

*Course title*        **Programming Fundamentals**  
*Instructor*        **Muhammad Ateeq**  
*Course Code*       **C**  
*Credit Hours*       **4 (3, 1)**  
*Semester*           **Spring 2020**  
*Pre-requisite*       **None**  
*Class Hours*       **Monday-Tuesday (10:00 to 11:30)**

### **Introduction**

Programming is an increasingly important skill, whether you aspire to a career in software development, or in other fields. This course is the first in the journey to learn computer programming in Python. However, the concepts can be generalized to any other programming language. This is because programming is fundamentally about figuring out how to solve a class of problems and writing the algorithm, a clear set of steps to solve any problem in its class. This course will undertake to build a search engine as a case study to solve a real-world problem and learn the problem solving approach along the way.

### **Course Objectives**

It is expected that at the end of this course the student will be able to

- Appreciate the need for computer programming
- Understand and use the programming environment (e.g., IDLE for Python)
- Understand and use basic arithmetic/logical operations
- Understand and use decisions and iterations structures
- Learn about modularity and put it to use
- Analyze a given problem and design, plan, and implement a solution

### **Course Outcomes**

At the end of the course, the student should be able to

- Understand basic problem solving steps and logic constructs.
- Apply basic programming steps
- Design and implement algorithms to solve real-world problems

### **Home Work/Programs**

Home work will be due at the beginning of the class. Under normal circumstances, late work will not be accepted. Students are expected to do their own programs, plagiarism will be observed strictly.

### **Attendance**

80% attendance is mandatory.

### **Evaluation Criteria**

S. No.	Assessment Items	%age
1	Assignments/Project	15
2	Quizzes / Class Participation	05
3	Midterm examination (after 8 weeks)	30
4	Final examination (after 16 weeks)	50
	<b>Total</b>	<b>100</b>

### **Course Outline**

Need for programming and languages, a brief review of Von-Neumann architecture, introduction to problem solving, introduction to programming, variables and their uses, numeric and strings data, input/output constructs, arithmetic, comparison and logical operators, conditional statements and execution flow for conditional statements, repetitive statements and execution flow for repetitive statements, lists and their organization, nested lists, operations on lists, function definition and calling, string and string operations, File I/O operations.

### **Recommended Material/Text Books:**

- Handouts for each lecture will be provided.

### **Reference Books:**

- [Practical Programming, An Introduction to Computer Science Using Python 3](#) (3<sup>rd</sup> Edition) by Paul Gries, Jennifer Campbell, and Jason Montojo. Pragmatic Bookshelf, 2017
- Head-First Python (2<sup>nd</sup> Edition) by Paul Barry. O'Reilly, 2016
- Python Crash Course (1<sup>st</sup> Edition) by Eric Matthes. No Starch Press, 2015.

### **Tentative Lecture Plan**

<b>Week #</b>	<b>Lecture Contents</b>
1	<b>Background Concepts and Preparation</b> <ul style="list-style-type: none"> <li>▪ The programming languages evolution: digital electronics, Binary/Assembly languages, high-level languages</li> <li>▪ Single-program machines, stored-program (Von Neumann) architecture, calculator vs computers</li> <li>▪ Installation of Python</li> <li>▪ Introduction to the course case study: a simple search engine</li> </ul>
2	<b>Programming Languages and Feel of Python</b> <ul style="list-style-type: none"> <li>▪ Programming languages vs natural languages</li> <li>▪ Grammar and expressions</li> <li>▪ Basic arithmetic statements</li> <li>▪ Variables</li> </ul>
3	<b>Numbers and Strings (Extracting the First Link)</b> <ul style="list-style-type: none"> <li>▪ Strings and binary operations: concatenation, multiplication</li> <li>▪ Indexing and slicing/sub-sequencing, find() method</li> <li>▪ Using the string operations to extract the first link from a webpage</li> </ul>
4	<b>Using Procedural Abstraction and Control (Extracting All Links)</b> <ul style="list-style-type: none"> <li>▪ Understanding the need of procedures</li> <li>▪ Defining/using the procedures, passing parameters and return statement</li> <li>▪ Making decisions, use of if and else</li> <li>▪ Understanding and writing logical conditions</li> </ul>
5	<b>Iterations and Control (Extracting All Links)</b> <ul style="list-style-type: none"> <li>▪ The need for iterative control</li> <li>▪ while loops, break statement</li> <li>▪ Multiple assignments</li> </ul>
6	<b>Extracting All Links on a Webpage (Learning to Crawl)</b> <ul style="list-style-type: none"> <li>▪ Writing procedure to extract all links on a webpage using numbers, strings, if-else and while loop</li> </ul>
7	<b>Structured Data and Nested Structures (Learning to Crawl)</b> <ul style="list-style-type: none"> <li>▪ Structured data using strings and lists</li> <li>▪ Nested lists</li> <li>▪ Mutation and aliasing</li> <li>▪ List operations: append, len</li> </ul>
8	<b>How Computers Store Data? Memory Hierarchy</b> <ul style="list-style-type: none"> <li>▪ DRAM</li> <li>▪ Size representation</li> <li>▪ Memory hierarchy</li> </ul>

	<b>Midterm Examination</b>
9	<b>List Operations (Learning to Crawl)</b> <ul style="list-style-type: none"> <li>▪ Loop over lists using while loop</li> <li>▪ Loop over lists using for loop</li> <li>▪ Using the index and pop methods</li> </ul>
10	<b>Completing the Crawler for Webpage</b> <ul style="list-style-type: none"> <li>▪ Using the learned concepts (numbers, strings, lists, loops, modularity) to implement a web crawler that can extract all the links in a webpage</li> </ul>
11	<b>Data Structure</b> <ul style="list-style-type: none"> <li>▪ Understanding the data structure abstraction</li> <li>▪ Emphasize the need for building an index of terms found on crawled webpages</li> </ul>
12	<b>Indexing a Webpage</b> <ul style="list-style-type: none"> <li>▪ Defining a data structure suitable for building an index of terms for search engine</li> <li>▪ Using string split method to make a list of keywords and adding those to an list</li> <li>▪ Add webpages to index one-by-one</li> </ul>
13	<b>Finishing the Web Crawler</b> <ul style="list-style-type: none"> <li>▪ Iterate over the links found in a webpage to continue the crawling and keep adding all the pages to index</li> <li>▪ Write a lookup procedure to search the desired terms in the index</li> <li>▪ Write a procedure to get a seed page by a given url</li> </ul>
14	<b>Algorithms and Runtime/Efficiency</b> <ul style="list-style-type: none"> <li>▪ Understanding the runtime and efficiency of algorithm/code</li> <li>▪ Measuring speed and predicting runtime</li> <li>▪ Index size vs runtime</li> <li>▪ Worst case analysis</li> </ul>
15	<b>Making Lookup Fast</b> <ul style="list-style-type: none"> <li>▪ Potential avenues: hashing, modulus operator, hash functions</li> <li>▪ Dictionaries</li> </ul>
16	<b>GUI and Revision</b> <ul style="list-style-type: none"> <li>▪ Nested procedure calls</li> <li>▪ Local and Global labels</li> <li>▪ Procedure Parameters</li> </ul>
	<b>Final Examination</b>