

**Langston University
School of Business**

Instructor:

Address:

Office Hours:

By Appointment

Course:

CS 3173 Programming Languages

Prerequisite:

CS 3153 or CS 3163

Assessment:

Six (6) 30 minute examinations (40%); individual assignments (20%); comprehensive final exam (40%).

Textbook:

Michael L. Scott, Programming Language Pragmatics

Course Description:

Covers the syntax, organization, and run-time behavior of a representative number of high-level languages used in problem-solving applications. Discusses control protocols, data types and structures, and primitive operations within these languages. Stresses the universality of primary concepts through hands-on assignments with a practical orientation.

S. O. B. Educational Philosophy:

Our classroom mission is to impart undergraduate education and to synthesize theory and practice that will equip students with mastery of the following skills:

- Professional communication skills [A]
- Critical thinking and self-reflection skills supported by qualitative and quantitative analysis essential to professional effectiveness in one's discipline [B]
- Team building and leadership skills [C]
- Effectiveness in cross-disciplinary teams through a larger understanding of other management functions' impacts on one's discipline [D]
- "Big picture" skills in discovering the utility of non-business discipline insights for enhanced business decision-making in a global economy, and professional effectiveness including ethical behavior [E]
- Technology that supports contemporary business analysis and decision-making [F]
- Application of one's disciplines to real problems "in the field" including "service" contributions, and practice in appropriate business etiquette. [G]

Coverage of these skills is indicated below in the Learning Outcomes.

Course Learning Objectives

Upon successful completion of the course, students should be able to learn different programming languages, understand how high level languages are translated into machine language, understand different program language constructs, and understand object oriented programming techniques

Course Learning Outcomes and Assessment

A: Knowledge and Understanding [B, F, G]

Relevant quantitative and computing techniques including linguistic methods, transformations to equivalent but more appropriate forms, and the translation of one language to another. Specifically:

1. Understand the history of programming language development.
2. Understand and use Backus-Naur and Extended Backus-Naur Form.
3. Understand and use regular expressions.
4. Understand the phases of compiler operation.
5. Understand the significance of types in a programming language.
6. Understand the concept of data abstraction.
7. Understand the concept of control abstraction.
8. Understand the implications of top down and bottom up compilation.
9. Understand the object oriented paradigm.
10. Understand the difference between compilation and interpretation.
11. Understand non-imperative programming models.

Teaching/learning methods and strategies

Formal lectures, practical computer laboratory sessions, supervisions supported by directed and assessed self-study. Feedback and guidance are an important part of study. Students shall write programs which demonstrate understanding of graphics and geometric principals.

Assessment

These are assessed through a combination of coursework and examinations.

B: Intellectual skills [B, D,E]

Cognitive skills of critical thinking, analysis, synthesis, including the ability to identify assumptions, evaluate statements in terms of evidence, to detect false logic or reasoning, to identify implicit values, and to define terms adequately and to generalize appropriately. This includes effective quantitative problem solving and decision making skills. Specifically:

1. Synthesis and evaluate the technical concepts of the syllabus.
2. Appraisal of theory and its relevance to different situations.
3. Analysis of tasks into understandable and manageable subtasks.
4. Synthesis of clearly and precisely stated solutions for problems.
5. The testing of proposed solutions for validity.
6. Correction of and refinement of proposed solutions.
7. Evaluation of proposed solutions for scale and scalability.

Teaching/learning methods and strategies

Substantive problems are illustrated in lectures and smaller groups. Project work and problems sets provide related opportunities for problem solving. Lectures supported by discussions provide basis of ensuring the growing knowledge base becomes comprehensible. The individual projects provide the central means of incorporating the skills.

Assessment

These are assessed through a combination of coursework and examinations.

C: Discipline Specific Skills [A,B,C,D,E,F,G]

Development of numeracy and quantitative skills including data analysis, interpretation and extrapolation; effective use of communication and information technology skills for business application; the ability to conduct research into business and management issues, either individually or as part of a team, including familiarity with a range of business data and research resources and appropriate methodologies; and facility with key concepts used in decision-making, including expectations. Specifically ability to:

1. Use Backus-Naur and Extended Backus-Naur Form.
2. Use regular expressions to express patterns.
3. Generate language phrases from an EBNF grammar.
4. Choose types correctly for use in programs.
5. Make use of data abstraction.
6. Make use of control abstraction.
7. Design special purpose languages.

Teaching/learning methods and strategies

Practical issues are illustrated in lectures and supporting lab classes, reinforced by problems sets and supervised project work.

Assessment

These are assessed through a combination of coursework and examinations.

D: Transferable skills [A, B, D, E, F]

Specifically, the student should be able to:

1. Learn new programming languages more easily with an understanding of the underlying concepts.
2. Use the object oriented programming paradigm.
3. Synthesize clearly and precisely stated solutions for problems.
4. Test proposed problem solutions for validity and usability.
5. Correct and refine proposed solutions.
6. Design for the effects of scale and scalability on solution choices.

Teaching/learning methods and strategies

The use of IT is an integral part of the practical side of the course. It is encouraged through programs which use a wide range of application disciplines as examples.

Skills will also be developed using examples in which the class participates as a group, developing a common solution.

Assessment

These are assessed through a combination of coursework and examinations.

Course Projects

Individual Activity: As part of the overall assessment for this course each student will be required to complete individual assignments which count for 20 percent of the overall grade. These assignments involve an empirical application of some of the techniques that have been covered in the course. Each member of the class is to work on his or her own assignment, with required personalization. All topics will be assigned in advance. Assignment due dates will be announced with the assignments. Data files will be supplied with assignments when needed. Project assignments will include relevant references when appropriate.

Tentative Class Calendar:

Session	Date		Lecture/Reading Assignment
August			
1	24	Chapter 1	Introduction to Programming Languages
2	31	Chapter 2	Programming Language Syntax
September			
3	7	Chapter 3	Names, Scopes, and Binding
4	14	Chapter 3	Names, Scopes, and Binding
5	21	Chapter 4	Semantic Analysis
6	28	Chapter 5	Assembly Language Computer Architecture
October			
7	5	Chapter 6	Control Flow
8	12	Chapter 6	Control Flow
9	19	Chapter 7	Data Types
10	26	Chapter 7	Data Types
November			
11	2	Chapter 8	Subroutines and Control Abstraction
12	9	Chapter 8	Subroutines and Control Abstraction
13	16	Chapter 9	Building a Running Program
14	23	Chapter 10	Data Abstraction and Object Orientation
15	30	Chapter 10	Data Abstraction and Object Orientation
December			
16	7	Chapter 11	Non-imperative Programming Models: Functional and Logical Languages
17	14	Chapters 1-12	Final Exam

AFFIRMATIVE ACTION STATEMENT

Langston University in compliance with Title VI and VII of the Civil Rights Acts of 1964, Executive Order 1121 as amended, Title IX of the Educational Amendments of 1972, American with Disabilities Act of 1990, and other Federal laws and regulations, does not discriminate on the basis of race, color, national origin, sex, age, religion, handicap, or status as a veteran in any of its policies, practices, or procedures. This includes but is not limited to admissions, employment, financial aid and educational services.

AMERICANS WITH DISABILITIES (ADA) STATEMENT

Langston University fully subscribes to all required standards of the ADA Act of 1990. Persons in need of assistance should contact the ADA Compliance Officer in the Office of Student Affairs, Room 119 Page Hall, 405-466-3445. This should be reported at some point before, during, or immediately after the first scheduled class period so accommodations can be provided for the student to be successful in that class.

- **Comportment:** Students are expected to abide by the School's dress code and code of ethics.
- This syllabus may be amended at anytime during the semester. All changes will be discussed in class.
- All **reading assignments** must be completed before class session.