

New Course or New Instructional Program Proposal

Directions:

Before completing this form, please discuss this proposal with the appropriate administrator(s) in your school. Complete this proposal form thoroughly, and attach any supporting documentation that would help the District Curriculum and Program Council understand this proposal better. Be sure that you adhere to all deadlines*, and be certain to acquire all required signatures. The deadline* for any course or program proposal that has budgetary implications and/or needs to be published in the NFHS *Program of Studies* is October 31, so please plan accordingly to make certain that all approvals of this application can be completed by October 31. All other proposals can be forwarded at any time of the year.

- 1. Please list the names and identify the school/department of those individuals who are making this proposal? If those making the proposal are not teachers, please explain thoroughly:**

Jay Greenberg – Technology/Business Education Department Chair
Stewart King –Technology Education Teacher

- 2. Give the title of new course or instructional program. Indicate the department in which this course/program will reside:**

Introduction to Programming – Technology Education Department

- 3. Please indicate if the new course or instructional program is a semester or year long, and indicate the applicable grade levels. Please indicate the course level if applicable:**

This course will be for students in grades 9 through 12. It is being designed initially as a semester-long course. This is an academic level course.

4. Please give the rationale for this proposal, and include its relationship to the past, current and future development of curricular offerings in New Fairfield:

Computers play a large role in the economy. Students with 21st Century Skills in programming concepts, such as abstraction, generalization, and being able to see the big picture, will be uniquely qualified to successfully maneuver themselves into a winning position in the quickly evolving 21st century.

Currently, the only programming class that is currently offered at New Fairfield High School is AP Computer Science A and it is geared toward satisfying the needs of students that intend to become professional computer programmers.

This course is not aimed at replacing the AP Computer Science A course. It is instead designed as an introductory programming course, geared toward a more diverse cross section of students by teaching coding along with effective problem solving; it will supplement our current offering by providing a stronger foundation for students entering AP Computer Science A who would not otherwise have a programming background.

By using computers to solve problems from a wider range of disciplines, this course will appeal to a wider group of students who want to learn about programming, but are not thinking of making it a career. This may include students who are interested in becoming scientists, engineers, business people, statisticians, and others.

5. Please indicate the target population for this proposal:

With a unique focus on creative problem solving in a wide range of disciplines, this course is designed for two groups: 1) Students that wish to become scientists, engineers, business people that may become managers of programmers and wish for essential understanding of programming, and perhaps even artists and musicians who wish to work within the digital arena without actually considering a career as a computer programmer, and 2) students who are exploring computer programming as a career and who plan to take AP Computer Science A.

6. Please explain if this course or instructional program is an addition or a replacement for an existing course or program.

This is a new course. It will serve as a precursor, but not a prerequisite, to the existing AP Computer Science A course.

7. List any prerequisite for this course or instructional program:

This course would be available to all high school students who have successfully completed Algebra I with a C or better.

8. Please write a short description of the new course or instructional program that would be suitable for the high school *Program of Studies* or for a curriculum document:

Introduction to Programming

This course uses modern programming tools to allow students to gain an understanding on how computation can help solve problems. The course begins with Snap!, a block-driven drag and drop programming language, to familiarize students with the programming process and then finishes with Python, an open-source professional programming language. This course is designed to be fun and rewarding for students who have no prior programming experience or who have some programming experience and want to improve their understanding. This course provides students with a foundation in programming that can lead to AP Computer Science A.

9. Please list (or attach a list) of the long-term course or program goals that define the broad outcomes that this course or program seeks to help students achieve:

After completing this course, students will have a basic understanding of computational thinking practices, including connecting computing in a meaningful way, creating computational artifacts, analyzing problems and artifacts, collaborating, and communicating. Students will also explore such concepts as outlines, creativity, abstraction, data and information, algorithms, the internet, and global impact.

10. Please indicate what topics, units, or material will be used to meet the long-term goals listed above.

Tentative Schedule:

Day 1

Unit 1: What is programming and why do we do it?

Topic: Very brief history of computers

Time: ¼ block

Essential Understanding: Computers were once crude, slow, and a tool for specialists. Now they are fast, more sophisticated, and a tool for the masses.

Topic: Class organization

Time: ½ block

Essential Understanding: Classroom procedures ranging from attendance to grading to anything else

Activities: Miscellaneous administrative tasks for students to complete

Topic: What is programming?

Time: ¼ block

Essential Understanding: Programming is a way to give instructions to a dumb machine

Activities: Write english instructions for simple things to practice developing an orderly sequence of steps

Day 2

Topic: Computer internals (at the most basic level)

Time: ¼ block

Essential Understanding: The electronics of a computer do not understand english or any human language. Computers must have a simpler and unambiguous language for communication

Topic: Programming languages

Time: ¾ block

Essential Understanding: There are a large number of programming languages; some are general-purpose and some are specific-purpose. At some point the programming language gets translated into something the computer can understand.

Activities: Students look at the Basic English word list and use only those words to define words not on the list.

Day 3

Unit 2: Programming From The Beginning

Topic: Basic control capabilities of a computer

Time: ½ block

Essential Understanding: Students understand that there are essentially only 4 types of program flow for a computer: Sequence, Selection, Repetition, Hand-off (routine invocation)

Activities: Students write english instructions incorporating the 4 basic types of program flow.

Topic: Data storage

Time: ½ block

Essential Understanding: Students understand that there are differences between numbers (integer, real, list, etc) and characters and that “1” is not the same thing as 1. Computers can store data in named “lockers.”

Activities: Students classify different types of data.

Day 4

Topic: Introduction to Snap! Programming (Univ of CA, Berkeley)

Time: 1 block

Essential Understanding: Students begin to become familiar with using 2 of the 4 basics of program flow through an introduction to the Snap! development environment (sequence, repetition).

Activities: Students use Snap! to construct very simple programs that use sequence and repetition programming flow “blocks.”

Day 5

Topic: Introduction to Snap! Programming, Part Two (Univ of CA, Berkeley)

Time: 1 block

Essential Understanding: Students begin to become familiar with using the 3rd of the 4 basics of program flow through an introduction to the Snap! development environment (selection).

Activities: Students use Snap! to construct very simple programs that use selection blocks.

Day 6

Topic: User interaction

Time: 1 block

Essential Understanding: For programs to be useful, communication with the user of the computer is a necessity.

Activities: Students write simple Snap! programs to communicate with the user, including displaying and accepting data from the user.

Day 7

Topic: Simple Math

Time: 1 block

Essential Understanding: Computers are good at specialized tasks.

Activity: Given the coordinates of where an olympic archer’s arrow lands, determine the arrow’s contribution to the archer’s score. (Details on program sheet)

Days 8-9

Topic: Delegation

Time ½ block

Essential Understanding: Students do not need to ‘reinvent the wheel’ but can write programs that are used in other programs. This is the 4th basic of program flow: hand-off (or routine invocation).

Activities: Students create simple Snap! programs that are then used in other Snap! programs.

Topic: Simple graphics

Time: 1½ blocks

Essential Understanding: Computers can be used to create simple graphics (which by extension of the 4 types of flow control could be used to create complicated graphics).

Activities: Students use the Snap! drawing tools to draw various simple geometric shapes based on user input. Geometric shapes are created by invoking student-created routines through selection.

Day 10

Topic: Nested Flow

Time: 1 block

Essential Understanding: Students learn that repetitive actions can be performed within other repetitive actions at any level desired.

Activities: Students write a program to display the individual digits as their program counts from 1 to 150.

Days 11-12

Topic: Simple music

Time: 2 blocks

Essential Understanding: Computers can create simple music (which by extension of the 4 types of flow control could be used to create complicated music).

Activity: Students research and then use the Snap! sound palette to duplicate or create a simple melody.

Day 13

Topic: Lists

Time: 1 block

Essential Understanding: Students become aware and understand the use of lists to hold program data

Activities: Students place the notes from their simple melody into a list and play them from the list.

Day 15

Unit 3: Programming with Python

Topic: Introduction to Python programming

Time: 1 block

Essential Understanding: Students understand that Python is a freely available programming language suitable for almost any environment, including for professional programming

Activities: Students use the editing environment and the programming to make their first Python program.

Day 16

Topic: Program flow in Python

Time: ½ block

Essential Understanding: Students begin to become familiar with using the sequence component of the 4 basics of program flow through Python.

Activities: Students use Python to construct very simple programs that use sequence flow control.

Topic: Basic data types in Python

Time: ½ block

Talk Away: Python has data types similar to Snap! and most other programming languages.

Activities: Students use simple statements to manipulate simple data types.

Day 17

Topic: Iteration

Time: 1 block

Essential Understanding: Students see that iteration is common across many programming languages

Activities: Students see and write various examples of iteration in Python

Day 18

Topic: Data as Lists

Time: 1 block

Essential Understanding: Much of the data manipulated by programs can conveniently be viewed as lists.

Activities: Students practise using the power of Python, such as list membership, addressing, slicing, etc. when dealing with lists

Day 19

Topic: Data as Lists

Time: 1 block

Essential Understanding: Students learn how to modify lists

Activities: Students practice inserting and removing members of the list, singly and in groups

Day 20

Topic: Prime Numbers in lists

Time: 1 block

Essential Understanding: Prime numbers are easy to compute

Activities: Students learn and use the sieve of Eratosthenes to compute prime numbers

Day 21

Topic: Functions

Time: 1 day

Essential Understanding: Students learn how to write reusable code

Activities: Students create various functions to be called from within main programs

Days 22-23

Topic: List Comprehension

Time: 2 days

Essential Understanding: Students learn a succinct way to manipulate lists using comprehension

Activities: Students combine comprehension with their own functions to act as filters of lists

Days 24-25

Topic: Lists within Lists

Time: 2 blocks

Essential Understanding: Lists in Python can be anything, including other lists

Activities: Students practice using Python constructs to manipulate lists within lists.

Day 26

Topic: Combined processing using matching and lists

Time: 2 blocks

Essential Understanding: Lists can help minimize solutions

Activities: Students complete the “Coin Change” programming exercise

Days 27-29

Topic: Permutations and ordering of lists

Time: 3 blocks

Essential Understanding: Permutations and combinations of lists can make problem solving easier

Activities: Students complete the “Isosceles Triangle Sticks” programming exercise

Days 30-31

Topic: Statistics through lists

Time: 2 blocks

Essential Understanding: Python can manipulate lists. How to calculate and interpret three common statistical functions: Mean (average), median, and mode

Activities: Students create lists of random numbers and use the three functions to compare datasets and draw conclusions about the importance and limitations of each of the three statistical functions.

Days 32-33

Topic: Dictionaries

Time: 2 blocks

Essential Understanding: Data can be managed and accessed by directly referencing it

Activities: Students create and update dictionaries using dictionary keys and end with the “Friendly Numbers” programming exercise.

Day 34

Topic: Sets

Time: 1 block

Essential Understanding: Sets have their use for data analysis

Activities: Students create sets from data and use them for analysis purposes

Days 35-45

Topic: Individual project

Time: 1-10 blocks

Essential Understanding: Problem solving with Python

Activities: Students take a problem from one of their other classes, either this year or the past and use Python to create a solution to the problem.

How will technology be utilized to enhance the course or program goals?

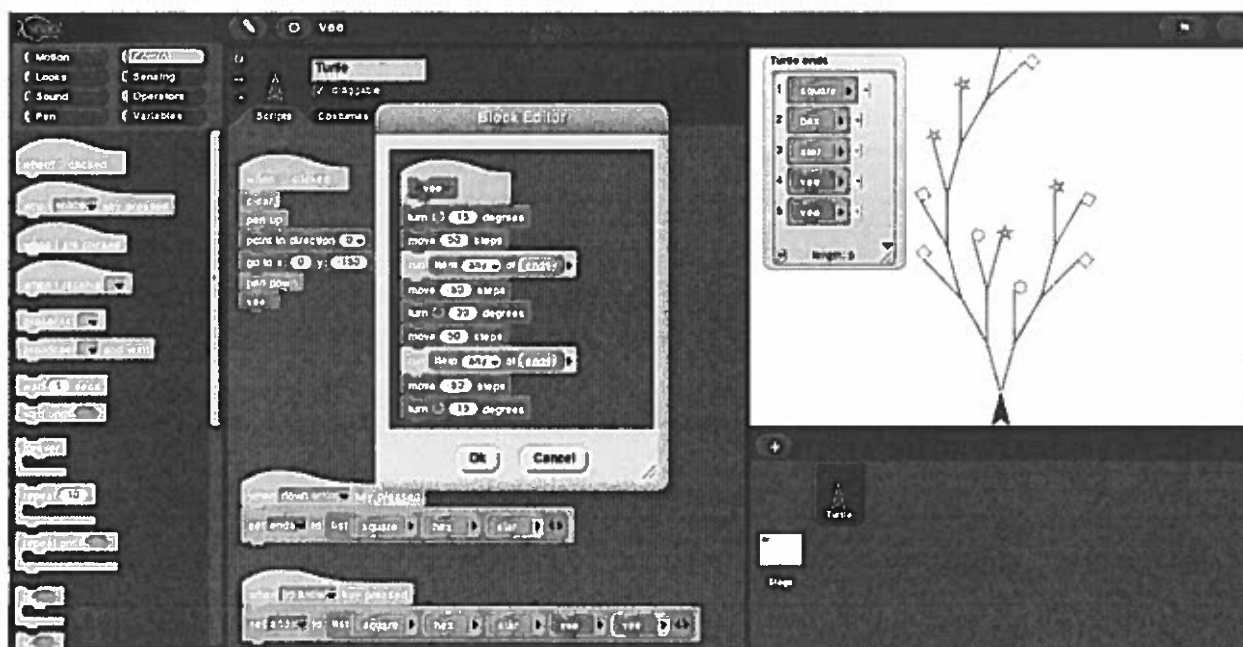
Because this is a computer programming course, computer technology will be used to as the primary component to assist the students in this course. Additionally, directed research using internet databases available through the school library will be a significant component of the course.

What assessment strategies will be used in this course or program?

The course is both task-based and project-based and students will be required to work in small groups and teams to complete projects. The tasks will be smaller, but meaningful mini-projects that will both have self-meaning and also prepare students for the techniques needed to tackle bigger projects. Each project will have a detailed description (as might be expected in an industry workplace) and a scoring rubric so that students are clear about what is expected of them and how their efforts and results will be graded. Some of the projects are student-generated and students will need to develop their own specifications and scoring rubric before the project starts. Students will make frequent presentations of their work to their peer colleagues and have the opportunity to receive feedback from both the teacher and their colleagues.

What are the unique components of this course or program content that makes it a worthwhile addition for our students?

This course will still give students a rigorous, but general, background in programming, starting with a drag-n-drop block language (Snap!) for the initial concepts and then moving to using the Python programming language. Snap! is powerful enough to solve major problems in computer science and brings all of that to the forefront in a fun easy to follow user interface that allows people to interact directly with the computer. The usability of Snap! avoids some of the dread students can experience with learning to program.



A sample Snap! user interface screen. *Illustration from <http://snap.berkeley.edu/>.*

This course will appeal to students with a wider range of interests than the AP Computer Science A course by not focusing solely on algorithms and data structures, but instead focusing on problems from a wide cross-section of problems in science, business, history, language, health, mathematics, astronomy, and others that can be solved with computing. Python is a language used widely as a tool in many industries that avoids requiring much of the ‘administrative details’ that a language such as C or Java requires. Python runs on every commonly available platform (Windows, Mac, Linux, etc.), and is available at no cost.

11. Please indicate any special location needs, such as the computer lab:

This course will be taught in Room 108 or Room 109 at the High School

12. Please enumerate the resources – both human and financial – that you anticipate will be needed to develop this course or program correctly. Please indicate any special training that will be necessary to implement this course or program, and give the cost of this training:

It is anticipated that this course will require the reallocation of one section taught by current staff at the high school based on student registration.

13. Please give the title and cost of the proposed text and attach it, if possible. Indicate any special equipment needs for this course and the anticipated cost of this equipment:

Most of the material will be assembled from *Wiki University, Khan Academy, MIT Open Courseware, UC Berkeley, edX, and Udacity* depending on the particular topic being presented in class.

14. Please address the questions below separately, and then attach your responses to this form:

a) What impact will this course/program proposal have upon other courses/programs currently being offered in the district?

- The impact to other courses should be minimal. It may draw some students from the other electives.
- This course may cause an increase in enrollment of the AP Computer Science A course presently being taught by the Math department at the high school, since this course will be a natural lead-in for that course. It may enable revision to the AP Computer Science A scope and sequence based on increased student understanding of programming prior to entering the course.

b) What impact would this proposal have on scheduling, staffing, and resources?

- The impact on scheduling should be minimal, as this course will more than likely be a singleton. Should the program gain in popularity in the future, then teaching resources might be affected as additional sections are added.
- There will be no consumable resource needed for this course other than occasional printing via the school printers already installed.

c) Do you anticipate that this course/program will have an impact on feeder programs and follow-up courses/programs currently being offered in the district?

- This course should help to increase enrollment in the AP Computer Science A - Java Programming course as this is a natural lead-in for that course and will introduce additional students to programming in general.


d) What do you anticipate will be the impact – in terms of new print and non-print materials on the library/media center?

- There will be little to no impact on library resources as the research material for this course are available on the internet.
- e) **Would adoption of this course/program proposal require specific staff adjustments, such as hiring new staff or retaining veteran staff?**
- None anticipated.

Signatures of those making this proposal: (The signatures indicate that all parts of this proposal have been thoroughly completed.)

 Date: 1/16/18

Signature of Department Chair indicating approval (if applicable):

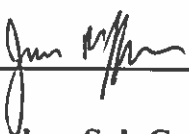
 Date: 1-16-18

Signature of Principal indicating approval: (Please note that this proposal must bear the principal's signature before it can be sent to District Curriculum and Program Council.)

 Date: 1/16/18

District Curriculum and Program Council Discussion Summary:

Signature of Assistant Superintendent indicating approval:

 Date: 1/16/18.

Curriculum Sub-Committee of the Board of Education Discussion Summary: