CITY
UNIVERSITY OF LONDON
EST 1894

STEM Digital Academy

School of Science & Technology

www.city.ac.uk

Algorithms I

Programming and Algorithms

```
n = 3
for i in range(1,n+1):
    print("Hello World!")
```

```
Hello World!
Hello World!
Hello World!
```


Lecture by

Dr Daniil Osudin

What will we Cover?

- Understanding the concept of algorithms
- Introducing the algorithm efficiency and big-O notation

Algorithms in Computing

- Step by step set of well-defined instructions
    - Presented in the right order
    - Can be executed by a machine
    - Has a finite sequence of steps
    - Easily accessible to analysis

Describing an Algorithm

Algorithms can be described as

- Unstructured text
- Structured text
- Flow chart
- Pseudocode
- Code (Python code in this module)

Algorithm Example

Problem:

Find the number of vowels in a given word

```
Algorithm: Number of vowels

Input: source word w, v=[a, e, i, o, u]          # could include A, E, I, O, U ?
Output: vowel count n
Steps: set n to 0                                # initially count is zero
       for each letter in w                      # iterate over w
            if letter exists in v
               increment n by 1
       print n
```

Computational Complexity

- Description of the resources needed by an algorithm. Complexity in:
  - (running) time
  - (memory) space
- Best, worst and average cases
- Representation by growth

Big O notation

- **Big-O** is used to describe the efficiency of an algorithm
- Measure of the time the algorithm takes to run as the size of the input increases
- **Best case** scenario is the best performance that the algorithm can produce
- **Worst case** scenario is the worst performance that the algorithm can produce

Implications

- Big-O is an upper bound, actual values may be less
- Saying that an algorithm **a** is in **O(g(n))** means that it grows no faster than **g(n)**, where **g(n)** is some function of n (e.g. $\mathbf{n^2}$)
- **Note:** Often the word 'in' is omitted and it is said that an algorithm **a** is **O(g(n))**.

Common Types of Complexity I

Complexity type	Big-O notation	
Constant	$O(1)$	Good
Logarithmic	$O(\log n)$	
Linear	$O(n)$	
Quadratic	$O(n^2)$	Acceptable (sometimes)
Polynomial	$O(n^k)$	
Exponential	$O(a^n)$	Intractable

Common Types of Complexity II