|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Principles of Programming Languages (Elective -VI) | **L** | **T** | **P** | **C** |
| **Version 1.0** | **Contact Hour -45** | **3** | **0** | **0** | **3** |
| **Pre-requisites/Exposure** | **B.Tech level Computer Programming Knowledge** | | | | |
| **Co-requisites** | **-** | | | | |

**Course Objectives:**

* To provide the knowledge of, and ability to use, language features used in current programming languages.
* To develop an ability to program in different language paradigms and evaluate their relative benefits.
* To introduce of the key concepts in the implementation of common features of programming languages.

**Course Outcomes:**

On completion of this course, the students will be able to

CO1. **Define** the major programming paradigms, and the principles and techniques involved in design and

implementation of modern programming languages.

CO2. **Classify** the notations to describe syntax and semantics of programming languages.

CO3. **Inspect** the behavior of simple programs in imperative languages using concepts such as binding, scope,

control structures, subprograms and parameter passing mechanisms

CO4. **Compare** the concepts of Imperative programming, Declarative programming and object oriented

programming for large scale software development.

CO5. **Develop** the application of prolog and LISP.

CO6. **Design** the concept of Formal Systematic and Functional Programming.

**Catalog Description:**

Presents examples of important programming languages and paradigms such as LISP, ALGOL, ADA, ML, Prolog, and C++. Students write sample programs in some of the languages studied. The languages are used to illustrate programming language constructs such as binding, binding times, data types and implementation, operations (assignment data-type creation, pattern matching), data control, storage management, parameter passing, and operating environment. The suitability of these various languages for particular programming tasks is also covered. All the lectures will be devoted on discussions of basic theories and advanced topics, focusing on practical implementation of knowledge. Classes will be conducted by lecture as well as power point presentation, as per requirement. The tutorials will familiarize the students with practical problem-solving techniques led by the course coordinator. Students will strongly grab the basic concepts of the subject via exercise and discussions with the coordinator.

**Course Content:**

**Unit-I 5 Lecture Hours**

**Module 1:**

**Introduction:** Programming language definition, brief history of programming

Languages, overview of programming paradigms.

**Language design principles:** Design criteria, efficiency, regularity

**Unit II: 8 Lecture Hours**

**Syntax:** Lexical structure, Context free grammar, BNF, syntax tree, parse tree, Expression syntax.

**Semantics:** Declaration, allocation, evaluation, symbol table, runtime environment, data types, type checking, weak typing, strong typing, parameter passing methods such as pass by value, pass by name, pass by result, pass by value-result, pass by reference, exceptions and exceptions handling.

**Unit III: 7 Lecture Hours**

**Garbage collection:** Advantages, explicit garbage collection, automatic garbage

Collection compacting.

**Imperative programming:** Impact of Von-Neumann architectures on programming

language, assignments, names, locations, L-value, R-value, memory allocation, scope rules, control flow, control abstraction, functions, exception handling, primitive and constructed data types, data abstraction.

**Unit IV: 10 Lecture Hours**

**Object oriented programming:** Objects, classes, methods, dynamic binding,

inheritance, polymorphism, design and implementation issues in object oriented

Languages, case study.

**Declarative programming:** Distinctive features of declarative programming, first order logic, Horn clauses, resolution unification, sequencing of control, negation,

Implementations issues, the language Prolog, constraint logic programming.

**Unit V: 10 Lecture Hours**

**Functional programming:** Distinctive features of functional programming languages, functional programming in imperative language, recursion, tail recursion, higher order functions, lazy evaluation, types in functional programming, mathematics of functional programming: lambda calculus. introduction to functional programming using Scheme Haskell ML.

**Unit VI:5 Lecture Hours**

**Brief introduction to multi-paradigm languages** (Python/Leda/Ada/C#).

**Formal semantics:** Operational semantics, denotational semantics, axiomatic semantics, proof of program correctness.

**Text Books:**

1. “Programming Languages: Principles and practice”, Kenneth C. Louden, 2003.
2. “Programming Languages and Paradigms”, D. A. Watt, Prentice-Hall, 1990.
3. “Advanced Topics in Types and Programming Languages”, Benjamin C. Pierce, ed., MIT Press, 2005.
4. “Foundations of Logic Programming”, J. Lloyd, Springer Verlag, 1984.

**Reference Books:**

1. “The Semantics of Programming Languages”, M. Hennessey, John Wiley, 1990.
2. “Elements of Functional Programming”, C. Reade, Addison Wesley, 1989.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Public Blockchain-Ethereum | **L** | **T** | **P** | **C** |
| **Version 1.0** | **Contact Hours-45** | **3** | **0** | **0** | **3** |
| **Pre-requisites/Exposure** | **Crypto currency and computer security basics** | | | | |
| **Co-requisites** | **--** | | | | |

**Course Objectives:**

* To gain knowledge about the building blocks of blockchain ethereum.
* To enable students to install and configure Mist browser,
* To give the students a perspective to learn the basics of EVM and Solidity programming.
* To enable students acquire knowledge about smart contract and tokens..

**Course Outcomes:**

On completion of this course, the students will be able to

CO1**. Understand** the basics of blockchain ethereum.

CO2**. Explain** the procedure of installation of Mist browser and its configuration.

CO3**. Explain** the role of Ethereum protocol in Banking.

CO4**. Understand** the basics of Solidity programming primer.

CO5**. Understand** the utility of smart contract and token.

CO6**. Evaluate** the ancestry of blocks and transactions.

**Catalog Description:**

This course is the definitive introduction to permissioned blockchain for the students. Beyond the technology, this course will introduce you to some of the philosophy behind decentralization and why there is so much excitement around it.

During the tenure of the course, the students will be introduced to blockchain and the technology behind it. In the later modules, the topics beyond bitcoin will be taken up and delve deeper into a next-generation blockchain called Ethereum to introduce students to what modern blockchains can do.

**Course Content:**

**Unit I: 6 lecture hours**

**Bridging the Blockchain Knowledge Gap**: What Ethereum Does, Three Parts of a Blockchain, Ether as a Currency and Commodity , The Power Is in the Protocol , You Can Build Trustless Systems, What Smart Contracts: Objects and Methods for Value , Just Add Commerce ,Content Creation; Where’s the Data? : What Is Mining? , Ether and Electricity Prices; EVM:The Mist Browser , Browser vs. Wallet or Keychain; What Ethereum Is Good For : State of Smart Contract Development Today, A Note to New Programmers : Ethereum Is Free and Open Source , The EVM Is Here to Stay ; What You Can Build Today : Private and Public Chains , The Promise of Decentralized Databases, What’s Next: New Ways of Working

**Unit II: 8 lecture hours**

**The Mist Browser:** introduction, The Bank Teller Metaphor , In Cryptocurrency, You Hold Your Own Assets , Visualizing Ethereum Transactions, Breaking with Banking History , How Encryption Leads to Trust, System Requirements , More about Eth.guide and This Book, Tools for Developers, CLI Nodes, Recommended: Using Parity with Geth, Finally, into the Mist! , Downloading and Installing Mist, Configuring Mist, Finding Your New Address, Sending and Receiving Ether, Understanding Ethereum Account Types, Backing Up and Restoring Your Keys, Using Paper Wallets, Using Mobile Wallets, Working with Messages and Transactions, So, What Is a Blockchain? , Paying for Transactions, Understanding Denominations, Getting Ether, Anonymity in Cryptocurrency, Blockchain Explorers .

**Unit III: 8 lecture hours**

**The EVM:**  The Role of the Ethereum Protocol in Banking , What the EVM Does, EVM Applications Are Called Smart Contracts, The Name “Smart Contracts” , Understanding State Machines , Digital vs. Analog, “Statements” , Data’s Role in State , How the Guts of the EVM Work, The EVM Constantly Checks for Transactions, Creating a Common Machine Narrative of What Happened, Cryptographic Hashing , Hash Functions (or Hash Algorithms), Blocks: The History of State Changes, Block Time, Drawbacks of Short Blocks, “Solo Node” Blockchain , Distributed Security, Mining’s Place in the State Transition Function , Renting Time on the EVM , Gas : Why Is Gas So Important?, Why Isn’t Gas Priced in Ether?, Fees as Regulation , Working with Gas , Gas Specifics, How Gas Relates to Scaling the System, Accounts, Transactions, and Messages, Externally Owned Accounts , Contract Accounts , Transactions and Messages, Characteristics of Transactions , Characteristics of Messages , Estimating Gas Fees for Operations , Opcodes in the EVM .

**Unit IV: 8 lecture hours**

Solidity Programming Primer: Global Banking Made (Almost) Real, Extra-Large Infrastructure, Worldwide Currency? , Complementary Currency, The Promise of Solidity, Browser Compiler, Learning to Program the EVM , Easy Deployment, The Case for Writing Business Logic in Solidity, Code, Deploy, Relax, Design Rationale , Writing Loops in Solidity, Expressiveness and Security, The Importance of Formal Proofs, Historical Impact of a Shared Global Resource , How Attackers Bring Down Communities, Hypothetical Attack Written in Solidity , Automated Proofs to the Rescue?, Determinism in Practice , Lost in Translation, Testing, Testing, Testing, Command Line Optional! , Formatting Solidity Files, Tips for Reading Code, Statements and Expressions in Solidity, What Is an Expression? What Is a Statement? , Functions, Public and Private , Value Types, Booleans, Signed and Unsigned Integers, Addresses, Members of Addresses, Address-Related Keywords , Less-Common Value Types , Complex (Reference) Types , Global Special Variables, Units, and Functions, Block and Transaction Properties , Operators Cheat Sheet, Global Functions, Exceptions and Inheritance .

**Unit V: 7 lecture hours**

**Smart Contracts and Tokens**: EVM as Back End, Smart Contracts to Dapps, Assets Backed by Anything , Bartering with Fiat Currency, Ether as Glass Beads, Cryptocurrency Is a Measure of Time, Asset Ownership and Civilization , Coins are Collectibles , The Function of Collectibles in Human Systems , Early Counterfeiting, Jewelry and Art as Money , The Step Toward Banknotes , Platforms for High-Value Digital Collectibles , Tokens Are a Category of Smart Contract , Tokens as Social Contracts, Tokens Are a Great First App, Creating a Token on the Testnet , Getting Test Ether from the Faucet, Registering Your Tokens , Deploying Your First Contract, Same House, Different Address , Playing with Contracts .

**Unit VI: 8 lecture hours**

**Mining Ether**: Ether’s Source, Mining , Self-Regulation, and the Race for Profit , How Proof of Work Helps Regulate Block Time , What’s Going on with the DAG and Nonce?, Making Fast Blocks Work , How Ethereum Uses Stale Blocks , Uncle Rules and Rewards, The Difficulty Bomb, Miner’s Winning Payout Structure , Limits on Ancestry, The Block Processing Play by Play , Evaluating the Ancestry of Blocks and Transactions, How Ethereum and Bitcoin Use Trees , Merkle-Patricia Trees, Contents of an Ethereum Block Header, Forking , Installing Geth on macOS , Installing Geth on Windows , Getting Comfortable with the Command Line, Installing Geth on Ubuntu 14.04 , Executing Commands in the EVM via the Geth Console, Launching Geth with Flags, Fire Up Your Miner! , Mining on the Testnet, GPU Mining Rigs, Mining on a Pool with Multiple GPUs.; use Cases: Chains Everywhere, The Internet of Ethereum Things, Retail and E-Commerce, Community and Government Financing, Human and Organizational Behavior, Financial and Insurance Applications, Inventory and Accounting Systems, Software Development, Gaming, Gambling, and Investing.

**Text Books:**

1.Mayukh Mukhopadhyay - Ethereum smart contract development\_ build blockchain-based decentralized applications using Solidity-Packt Publishing (2018)

2.Chris Dannen (auth.) - Introducing Ethereum and Solidity\_ Foundations of Cryptocurrency and Blockchain Programming for Beginners-Apress (2017)

3.Mastering Bitcoin: Programming The Open Blockchain, Andreas M. Antonopoulos, O'Reilly, ISBN: 9789352135745.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Public Blockchain-Ethereum Lab | **L** | **T** | **P** | **C** |
| **Version 1.0** | **Contact Hours -45** | **0** | **0** | **3** | **2** |
| **Pre-requisites/Exposure** | **Crypto currency and computer security basics** | | | | |
| **Co-requisites** | **--** | | | | |

**Course Objectives:**

* To gain knowledge about the building blocks of blockchain ethereum.
* To enable students to install and configure Mist browser,
* To give the students a perspective to learn the basics of EVM and Solidity programming.
* To enable students acquire knowledge about smart contract and tokens..

**Course Outcomes:**

On completion of this course, the students will be able to

CO1**. Understand** the basics of Solidity programming.

CO2**. Apply** the decision making constructs and loops to perform conditional execution.

CO3**. Apply** the various types of inheritance in Solidity programming.

CO4**. Distinguish** between function and contract polymorphism.

CO5**. Explain** and apply error handling use cases.

**Catalog Description:**

This course is the definitive introduction to permissioned blockchain for the students. Beyond the technology, this course will introduce you to some of the philosophy behind decentralization and why there is so much excitement around it.

During the tenure of the course, the students will be introduced to blockchain and the technology behind it. In the later modules, the topics beyond bitcoin will be taken up and delve deeper into a next-generation blockchain called Ethereum to introduce students to what modern blockchains can do.

**Course Content:**

**Experiments:**

|  |  |
| --- | --- |
| 1 | Write a Solidity program to create a smart contract. |
| 2 | Solidity program to demonstrate the use of decision making statements. |
| 3 | Solidity program to demonstrate the use of loops. |
| 4 | Solidity program to demonstrate the creation of an event. |
| 5 | Solidity program to demonstrate the use of pure and view functions. |
| 6 | Solidity program to demonstrate the use of different types of inheritance. |
| 7 | Solidity program to demonstrate the use of abstract contract. |
| 8 | Solidity program to demonstrate the use of Function polymorphism. |
| 9 | Solidity program to demonstrate the use of contract polymorphism. |
| 10 | Solidity program to demonstrate the error handling. |

**Text Books:**

1.Mayukh Mukhopadhyay - Ethereum smart contract development\_ build blockchain-based decentralized applications using Solidity-Packt Publishing (2018)

2.Chris Dannen (auth.) - Introducing Ethereum and Solidity\_ Foundations of Cryptocurrency and Blockchain Programming for Beginners-Apress (2017)

3.Mastering Bitcoin: Programming The Open Blockchain, Andreas M. Antonopoulos, O'Reilly, ISBN: 9789352135745.