Session 2: Reading Documentation and Debugging

A. Exploring Existing Functions

```
[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.
```

Google "Python print examples" for examples of how to use this function. For advanced formatting, Google "Python f-string formatting."

Q1: Write a single line to print the following multi-line output based on the given variables. Your command must not hardcode the variable values but should work if the values are changed. Each line after "Output:" should be prefixed by a tab instead of by spaces. (Hint: Google "Python f-string formatting decimal places" and "Python print tab new line.")

```
[3]: # Given variable values
     total=16
     count=3

[4]:
```

```
Output:
        total: 16,
        count: 3
        average (2 decimal places): 5.33.
```

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

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', '
```

```
[5]: type(pow)
```

```
builtin_function_or_method
```

```
[6]: type(str)
```

```
type
```

```
[7]: type(__import__)
```

```
builtin_function_or_method
```

```
[8]: import math
     print(dir(math))
```

```
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh',
```

Q2: Identify three built-in objects/functions that look interesting to you. Use `type`, `help`, Google and trial and error to find out what each of these are and what you can do with them. Explain to your neighbor. Do the same with three objects/functions in the math module.

Q3: Google "Python list documentation" and "Python dict documentation." Based on what you find, explain to your neighbor the difference between lists and dictionaries, as well as how to create them, add elements, access elements, and iterate through them using a for loop.

Case 3a. Mortgage Calculator I

Write a function `numberMonths` 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.

```
[10]: numberMonths(500000,4000)/12
```

```
13.833333333333334
```

```
[11]: numberMonths(500000,4000,interest=0.05)/12
```

```
14.75
```

(Optional): use f-string formatting to display the outputs in a more readable sentence.

B. Debugging

```
[12]: 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-12-41abbd54549f> 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-12-41abbd54549f> 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(T-D)/M)/math.log(1+I))
```

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)

[16]:

```
166
```

D. Putting correct logic back into function

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

[17]:

```
13.833333333333334
```

Case 3b. 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.

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

```
2459.7
```

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

```
2684.11
```