Computational Thinking
Lecture 0: Course Introduction

University of Engineering and Technology
VIETNAM NATIONAL UNIVERSITY HANOI

Outline

- Course Information
- Learning Outcomes
- Course Content
- Course Assessment
- Learning Strategy Suggestions

Course Information

- Course Name: Computational Thinking
- Course Code: UET.COM1050
- Number of Credits: 5 (54/42/0/154)
- Course Organization:
  - Lecture: 15 weeks ~ 13 lectures + 02 progress exams
    (3-hour offline lecture + 1-hour online lecture/tutor)
  - Progress exams: Week 6, Week 13
    (Offline, programming, no internet)
  - Practice: 14 weeks (from week 2 on wards)
    (3-hour session, offline, personal computer)

Course Objectives

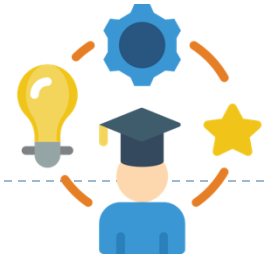
Knowledge:

- *Conceptual Understanding*: Understand computational thinking to analyze and model problems.
- *Programming Knowledge*: Represent solutions in Python with core constructs and execution.

Skills:

- *Problem-Solving Skills*: Solve problems programmatically through clear input, process, output.
- *Practical Programming Skills*: Write, debug, and optimize readable, maintainable Python code.
- *Independent Learning & Adaptability*: Use tools and English resources for self-learning and adaptability

Learning Outcomes



Knowledge:

- CLO1. Understand (2) the thinking process in the way computers do to solve problems logically and systematically.
- CLO2. Grasp (2) knowledge about how to represent thinking, execution processes and error handling of computer programs to write Python programs to solve simple problems.

Skills:

- CLO3. Apply (3) computational thinking methods to solve problems using the Python programming language and supporting software tools.
- CLO4. Practice (3) skills in presenting source code, debugging and improving programs, combining the use of English to learn and use libraries and programming support tools.

Level of Autonomy and Responsibility:

- CLO5. Demonstrate the ability to work independently through building complete programs (3), while adhering to the principles of honesty and responsibility during the practice process.

Course Materials

- **Primary Book:**
  - VNU-UET Computational Thinking Slides (Annually update)
  - Allen, Downey. Think Python: How to Think Like a Computer Scientist. Green Tea Press, 2015.
- **Reference Books:**
  - De Jesús, Sofía, and Dayrene Martinez. Applied Computational Thinking with Python: Design algorithmic solutions for complex and challenging real-world problems. Packt Publishing Ltd, 2020.
  - Peter J. Denning and Matti Tedre. Computational Thinking. The MIT Press, 2019.
  - Sedgewick, Robert, and Kevin Wayne. Computer science: An interdisciplinary approach. Addison-Wesley Professional,
- **Other sources** (Similar courses, Internet, etc.)

Course Content

1. Introduction
2. AI-assisted Programming
3. Expressions, Operators, Simple I/O
4. Selection Control Structure (If/Else)
5. Loop Control Structure (for, while, range)
6. Functions - Specification - Testing
7. Data Structures: List, Tuple, Dictionary
8. Searching and Sorting
9. Classes and Methods
10. Advanced Functions: Recursion
11. Exception Handling & Program Debugging
12. File I/O - File Reading/Writing
13. Popular Python Libraries

Course Assessment

- **Rubric-based Assessment**
- **Weekly Assessment: 40%**
  - Class Attendance (via Canvas quizzes): 10%
  - Programming Skill (10 offline tests): 30%
- **Progress Assessment: 30%**
  - Two progress exams (Week 6, Week 13): 15% each
  - Offline, programming, no Internet/Mobile
- **Final Exam: 30%**
  - Paper-based MCQs, Offline, no Internet/Mobile
  - Time: as school exam schedule

Learning Strategy Suggestions

- **DO NOT**
  - Depend entirely on AI – Use it as support, not a crutch.
  - Limit your sources – Explore books, online docs, forums, and projects.
  - Fear mistakes – Debugging is part of learning, not failure.
  - Rush to code without planning – Think in steps before writing code.
- **DO**
  - Use AI as a personal tutor – Ask for hints, not full solutions.
  - Get your hands dirty – Code actively, not just read or watch.
  - Learn from examples – Study sample problems, then modify them.
  - Test and debug often – Expect errors; practice fixing them.
- **Practice, Practice and Practice**

Keep calm and learn coding!