INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

COMPUTER SCIENCE AND ENGINEERING

COURSE DESCRIPTION FORM

Course Title	**PRINCIPLES OF PROGRAMMING LANGUAGES**			
Course Code	**A50511**			
Regulation	**R15 - JNTUH**			
Course Structure	**Lectures**	**Tutorials**	**Practicals**	**Credits**
	4	-	-	4
Course Coordinator	Ms K. Radhika, Associate Professor, CSE			
Team of Instructors	Ms. B.Jaya Vijaya, Assistant Professor, CSE Mr. P. Sunil Kumar, Assistant Professor, CSE			

I. COURSE OVERVIEW:

The course addresses growing importance of Programming languages, their uses, and importance of using different programming tools. Course addresses various influences of language design and language implementation techniques like, compilers, interpreters. This course also explains about different expressions and statements used in different programming languages. Comparison of functional programming with logic programming, structure of imperative programming. Exceptions and exception handling procedures of different programming languages like, C#, C++, Java, Ada95. It also introduces the importance of scripting languages features and data types of Python language.

II. PREREQUISITE(S):

Level	Credits	Periods/ Week	Prerequisites
UG	4	4	Computer Programming, Formal Languages and Automata Theory

III. MARKS DISTRIBUTION:

Sessional Marks	University End Exam marks	Total marks
Midterm Test There shall be two midterm examinations. Each midterm examination consists of essay paper, objective paper and assignment. The essay paper is for 10 marks of 60 minutes duration and shall contain 4 questions. The student has to answer 2 questions, each carrying 5 marks. The objective paper is for 10 marks of 20 minutes duration. It consists of 10 multiple choice and 10 fill-in-the blank questions, the student has to answer all the questions and each carries half mark. First midterm examination shall be conducted for the first two and half units of syllabus and second midterm examination shall be conducted for the remaining portion. Five marks are earmarked for assignments. There shall be two assignments in every theory course. Assignments are usually issued at the time of commencement of the semester. These are of problem solving in nature with critical thinking.	75	100

`

Sessional Marks	University End Exam marks	Total marks
Marks shall be awarded considering the average of two midterm tests in each course.		

IV. EVALUATION SCHEME:

S. No	Component	Duration	Marks
1.	I Mid Examination	80 minutes	20
2.	I Assignment	-	5
3.	II Mid Examination	80 minutes	20
4.	II Assignment	-	5
5.	External Examination	3 hours	75

V. COURSE OBJECTIVES:

At the end of the course, the students will be able to:

I. Be familiar with the structure and design principles of programming languages.

II. Master the skills in analyzing and using the features of programming languages.

III. Be familiar with the preliminary concepts like context-free grammar, Backus-Naur form, Parse trees.

V. Be familiar with logic programming and functional programming languages features.

VI. Be familiar with variable declarations in programming languages, in particular to binding, scope, and substitution of variables.

VII. Be familiar with Python scripting language.

VI. COURSE OUTCOMES:

After completing this course the student must demonstrate the knowledge and ability to:

1. **Review** the concepts of programming languages.
2. **List** out various programming paradigms used in different languages.
3. **Recall** the design issues of various programming language implementation.
4. **Discuss** various programming environments.
5. **Elaborate** the features of attribute grammars and draw parse trees.
6. **List** out various data types in different programming languages.
7. **Tabulate** different parameter passing techniques of different programming languages.
8. **List** out the concepts of object oriented programming in C++, Ada95, and Smalltalk.
9. **Recall** the importance of semaphores, monitors, message passing.
10. **Apply** logic programming concepts by using PROLOG.
11. **Use** of functional programming languages like LISP, ML, Haskell.
12. **Apply** scripting languages in web design and real-time applications.

`

II. HOW PROGRAM OUTCOMES ARE ASSESSED:

Program Outcomes		Level	Proficiency assessed by
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.	H	Assignments, Tutorials
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.	H	Assignments
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.	S	Mini Projects
PO4	**Conduct investigations of complex problems** : Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.	S	Mini Projects
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.	S	Projects
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.	N	--
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.	N	--
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.	N	--
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.	N	--
PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.	N	--
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.	N	--
PO12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.	S	Projects, Discussions

N – None **S - Supportive** **H - Highly Related**

`

III. HOW PROGRAM SPECIFIC OUTCOMES ARE ASSESSED:

Program Specific Outcomes		Level	Proficiency assessed by
PSO1	**Professional Skills:** The ability to research, understand and implement computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient analysis and design of computer-based systems of varying complexity.	H	Lectures, Assignments
PSO2	**Problem-Solving Skills:** The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.	S	Tutorials
PSO3	**Successful Career and Entrepreneurship:** The ability to employ modern computer languages, environments, and platforms in creating innovative career paths, to be an entrepreneur, and a zest for higher studies.	S	Guest Lectures

N - None **S - Supportive** **H - Highly Related**

IX. SYLLABUS:

UNIT- I:

Preliminary Concepts: Reasons for studying, concepts of programming languages, Programming domains, Language Evaluation Criteria, influences on Language design, Language categories, Programming Paradigms– Imperative, Object Oriented, functional Programming , Logic Programming. Programming Language Implementation – Compilation and Virtual Machines, programming environments.

Syntax and Semantics: general Problem of describing Syntax and Semantics, formal methods of describing syntax - BNF, EBNF for common programming languages features, parse trees, ambiguous grammars, attribute grammars, denotational semantics and axiomatic semantics for common programming language features.

UNIT – II

Data types: Introduction, primitive, character, user defined, array, associative, record, union, pointer and reference types, design and implementation uses related to these types. Names, Variable, concept of binding, type checking, strong typing, type compatibility, named constants, variable initialization.

Expressions and Statements: Arithmetic relational and Boolean expressions, Short circuit evaluation mixed mode assignment, Assignment Statements, Control Structures – Statement Level, Compound Statements, Selection, Iteration, Unconditional Statements, and guarded commands.

UNIT – III

Subprograms and Blocks: Fundamentals of sub-programs, Scope and lifetime of variable, static and dynamic scope, Design issues of subprograms and operations, local referencing environments, parameter passing methods, overloaded sub-programs, generic sub-programs, parameters that are sub-program names, design issues for functions user defined overloaded operators, co routines

UNIT – IV

Abstract Data types: Abstractions and encapsulation, introductions to data abstraction, design issues, language examples, C++ parameterized ADT, object oriented programming in small talk, C++, Java, C#, Ada 95.

Concurrency: Subprogram level concurrency, semaphores, monitors, massage passing, Java threads, C# threads.

Exception handling: Exceptions, exception Propagation, Exception handler in Ada, C++ and Java.

Logic Programming Language: Introduction and overview of logic programming, basic elements of prolog, application of logic programming

UNIT – V

Functional Programming Languages: Introduction, fundamentals of FPL, LISP, ML, Haskell, application of Functional Programming Languages and comparison of functional and imperative Languages.

`

Scripting Language: Pragmatics, key concepts, case study: Python- values and types, variables, storage and control, bindings and scope, procedural abstraction, data abstraction, separate compilation, module library.

Text books:

1. Robert .W. Sebesta, "Concepts of Programming Languages", 9/e, Pearson Education.

References:

1. A. B. Tucker, R. E. Noonan, "Programming languages", 2e, TMH.
2. K. C. Louden, "Programming Languages", 2e, 2003.
3. Patric Henry Winston and Paul Horn," LISP", Pearson Education.
4. W. F. Clocksin, C. S. Melish, "Programming in Prolog", 5e, Springer.

X. COURSE PLAN:

At the end of the course, the students are able to achieve the following course learning outcomes:

Lecture No.	Topics to be covered	Course Learning Outcomes	Reference
1	Reasons for studying concepts of programming languages.	**Identify the** importance of programming languages.	T1: 1.2
2	Programming domains.	**Understand** different programming domains.	T1: 1.5
3 – 4	Language evaluation criteria, influences on language design	**Evaluate language** criteria that influence on language design.	T1: 1.7
5	Language categories.	**Categorize** the languages	T1: 1.22
6 – 7	Programming Paradigms – imperative, object oriented, functional programming, and logic programming.	**Compare and Contrast** different programming paradigms	T1: 1.25
8 – 9	Programming language implementation – compilation and virtual machines	**Reproduce** programming language Implementation	T1: 2.2
10	Programming environments.	**Distinguish** programming environments	T1: 1.32
11	General Problem of describing syntax and semantics	**Understand** Syntax and Semantics	T1: 3.3
12 – 15	Formal methods of describing syntax - BNF, EBNF for common programming languages features	**Contrast** BNF, EBNF.	T1: 3.5
16	Parse trees	**Construct** parse trees for given grammar	T1: 3.6
17 – 18	Ambiguous grammar, attribute grammar.	**Distinguish** ambiguous grammars and define attribute grammar	T1: 3.7
19 – 21	Denotational semantics and axiomatic semantics for common programming language features.	**Understand** semantics of common programming language features	T1: 3.27
22 – 26	Introduction, primitive, character. user defined, array, associative, record, union, pointer and reference types, design and implementation uses related to these types.	**Use** different data types	T1: 6.4
27 – 30	Names, variable, concept of binding, type checking, strong typing, type compatibility, named constants, variable initialization.	**Review** the concept of binding conversion and compatibility of data types	T1: 5.2,6.5
31 - 33	Arithmetic relational and Boolean expressions.	**Illustrate** different type of expressions	T1: 7.3

`

34 – 38	Short circuit evaluation mixed mode assignment, assignment statements. Control Structures – Statement Level, compound statements. selection, iteration, unconditional statements, guarded commands.	**Understand** different types of Statements.	T1: 7.1,8.3
39– 40	Fundamentals of sub-programs, scope and lifetime of variable, static and dynamic scope, design issues of subprograms and operations	**Able** to write subprograms	T1: 9.2
41	Local referencing environments.	**Understand** local referencing environments	T1: 9.1
42 – 43	Parameter passing methods. overloaded sub-programs, generic sub-programs, parameters that are sub-program names	**Distinguish** different types of parameter passing methods	T1: 9.4
44 – 46	Design issues for functions , user defined overloaded operators, co routines.	**Illustrate** design issues for functions and co routines	T1: 9.5
47	Abstractions and encapsulation, introductions to data abstraction.	**Understand** data abstraction.	T1:10.3
48 – 49	Design issues, language examples. C++ parameterized ADT.	**Illustrate** design issues with examples	T1:10.7
50 – 51	Object oriented programming in small talk, C++, Java, C#, Ada 95.	**Compare and contrast** oops concepts of different languages	T1:11.1
52 – 53	Subprogram level concurrency, semaphores, monitors, Massage passing, Java and C# threads.	**Understand** concurrency concepts	T1:12.6
54 – 56	Exceptions, exception Propagation, Exception handler in Ada, C++ and Java.	**Illustrate** exceptional handling concepts of different languages	T1: 13.2
57 – 58	Introduction and overview of logic programming, basic elements of prolog, application of logic programming	**Understand** the basic concepts of logic programming and its applications	T1: 14.3
59 – 60	Introduction, fundamentals of FPL, LISP, ML, Haskell. Application of functional programming languages , comparison of functional and imperative languages	**Understand** the basic concepts and applications of different functional programming languages	T1:14.7
61 – 62	Pragmatics, key concepts, case study: Python- values and types, variables, storage and control, bindings and scope, procedural abstraction, data abstraction, separate compilation, module library	**Understand** about scripting languages	T1:16.5

`

XI. MAPPING COURSE OBJECTIVES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:

Course	Program Outcomes												Program Specific Outcomes		
Objectives	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
I	H	H			H							S	H	S	S
II	S	H			S							S	H	S	S
III	H	S	S	S	S							S	S		
IV	H	S			H							H	H	H	S
V	S		S										S		
VI			S		S							S	H	H	S
VII			S	S	S							S			H

S – Supportive **H - Highly Related**

XII. MAPPING COURSE OUTCOMES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:

Course	Program Outcomes												Program Specific Outcomes		
Outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
1	S	S											H	H	H
2	S												H	S	
3					S							S	H	H	
4			S		S								S		
5	H	S											S	S	
6	H	H											H	S	
7						H						H	H	H	S
8				S		S							S	S	
9	S					H						H	H	H	S
10		S	S	S		S							S		
11		S	S			S							S		
12	S		S	S		H						S	S		

S – Supportive **H - Highly Related**

Prepared by **:** Ms K Radhika, AssociateProfessor, CSE

HOD, CSE