SABANCI UNIVERSITY
IF 100 - Computational Approaches to Problem Solving
Summer 2021-2022

Instructors

İnanç Arın, UC 1083/1089, inanc.arin@sabanciuniv.edu

Course Description

The course is an introduction to the key concepts in computational thinking such as algorithmic thinking, abstraction and decomposition. The students will also gain basic programming skills in order to apply computational thinking concepts in practice.

Through the lectures, take-home exams, and interactive recitations specific to different disciplines, the students will learn how to design algorithms, how to divide a problem into subproblems, and how to build a solution by means of compositions. Evaluation of the solutions in terms of correctness and efficiency will also be covered.

In order to enable students to apply computational thinking skills in practice, basic programming concepts, such as variables, statements, conditionals, iteration, and functions will be introduced by using a simple programming language such as Python.

Course Website

https://sites.google.com/sabanciuniv.edu/if100

Exam Dates

- *Midterm Examination*: TBA
- *Final Examination*: TBA (by SR)

Tentative Grading

Recitation Grade	15%
Take-home Exam (THE)	20%
Midterm Examination	30%
Final Examination	35%

Please note that weighted average is not the only criterion in letter grading!

- ❖ We will have physical midterm and final exams. Details regarding the examinations will be announced later. University regulations will be followed for special cases.
- ❖ At the end of the semester, course grades will be calculated using a curve grading system.
  - ➢ We have no intention of releasing the letter grade boundaries.
- ❖ There will be around 4-5 take-home exams (THEs) during the semester. All of the assigned take-home exams will be graded and taken into consideration in the overall grade. Each take-home exam will have equal weight in overall grading.
- ❖ Contribution of the take-home exams to the overall grade will be calculated according to the formula given below:

$$take\_home\_grade \begin{cases} take\_home\_avr & if \quad ratio \leq 2 \\ take\_home\_avr \times (3 - ratio) & if \quad 2 < ratio < 3 \\ 0 & if \quad ratio \geq 3 \end{cases}$$

$ratio = submitted\_take\_home\_avr \,/\, weighted\_exam\_avr$
$weighted\_exam\_avr = (0.30 \times midterm\_grade + 0.35 \times final\_grade) \,/\, 0.65$
$course\_numeric\_grade = recitation\_grade \times 0.15 + take\_home\_grade \times 0.20 +$
$midterm\_grade \times 0.30 + final\_exam\_grade \times 0.35$

- ❖ Recitation grade is based on both your attendance and participation. This grade will be decided by the group of assistants of your own recitation section. For that reason, *students* ***must attend their own recitation sections***.

- ❖ There won't be any make-up for take-home exams in any case, including medical health reports and official university activities.
- ❖ If you miss all of the take-home exams, midterm, and the final exam as well; then you will get an NA grade.
- ❖ If you miss one of the midterm/final examinations and if you do not take the make-up exam for that missing exam (check the make-up exam policy for details); then you will directly get an F grade.
- ❖ The instructor has the right to have an oral interview for any grading item given in the syllabus. Students who will have the oral interview may be selected randomly or according to a suspicious situation observed by TAs or the instructors. After having the oral interview, the grade obtained by the student may change in a positive or a negative way.

CodeRunner Feature of SUCourse+

We apply an automated grading process on the take-home exams. You can use CodeRunner to check your expected grade, before submitting your take-home exam.

CodeRunner can be pretty busy and unresponsive during the last day of the submission. Thus, leaving the submission to the last minute is not a good idea.

CodeRunner and Sample Runs together give a good estimate of how correct your implementation is. However, we may test your programs with hidden test cases as well, but you will still be able to see your final grade before submitting your work.

Submit your take-home exam via SUCourse+ ONLY! Any other methods (paper, email, etc.) are not acceptable, either.

The internal clock of SUCourse+ might be a couple of minutes skewed, so make sure you do not leave the submission to the last minute. Do not forget that "*No successful submission on SUCourse+ on time = a grade of 0 directly for that take-home exam*."

Make-up Policy

No make-up is allowed for take-home exams (THEs). Students automatically get 0 (zero) from the respective THE grade if any of them is missed.

Make-up is only allowed for the midterm and final examinations to those with an official report and to those with an official permission notice from the university on the date of the exam in question. Students must submit their reports/notices to one of the instructors **before** the exam in question. The ones having other excuses should contact the instructors within the day of the exam to be missed and then the instructors will decide whether these students are allowed to take the make-up exam. Any excuses to be brought to the attention of the instructors after the exam will **not** be considered. **No exceptions to these rules**!

Dates and details of the make-up examinations will be announced later.

Make-up examinations will be written and oral.

Exam/Submission Review Policy

Students are allowed to object to their midterm and final examinations, as well as their take-home exams. Time interval for each objection will be announced together with the respective grade. Grade bargaining will absolutely not be tolerated.

Plagiarism Policy (Academic Integrity)

Plagiarism means presenting someone else's work as yours. This is a very serious and ethical problem.

This part is prepared in order to explain the actions that yield to plagiarism, and the sanctions against it. Most of the actions and sanctions mentioned in this part will also be valid for CS201 and CS204 courses as well; but there might be small differences for the rules among these courses.
(http://people.sabanciuniv.edu/levi/cs204/policy_plagiarism.html)

A plagiarized work may or may not be a verbatim copy of another submission. Verbatim copies are of course plagiarized ones. However, if a submission is derived from another one by partially changing some parts, this action is also plagiarism. A common fallacy is that the graders cannot catch a program that is developed by partially changing another program. Believe it or not, such programs are caught very easily by using some special software.
From the above paragraph, it should be clear that "similar" submissions might be

treated as plagiarized ones. It is in your hands to avoid ending up with a "similar" submissions. Some precautions are as follows:

- Do not share your submission, wholly or partially.
- Do not discuss the crucial details of your submission with others.
- Keep any electronic/paper storage that includes your work in a secure place.
- Do not allow other people access your computer through sharing facilities and programs. Some cheaters use network scanners to search for computers with open shares and steal your data (happened in the past).
- Do not help or accept help that yields similar codes.
- Do not share your password.
- Do not enter your password in public computers (in the past, we have seen cases where the passwords and then the submission are stolen).
- Do not work on the submissions together.
- Do not have a friend make your submission.

When a plagiarism case is detected, sanctions are applied to all parties regardless of the actual source of the submission. These sanctions are as follows:

- **For the midterm/final examinations,**
    - **students directly fail the course, even in the first offense.**
- **For the take-home exams,**
    - **for the first time, all plagiarized submission owners receive 0 (zero),**
    - **the second time, the student fails the course automatically.**

Below you can find a preliminary classification of plagiarizers according to our experience:

- Hard-workers and lazy friends: This is the most common method. In this case, the hard-worker student makes the submission and gives it to his/her lazy friend(s). **But do not forget both of them are punished.** You have to be able to say "NO", if one of your friends asks for your submission. Moreover, please do not get fooled with arguments such as "*I will just have a look at it in order to see the main idea. Then I will do it by myself*". Even if your friend really tries to do so, (s)he may not succeed to make a legitimate submission and may submit a plagiarized submission derived from yours (this happened several times in the past). Moreover, your friend may share your submission with some other people and these other people may derive plagiarized submission out of yours. Please do not forget that your friend who takes your submission has nothing to lose. When (s)he is punished, (s)he would just accept it, but you, as the actual person who spent time and effort, will also be punished. This is really a very bad experience (just imagine it and you will see that it is really a very bad feeling). That is why **PLEASE DO NOT SHARE YOUR SUBMISSION**.

- Collaborators: More than one person makes the same submission with minor or major modifications. The common excuse for this category is "*we made the submission together*". In IF100, submissions are to be done personally. They are not group submissions. Another excuse is "*I applied the university motto: creating and developing together*". Our motto is a valid argument <u>if and only if</u> you can write all of the creators and developers' names on the final product (the submission in our case) and you can share all the benefits of that product. Since this is not possible in IF100, you cannot create and develop your submissions together.

- Deceived cheaters: This category includes people who pay other people to make their own submission. But the one who takes money makes the same submission for more than one person. So there are people that do not know each other but submit the same submission. Moreover, when we catch a student who pays or gets paid for such a submission trade, both parties are directly reported for an immediate disciplinary investigation. In this respect, we are in collaboration with freelancer sites, at which programmers are looking for jobs. We are immediately informed when our submission appears at these sites. These activities are both against the law and against university disciplinary regulations. All people involved in these activities (both the ones looking for freelancers and/or programmers to have a submission done and the ones who contact submission traders) will both be directly reported to the disciplinary committee and to the prosecutor's office.

- Forgetters: The ones who forget their submissions in public places. "*I forgot it someplace and X took it*" is not a valid argument. You are responsible for keeping your submission in a secure place. From the discussion above, it should be clear that even if you are not aware that your program files are taken by some other people, you still are responsible and subject to sanctions. Otherwise, the argument of "*I do not know how my files are taken*" is put forward by everybody and we cannot overcome the problem of plagiarism. Moreover, we, as the IF100 instructional team, do not have the right of investigating the actual source and the distribution mechanism of the submission in order to reach a guilty/not-guilty verdict. This is a legal problem. Therefore, in case of a situation where your submission files are taken without your consent, you have the right to report these other people to the dean's office for a disciplinary investigation. After this investigation, if you are proven to be innocent and the other people get some disciplinary penalties, then your grade will be given back. However, without your formal written complaint to the dean's office and a disciplinary investigation, no grades can change.

- Open computer sharers: Some people willingly/unwillingly open some part of their computers' disk or cloud accounts to the shared environment probably for multimedia exchange. However, if you are not careful and experienced enough, you may end up with a situation where your submission files are also in open share. This way, some other people may steal your submission files. Such incidents happened in the past. Thus, please do not open your computers' share. Our policy is the same as the previous bullet in case your submission is stolen in this way.
- Password sharers: Passwords are personal information. Even if you share it with your best friend, your private data is no longer secure. Moreover, passwords entered in public computers can be stolen. Thus, please enter your password only on your own computer. Our policy is the same as above, if your password, and then your submission is stolen in this way.

By this policy, we are not aiming to stop any kind of correspondence and cooperation among the classmates. Of course, the students will talk about their submission and discuss some solutions. However, the students should know where to stop these discussions. In that respect, there is a thin line between plagiarism and cooperation. And unfortunately it is not possible to quantify this line. So when you start cooperation, you are taking a risk of being treated as a plagiarizer. Please keep this in mind and do not get into the forbidden area.

It is the student’s responsibility to ensure that (s)he completely understands any material that (s)he submits and that (s)he is actively engaged in the production of the solution. The instructors and TAs of this course reserve the right to ask the students to explain the reasoning behind their work without the presence of any collaborators. Students should know that the ***written submitted work is not the only material that will be graded***. The instructors or TAs might **request a viva** (oral exam), and **grade it instead of the written submitted work**.

Additionally, cases of plagiarism will be directly referred to the Dean’s Office for disciplinary action. This course does not tolerate any breach of academic integrity (more info on https://www.sabanciuniv.edu/en/academic-integrity-statement).

Special Note from the Instructors:

Please don't think that this policy has been written under the classical instructor's thought of "*No students can fool us*". We can assure you that when a student cheats or submit a plagiarized take-home exam, we do not feel anything personally. We are just trying to behave fairly in a crowded course. When a student plagiarizes, (s)he may obtain some points that (s)he does not deserve. In this way, (s)he may unfairly pass some other (honest) students in the overall grade list. This may affect the letter grade of these honest students in a negative way. Our only concern is to prevent this kind of unfair earnings and loss of rights. In other words, we are trying to protect honest students. Thank you in advance for your cooperation.

Additional Notes

Students are responsible for every announcement made in lecture/SUCourse+ or sent via email. Students are expected to check their Sabanci University mail inboxes regularly as important announcements will be sent to them via email. Not attending the class, not following SUCourse, not checking emails regularly is not an excuse, in case they miss something.

Tentative Course Outline

Introduction & Motivation We will introduce the course to students and talk about our motivations. The reason behind this course being a university course (not for only CS students, but everyone) will be discussed and the importance of this course will be emphasized. Also, general idea of computational thinking will be covered and some examples from different disciplines are provided.

From Puzzles to Real World Problems Computational thinking includes 4 important concepts: (i) decomposition, (ii) pattern recognition, (iii) abstraction, and (iv) algorithm design. We will explain these concepts in detail. It is very important to understand the features of each aspect and their roles in solving a problem computationally. Several examples will be provided for these aspects, from different disciplines and daily life.

Developing Algorithms We will talk about designing algorithms in real problems. Writing pseudocodes and designing flowcharts will be covered. This way, components of Computational Thinking will be practiced with the examples without any coding. Additionally, Python will be mentioned for the first time to be used in the following weeks. We will be using Python to realize each one of the aspects of computational thinking separately in different topics.

Let the Coding Begin We will introduce the first Python program to the students. Syntax, variables, operators, standard input/outputs, basic problems with simple calculations, strings and lists will be covered. Besides, we will discuss using different libraries, especially for graph drawing purposes.

To Be or Not To Be: Making Decisions Some problems can be solved with basic operators and calculations, however some problems need decision making. Decision making is required when we want to execute a code only if a certain condition is satisfied, which is examined through Boolean expressions. Students will learn to determine which action to take and which statements to execute depending on the outcome. Decision making is a crucial point of "Algorithm Design", thus various examples from different disciplines will be covered.

String Like a Bee Strings and lists are amongst the most popular types in Python. We will discuss using some of Python's built-in methods in order to be able to manipulate sequences. It will be good practice for the "Abstraction" aspect of computational thinking.

The Copying Game In general, statements are executed sequentially. However, some statements may be executed depending on a decision, and there may also be a situation when a block of code needs to be executed several times. A loop statement allows us to execute a statement or group of statements multiple times. Loops are also a crucial point of "Algorithm Design", thus various examples from different disciplines will be covered.

Civilization with Modularity A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for our applications and a high degree of code reuse. In this week, students will learn to implement their own "user-defined" functions. Functions will be a good practice for decomposition and abstraction aspects of computational thinking. Additionally, we will discuss the scope of the defined and referred variables.

Let's Get Organized The most basic data structure in Python is the sequence. Each element of a sequence is assigned a number - its position or index. There are certain things you can do with all sequence types. These operations include indexing, slicing, adding, multiplying, and checking for membership. Some built-in functions of lists as finding the length of a sequence or finding its largest /smallest elements will also be covered. Additionally, tuples and its differences will be mentioned. Besides, Python dictionary is a container of the unordered set of objects. It is aimed to show the role of dictionaries in specific problems. Data structures have an important role for the abstraction of problems, thus should be emphasized under Computational Thinking approaches.

Verba Volant Scripta Manent (The faintest ink is more powerful than the strongest memory) Sometimes we need to read/write from/to files. Python provides basic functions and methods necessary to manipulate files by default. Students will learn these details.

Winter is Coming Some futuristic and eye-opening topics will be covered.