python?
- f-string formatting in Python 3
- Tutorial on functions in Python 3.
- "[Copy/Paste the error message and enclose with quotes]"

Q1. Mortgage Calculator I

Write a function `numberMonths` 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`, `annualInterest`, 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.

```
[5]: print('Number of years needed to pay off mortgage:', numberMonths(500000,4000)/12)
```

```
Number of years needed to pay off mortgage: 13.833333333333334
```

```
[6]: print('Updated number of years:', numberMonths(500000,4000,interest=0.05)/12)
```

```
Updated number of years: 14.75
```

2. Debugging

The following cells illustrate an efficient method of debugging based on isolating the problem and reproducing it in the simpliest manner.

```
[7]: import math
     def numberMonths(total,monthly,interest=0.0425,downpay=0):
         T=total
         M=monthly
         I=interest
         D=downpay
         return math.ceil(-math.log(1-I(T-D)/M)/math.log(1+I))
     numberMonths(500000,4000)/12
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

<ipython-input-7-41abbd54549f> in <module>()
  6     D=downpay
  7     return math.ceil(-math.log(1-I(T-D)/M)/math.log(1+I))
----> 8 numberMonths(500000,4000)/12

<ipython-input-7-41abbd54549f> in numberMonths(total, monthly, interest, downpay)
  5     I=interest
  6     D=downpay
----> 7     return math.ceil(-math.log(1-I(T-D)/M)/math.log(1+I))
  8 numberMonths(500000,4000)/12

TypeError: 'float' object is not callable
```

A. Recreating the error outside of the function

```
[ ]: T=500000
     M=4000
     I=0.0425
     D=0
     math.ceil(-math.log(1-(I/12)*(T-D)/M)/math.log(1+(I/12)))
```

B. Dissecting the line containing the error

```
[ ]: T-D
```

```
[ ]: I(T-D)
```

```
[ ]: I*(T-D)/M
```

```
[ ]: 1-I*(T-D)/M
```

```
[ ]: math.log(-4.3125)
```

C. Correcting the logic (this time building up one component at a time)

```
[ ]: T=500000
     M=4000
     I=0.0425
     D=0
     math.ceil(-math.log(1-(I/12)*(T-D)/M)/math.log(1+(I/12)))
```

D. Putting correct logic back into function

```
[ ]: import math
     def numberMonths(total,monthly,interest=0.0425,downpay=0):
         T=total
         M=monthly
         I=interest
         D=downpay
         return math.ceil(-math.log(1-(I/12)*(T-D)/M)/math.log(1+(I/12)))
     print('Number of years needed to pay off mortgage:', numberMonths(500000,4000)/12)
```

(Optional: Shortening the code to work with original named variables directly.)

```
[ ]: import math
     def numberMonths(total,monthly,interest=0.0425,downpay=0):
         i=interest/12
         A=i*(total-downpay)/monthly
         top=-math.log(1-A)
         bottom=math.log(1+i)
         return math.ceil(top/bottom)
     print('Number of years needed to pay off mortgage:', numberMonths(500000,4000)/12)
```

Q2. Mortgage Calculator II

Write a function `monthlyPayment` 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.

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

```
2459.7
```

```
[31]: monthlyPayment(500000,12*30,interest=0.05)
```

```
2684.11
```