Form E-1-A for Boston College Core Curriculum

Department/Program : _Computer Science, June 2023

NOTE: At present, the Computer Science satisfies a university Core requirement with two courses—CSCI1101-Computer Science 1 and CSCI1080-Principles of Computer Science, both satisfies the Mathematics requirement. In Fall, 2019, we offered CSCI1701, and Enduring Questions course on the theme of privacy, paired with a course taught by faculty from the Law School. We have not offered this course a second time, largely because of difficulty coordinating the two instructors' schedules.

We have made an important change to our introductory offerings beginning in the Fall semester 2022: instead of having a single introductory course CSCI1101, we are creating two different courses: One, which will still be called CSCI1101, is intended principally for prospective Computer Science majors and minors, while the new course CSCI1080-Principles of Computer Science, is intended for non-majors or students that maybe considering the major and have no exposure to computer science or coding at all in high school. This is the result of our assessment of outcomes from the single-track CS1101.

1) **Have formal learning outcomes for the department's Core courses been developed? What are they?** (What specific sets of skills and knowledge does the department expect students completing its Core courses to have acquired?)

   For CSCI1101 a specific set of outcomes has been developed by the group of instructors teaching the course. Attached to this document is a recent working version of these outcomes used by the instructors in 2018-2019. For CSCI1080 a specific set of outcomes was used in the course development, that looks similar to CSCI1101, but the depth level of contents are really different.

   Where are these learning outcomes published? Be specific. (Where are the department's expected learning outcomes for its Core courses accessible: on the web, in the catalog, or in your department handouts?)

   The attached document concerning CSCI1101 circulated among the instructors, but some version of the desired outcomes appears in the individual syllabi for each section. A less-detailed description of learning outcomes appears in the course description posted on the department's website.

   https://www.bc.edu/content/bc-web/schools/mcas/departments/computer-science/academics/courses.html
   The document concerning CSCI1080 is part of the course syllabi that the instructor publishes on the LMS every year.

2) **Other than GPA, what data/evidence is used to determine whether students have achieved the stated outcomes for the Core requirement?** (What evidence and analytical approaches do you use to assess which of the student learning outcomes have been achieved more or less well?)

Since CSCI1101 is also a required introductory course for the major, we have tended to assess it through that lens, looking at how well it prepares students for the *next* course. The evidence here is gotten through observation of the students' work and discussion of what skills the students have acquired satisfactorily in the first course, where they appear to be deficient, etc.

In the course CSCI1080, students are asked to do a pre/post assessment, and the learning outcomes are measured in these assessments, and they are also measured in course homework's and evaluations.

3) **Who interprets the evidence? What is the process?** (Who in the department is responsible for interpreting the data and making recommendations for curriculum or assignment changes if appropriate? When does this occur?)

We hold a year-end meeting of the department where curriculum matters are discussed. For CSCI1101, the team of instructors for the course meets to discuss how the students are faring, and this often translates into recommendations for curricular change. For CSCI1080, the pre/post assessment is evaluated together with students' evaluations and the results are presented in a departmental meeting.

4) **What were the assessment results and what changes have been made as a result of using this data/evidence?** (What were the major assessment findings? Have there been any recent changes to your curriculum or program? How did the assessment data contribute to those changes?

Some years ago, we made a major change to CS1 by switching the language in which it was taught from Java to Python. This change was based on some years of observation by CS1 instructors that led us to conclude that Java put some significant learning obstacles for introductory-level students at the very beginning of the course. After a year or two of experience with Python, we eliminated the unit in CSCI1101 concerning object-oriented programming in Python, because we found that it crowded out other material, and tended to be confusing rather than helpful for continuing students who went on to study object-oriented programming in Java.

Also, see the preamble to this report, concerning the split of CS1 into separate majors and non-majors courses. This change came about as a result of our assessment, largely reports from instructors of intermediate and advanced courses that many of our majors were deficient in essential skills. It was felt that trying to serve the needs of students coming from a variety of academic backgrounds had the effect of holding the majors back.

5) **Date of the most recent program review.** (Your latest comprehensive departmental self-study and external review.) **April 2019**.

CSCI 1080 – Topics

This course starts with the use of visual programming language (BlockPy), and from week 8 forward the course starts to use regular Python with visual programming and during the last two weeks of course we move towards regular Python.

1) Numeric types (integer & float)

    a. Know the difference between integer and float, and how to convert between types
    b. Be familiar with arithmetic operators and expressions

2) Strings
    a. String slicing methods
    b. Other string methods: upper(), lower(), len()
3) Iteration – for loop and while loop
    a. Be familiar with for loops and while loops and the break command for "escaping" from them.
4) Conditional statements (if-elif-else statements
    a. Be familiar the relational operators: >, <, >=, <=, ==, and, !=
    b. Be familiar with how to construct complex expressions using and & or operators
    c. Be familiar with the basic if-elif-else statement, and nested conditional statements.
5) Defining function of your own
    a. You should be able to write a function and know the difference between printing the result vs using the return statement.
6) Lists
    a. Be familiar with creating lists, traversing lists, and using the **in** operator with lists.
    b. Be familiar with built-in list functions: append, remove, len. etc
    c. Know that lists are mutable, and can be updated.
    d. Python tuples versus Python lists
7) Creating Basic Python Functions
8) Using Basic Python Functions

General: abs bin chr float hex in int len list max min ord range raw_input randint sqrt str sum type also any math functions.

For Strings: capitalize(), count("") find(["", index]), islower(), isnumeric(), isupper(), replace("","") , upper(), lower()

Additions Functions For lists: append(x), count (x), index(), insert(i,x), pop(x), remove(x), reverse(), sort(x), split(), strip()

12) Dictionary Concept and Structure

CSCI 101 – Topics

0) Binary/Hex/Ascii

1) Numeric types (integer & float)

   a. Know the difference between integer and float, and how to convert between types
   b. Be familiar with arithmetic operators and expressions

9) Strings
   a. String slicing methods
   b. Other string methods: upper(), lower(), len()

10) Iteration – for loop and while loop
   a. Be familiar with for loops and while loops and the break command for "escaping" from them.

11) Conditional statements (if-elif-else statements
   a. Be familiar the relational operators: >, <, >=, <=, ==, and, !=
   b. Be familiar with how to construct complex expressions using and & or operators
   c. Be familiar with the basic if-elif-else statement, and nested conditional statements.

12) Defining function of your own
   a. You should be able to write a function and know the difference between printing the result vs using the return statement.

13) Lists
   a. Be familiar with creating lists, traversing lists, and using the **in** operator with lists.
   b. Be familiar with built-in list functions: append, remove, len. etc
   c. Know that lists are mutable, and can be updated.
   d. Basic List Comprehension
   e. Multidimensional lists
   f. Python tuples versus Python lists

14) Recursion/Iteration
   a. You should be able to define what recursion is, i.e. when a function calls itself.
   b. You should be able to write simple recursive functions (be sure to define a base case and a general case).
   c. You should be able to trace the calls of a recursive function.
   d. You should be able to write iterative and recursive versions of functions

15) Searching Algorithms (Linear Search & Binary Search)
   a. Be able to write binary search and trace the calls of the function.

16) Sorting Algorithms (Bubble/PushDown, Selection, Insertion, Merge/Sort)

17) Python Functions

General: abs bin chr float hex in int len list max min ord range raw_input randint sqrt str sum type also any math functions

Additional Functions For Strings: capitalize(), count(“”) find([“”, index]), islower(), isnumeric(), isupper(), replace(“”,””) , upper(), lower()

Additions Functions For lists: append(x), count (x), index(), insert(i,x), pop(x), remove(x), reverse(), sort(x), split(), strip()

11) Basic Exception Handling

12) Dictionary Concept and Structure

13) File Handling