Introduction to Programming
Syllabus

Aidin Biibosunov

Course Information

- Title: Introduction to Programming
- Class: IT-23
- Semester: Fall semester
- Credits: 3
- Hours:
  - Lectures: 1 lesson / week, 45 min / lesson
  - Seminars: 2 lessons / week, 45 min / lesson

Contacts

- Name of the instuctor: Aidin Biibosunov
- Teacher at the department of IT & Product Design
- Email: a.biibosunov@compasscollege.art

Prerequisites

- A basic understanding of how to use a computer, like navigating the operating system, creating and managing files and folders, and installing software, and using the Internet.
- A basic understanding of mathematics, concepts like algebra and arithmetics.

- Almost all programming languages use English-based syntax and documentation. In this spirit, all written material in this course will be also in English. So, some level of English will be helpful.
- This course is designed for beginners, so prior programming knowledge and experience is not required. Nor any knowledge of physics.

Course Description

In today's digital age, programming forms the backbone of technology, powering everything from websites and mobile apps to data analysis and artificial intelligence. This course equips students with the essential skills to navigate this dynamic landscape. Whether a student is aiming to become a software developer, interested in data analysis, or simply wants to understand how programming works, this course will provide a student with a strong foundation in programming concepts.
Through hands-on learning and practical problem sets, students will gain proficiency in programming fundamentals and problem solving. In this course, students will learn Python programming language using physics as an example. That is, students will apply the concepts they have learned to solve various physics problems. The emphasis of this class is on students programming by themselves with the guidance of the course instructor.

Course Objectives

Through a combination of theoretical understanding, practical exercises, and project-based learning, this course aims to equip students with the foundational programming skills necessary to excel in various technology-related fields and to prepare them for advanced programming courses.

- Lay the Foundation for Advanced Programming Languages: Build a solid understanding of programming concepts that serves as a stepping stone to mastering more advanced languages.
- Facilitate Smooth Transition to Specialized Roles: Set the stage for specialization in areas such as web development, data analysis, machine learning, or mobile app development by mastering programming fundamentals.
- Empower Problem Solving: Develop the ability to dissect complex problems and come-up with solutions through coding.

- Promote Lifelong Learning Mindset: Foster a mindset of continuous growth and exploration, reflecting the rapid pace of innovation in the tech industry.

Course Goals

The goal of this course is to provide students with a strong foundation in programming principles and practices. By the end of the course, students will be able to:

- Write and execute programs using Python.
- Write clean code
- Utilize control structures such as loops and conditionals to create efficient code.
- Debug and troubleshoot common errors in their code.
- Understand and implement object-oriented programming concepts.
- Break down complex problems into manageable steps.

Course Content

- Introduction to Python: Basic concepts like variables, data types, and operations. Output using print() function. Basic input using input() function. Basic arithmetic operations.
- Boolean operators and Control Flow: Conditional statements (if, elif, else). Loops (for and while loops). Loop control statements (break, continue).
- Data Structures: Lists: Creating, indexing, slicing, appending, modifying. Strings: Indexing, slicing, string methods.
- Data Structures(Continued): Tuples: Creating, indexing, unpacking. Dictionaries: Creating, accessing, adding, modifying.
- Functions: Defining and calling functions. Parameters and return values. Scope and lifetime of variables. Built-in functions vs. user-defined functions.

- Data Visualization: Using libraries like Matplotlib or Seaborn for data visualization.
- Midterm Exam
- File Handling: Reading and writing files. File modes (read, write, append).
- Error Handling: Handling exceptions using try and except. Raising exceptions.
- Intro to Object-Oriented Programming (OOP): Classes and objects. Methods and attributes. Constructors and destructors. Inheritance and polymorphism.
- Data Manipulation: Using libraries like NumPy for numerical operations. Using libraries like Pandas for data analysis and manipulation.
- Final Exam

Calendar

- Link to the excel sheet: Calendar
- *Note*: the exact order and topic is subject to change depending on the progress of the students.

Week	Python Topic	Physics Topic	Problem Set
Week 1	Intro (with slides); Tools(Jupyter notebook, conda, etc); Variables, Variable Types (with concepts from physics), arithmetics & operators,Input and output operations, import, comments	Kinematics with constant acceleration	PS 1

Python Programming Topics	Corresponding Physics Topics
Introduction to Python	Introduction to Physics Concepts
Variables and Data Types	Units and Measurements
Control Flow	Kinematics
Functions	Forces and Newton's Laws
Lists and Data Structures	Energy and Work
Loops and Iteration	Circular Motion
File Handling	Thermodynamics
Object-Oriented Programming	Waves and Sound
Libraries and Modules	Electricity and Magnetism
Data Analysis with Pandas	Optics and Light
Basic Data Visualization	Simple Harmonic Motion
Numerical Computing with NumPy	Fluid Mechanics (basic concepts)
Web Scraping (Optional)	Quantum Mechanics (basic concepts)
Final Project	Integrating Multiple Physics Topics

Course Materials

- All materials and links will be provided via Edupage platform. I.e. lectures notebooks, PSets, etc.
- All written material will be in English & oral explanations in Russian.
- There is no required textbook for this course.
- Handy links will be provided during class in the appropriate lessons.

Grading Policy

Grading:

- Attendance: 15%
- Problem sets: 35%
- Midterm exam: 25%
- Final exam: 25%

  written material in english & oral explanations in russian

Workflow: lecture workflow: prepare a lesson notebook with exercises in them. Then have a separate copy of this notebook with answers. Whilst, give them without answers so that they solve them during the lecture(interactive). In the end of the lecture share with them the notebook with answers. Or: with the lecture notebook go throught it together interactively(asking questions, solving exercises). Then explain the physics topic and elaborate on the PSet. To make them engaged in the lesson.

points	grade	meaning
80-100	5	very good
60-79	4	good
40-59	3	satisfactory
0-39	2	failed

Attendance

- Attending the lesson gives an oppurtinity to ask questions about the lecture material and the problem set. Also, have an in-class discussion with the teacher and classmates.

Problem Sets

- You can work together to solve PSets. Even more, it is encouraged - programming is often done in teams, as is most interesting work!
- Said that, you are personally responsible for understanding everything you turn in. While you may ask one another about PSets, you may not copy directly from a classmate.
- Also you are not allowed to directly copy-and-paste their code.
- No late assignments will be accepted. Once the deadline passes, assignments will no longer be able to be submitted for credit.

Midterm

The midterm will have two parts and will cover all the passed material:

1. A conceptual, in-person part taken in class. It will be closed-notes.
2. A practical, take-home portion, after the first part. The take-home portion will be open-notes and open-Google. All parts will be completed individually. Students will not be permitted to discuss the questions on the exam with anyone.

Final exam

The final exam will have the same structure as the midterm. It will cover all the course material.

Academic integrity