EMAT10007 – Introduction to Computer Programming

Exercises – Week 4. Functions

Reminder: The exercise sheets use `<?>` as a placeholder to represent something missing - this could be an operator, a variable name, function name, etc. Multiple `<?>` in the same question are not necessarily representing the same thing.

Part 1 Functions

Exercise 1 - Functions (Essential)

1. Can you complete the `RetSquared` function below by replacing the blank `<?>` with the correct variable name?

```
def RetSquared(Number):
    # Returns the square
    Squared = <?> ** 2
    return <?>
```

2. Use the `RetSquared` function and a `for` loop to print the squares of all integers from 1 to 10.
3. Can you write a general function for raising numbers to powers? The function should take two arguments `Number` and `Power` and return the number raised to the given power. Remember to give your new function a sensible name.
4. Use your new general function to print the powers of 2 up to 2^{10}.
5. Change this function so that the argument `Power` has the default value `1`.
6. Write a new function which has arguments `Number` and `Powers`. If I have inputs `Number=2` and `Powers=[1, 3, 4]`, the function should return the list `[2, 8, 16]`.
   How else can you write a function when you don't know the exact number of inputs?
7. Write a function called `inverse` that computes the inverse of a number x. That is, the function returns $1/x$ if $x \neq 0$. If $x = 0$ then your function should return the string "undefined". Show that your function works by computing `inverse(2)` and `inverse(0)`.
8. Write a function that swaps the values of two variables. Make sure your function returns the new values. Include a docstring that explains how to call the function. Use the `help` function to print the docstring and read about your function. Then, create two variables `a = ['red', 'blue', 'green']` and `b = {1, 3, 5}` and use your function to swap their values.

Exercise 2 - Modelling with functions (Essential)

Calculating the area under a curve given by $y(x)$ is important for many engineering problems. Calculus tells us that the area A under the curve $y(x)$ between the points $x = a$ and $x = b$ is given

by the integral

$$A = \int_a^b y(x)\,dx. \tag{1}$$

However, in real-world engineering problems, the formulas for the curves are often so complex that it is impossible to carry out the integration exactly. Therefore, the area under the curve must be found approximately. One way to approximate the area under the curve is by the trapezium rule, which divides the area under a curve into N trapezoids. According to the trapezium rule, the area under the curve can be approximated by the formula

$$A \approx \frac{1}{2}[y(a) + y(b)]\Delta x + \sum_{i=1}^{N-1} y(x_i)\Delta x, \tag{2}$$

where $\Delta x = (b-a)/N$ and $x_i = a + i\Delta x$.

1. Write a Python program that uses the trapezium rule to calculate the area under the curve $y = x^2$ between $x = 0$ and $x = 1$ by setting $N = 1$, $N = 10$, and $N = 100$. Use a Python function to define the curve $y(x)$. In this case, the area can be calculated exactly as $A = 1/3$. Use this result to check that your code is working correctly. You should find that as N increases, the value of A calculated from the trapezium rule becomes more accurate.
2. The density of a hot bar of length $L = 0.1$ m is given by the function

   $$y(x) = \alpha e^{-x^2/\beta^2}, \tag{3}$$

   where x is a point in the bar, $\alpha = 2$ kg m^{-1} and $\beta = 0.2$ m are constants. Assuming that $0 < x < L$, calculate the mass of the bar. Hint: the mass of the bar can be determined by calculating the area under the curve from $x = 0$ to $x = L$. Use the code `from math import *` to access the exponential function. The exponential of a variable `x` can be computed using the code `exp(x)`.
3. Write a Python function that calculates the area under any curve $y(x)$ between $x = a$ and $x = b$ using the trapezium rule. Your Python function should allow the equation for the curve $y(x)$ to depend on parameters. Hint: functions can be passed to other functions as arguments.

Part 2. Functions arguments and scope

Exercise 3 - Variadic functions (Essential)

Variadic functions can take any number of arguments, just like the print function, which works when used in the following way:

```
print("Hello, I am", Age, "years old, born in the month of", Month, ".")
```

1. Write a function that takes an unknown amount of numbers (integers or floats) and then:
   - Calculates the sum of the numbers
   - Calculates the product of all of the numbers

   and **returns** these values as opposed to printing them from the function.

2. Write a function that takes an unknown amount of numbers as its arguments and returns the `min`, `max` and `average` of those numbers.

Exercise 4 - Variable scope (Essential)

Variable scope. It is important to understand a little bit about **scope** – an important concept in programming. The following exercises will demonstrate how variables are treated depending on whether they are declared inside (local scope) or outside (global scope) of a function.

1. Examine the following code and predict the value(s) printed to the screen. Then run the code to check your prediction.

```
def F():
    print(S)

S = "I hate spam"
F()
```

2. Examine the following code and predict the value(s) printed to the screen. Then run the code to check your prediction.

```
def F():
    S = "Me too"
    print(S)

S = "I hate spam"
F()
print(S)
```

3. Examine the following code and predict the value(s) printed to the screen. Then run the code to check your prediction.

```
def F():
    global S
    S = "Me too"
    print(S)

S = "I hate spam"
F()
print(S)
```

4. Examine the following code and predict the value(s) printed to the screen. Then run the code to check your prediction.

```
def F():
    T = "I am a variable"
    print(T)

F()
print(T)
```

Part 3. Recursive functions

Exercise 5 - Recursive functions (Essential)

1. Write a recursive function that computes the factorial of a positive integer n. Recall that the factorial is defined as $n! = n \times (n-1) \times \ldots \times 1$. By definition, $0! = 1$. Check your code works by computing $5! = 120$. **Hint:** Notice that $n! = n \times (n-1)!$.
2. Write a recursive function called `Fib(n)` that calculates n-th number in the Fibonacci sequence using a recursive function. The following code

   ```
   for i in range(1,15):
       print(Fib(i), end=" ")
   ```

   should print:
   `1 1 2 3 5 8 13 21 34 55 89 144 233 377`
   If you are struggling, start by researching how to calculate the Fibonacci sequence. There are lots of tutorials online, so be sure to search for implementations of the Fibonacci function using *recursion*. **Hint:** You will need to identify the "special cases" of your program which prevent the function from calling itself infinitely many times.
3. Read about the relative advantages and disadvantages of using *recursive* functions i.e. what happens if `n >= 100`? Is there another version of the Fibonacci function that does not use recursion? Can this be called on large values of `n`?

Part 4. Advanced questions

Exercise 6 - More functions

1. Write a function that computes the median of three distinct numbers that are input in arbitrary order. Recall that the median in the number in the middle; it's neither the largest nor the smallest. For example, the median of 4, 1, and 2 is 2.
2. Write a function called `sum_digits` that computes the sum of the digits of an integer. For example `sum_digits(1234) = 1 + 2 + 3 + 4 = 10`. There are several ways of doing this; however, if you want a challenge, try to do this using only mathematical operations.
3. In Python it's easy to sort a list of numbers `L` using the `sort` method, that is, by calling `L.sort()`. Write your own function that sorts a list with an arbitrary number of numbers.
4. Write a function which returns the prime factorization of a number. **Note:** Learn about prime factorization here: `https://www.mathsisfun.com/prime-factorization.html`