THE UNIVERSITY OF SYDNEY

# INFO1110 / COMP9001 Week 13 Tutorial

**Revision**

## Feedback

Please take some time to provide feedback to the unit of study

http://sydney.edu.au/itl/surveys/complete/

## Revision questions

The following questions have been picked from previous tutorials to help you prepare.

## Question 1: Convert list to a string

Without invoking the implicit string function on list and using a loop. Write a function called `list_to_string(items)` that will iterate through all the elements in a list and create a string of all the items.

```
l = [1, 2, 3, 4, 5, 6]
s = list_to_string(l)
print(s)
```

Your program should output:

```
1, 2, 3, 4, 5, 6
```

## Question 2: Finding the length of a string

You are tasked with writing a program (without using `len()`) that will find the length of a string. You will call this function:

```
def py_len(s)
```

Remember that you can iterate through a string using a `for` loop.

## Question 3: Reverse a string

When given a string as input, reverse it and exit. You may only use a while loop for this question (no slices or reversed function).

```
python reverse
Yellow
wolleY
```

## Question 4: Your own reversed iterator

Construct an iterator that will move through a list or string in reverse.

```
l = [1, 2, 3, 4, 5]
for i in my_reversed(l):
        print(i) #5, 4, 3, 2, 1
```

## Question 5: Find a number sequence

You are to write a function that will find a number sequence in a list of numbers. For example, given the sequence `[3, 4, 5]` you will need to find if the exact sequence exists in the list of numbers `[6, 8, 7, 4, 2, 3, 4, 5, 7, 8]`.

```
def find_seq(input_list, sequence)
```

Given the following examples:

```
print(find_seq([1, 6, 4, 2, 7, 4, 2, 6, 3], [4, 3])) # True
print(find_seq([1, 6, 4, 2, 7, 4, 2, 6, 3], [2, 6, 3, 3])) # False
```

# Question 6: Password generator

Using the random module and the function `randint`. Your program should utilise multiple character arrays containing letters, numbers and symbols to produce a password.

The length of the password will be specified as a command line argument.

```
python pass_gen.py 8
kl^f832n
```

# Question 7: Bouncy String

Using a while loop, you will write a program that will take a command line argument, starting position and index and output the characters in that sequence. During the iteration the index is at the last element it should start iterating in reverse until it reaches the first element.

Example:

```
python bouncy.py quokkas 3 6
kkasak
```

Example:

```
python bouncy.py jumbo 4 9
obmujumbo
```

# Question 8: Hailstones

Implement the Hailstone number sequence as a generator, outputs all the numbers until it reaches 1.

```
def gen_hailstone(n)
```

- If the number is even divide it by two
- If the number is odd multiply it by three and add one.

Test your code with the following statements:

```
for i in hailstone(5)
        print(i) # 5, 16, 8, 4, 2, 1
```

# Question 9: Matrix Multiplication

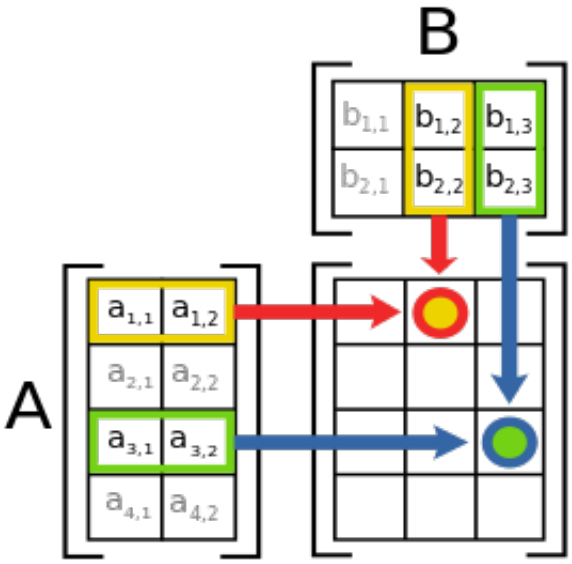
Matrix multiplication is a mathematical procedure that produces a resultant matrix from two input matrices. Given the following function prototype:

```
def matrix_mult(m1, m2)
```

Mathematically, the multiplication of $A \times B$ can be expressed as:

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj} \,.$$

where A is an $n \times m$ matrix and B is an $m \times p$ matrix. Graphically, this can be seen as:




Implement your own matrix multiplication function without using numpy. After implementing your own, use the numpy's `numpy.matmult` function while creating arrays using `np.array([1,2,3,4,5], dtype=np.int32)`.

```
matrix_mult([[1,2],[3,4]], [[2,0],[1,2]]) #[[4,4], [10,8]]
matrix_mult([[1, 2, 3], [4, 5, 6]], [[9, 8, 7]]) # [[46, 118]]
```

# Question 10: Sudoku Verifier

You are tasked with writing a program that will verify that a sudoku puzzle is solvable. If you are unfamiliar with a sudoku, Sudoku is a puzzle game that requires the player to place numbers in a cell that do not:

- Repeat on the same row
- Repeat on the same column
- Repeat within an 3x3 space.

A sudoku grid dimensions are multiples of 3, This means that both the length and height of the board have to be divisible by 3. In this case, we will be denoting empty spaces in the board using 'None'.

You are required to implement the function `is_solvable(board)` that will check to see if a sudoku puzzle is solvable. You are encouraged to use backtracking (a kind of brute forcing technique) for solving this problem.

Each grid space will either be represented as a number between 0-9 or with None (meaning the player could place a number in that spot).

```
is_solvable(
[[3, 2, 1],
[None, None, None],
[9, 8, 6]])

True


is_solvable(
[[1, 2, 2],
[3, 4, 6],
[7, 5, 9]])

False
```