Brock University

Activity Summary*

Manipulating Expressions and Solving Equations (Part I: Intro)

(Assuming students have completed the Python Basics activity)

**Activity summary format inspired from the ScratchMaths program and their presentation of the learning objectives by using the 5E's pedagogy*

Learning Objectives:

- **Explore** how polynomials can be added, subtracted, multiplied, and divided using Python
- **Explore** how to solve and check solutions of first-degree equations using Python
- **bridgE** some of the processes of by-hand computations and programming-based computations

Activity Description:

Adding and Subtracting Polynomials

- Students read two examples that illustrate a method for adding or subtracting polynomials by hand.
- After importing the required modules and declaring the variables they will use later in the activity, students are introduced to two approaches for adding or subtracting polynomials in Python.
- The first approach (using *For Loops*) shows students how they can write their own code for adding or subtracting polynomials, which may help them to better understand the process (i.e., it is all about adding or subtracting the coefficients of like terms!).
- The second approach demonstrates Python's capabilities to perform addition and subtraction of polynomials for us. It is suggested as a manner of checking the results of the first approach.
- Students are provided with some questions to practice both coding approaches.

Multiplying and Dividing Polynomials

- Once again, students are provided with some examples to read, which illustrate the by-hand process of multiplying or dividing polynomials by a number or a monomial.
- In a similar manner as above, students are shown how multiplying (or dividing) a polynomial by a number can be done through two coding approaches, and they are invited to practice these.
- When it comes to dividing (or multiplying) a polynomial by a monomial, students are instructed to use only the second approach: i.e., asking Python to perform the division (or multiplication) directly. A practice question about division shows students how they may sometimes need to call upon Python's *simplify()* function to obtain the result.
- In an optional extension, students may see the Python code that can be used to multiply binomials.

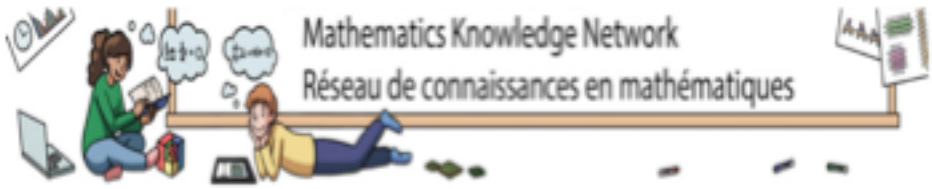
Mathematics Knowledge Network
Fields Institute for Research in Mathematical Sciences
222 College Street, 2nd Floor, Toronto ON, M5T 3J1

Solving First-Degree Equations

- Students are provided with an example of how to solve (and check the solution of) a first-degree equation by hand using the method of inverse operations (and substitution).
- Students are then shown (with opportunity to practice) how to use Python to solve and check the solution of a first-degree equation. This includes how to use Python's *solve()* function.

Links to Curriculum Expectations:

Curriculum Expectations	Links to Jupyter Notebook
Add and subtract polynomials with up to two variables [e.g., $(2x - 5) + (3x + 1)$, $(3x^2y + 2xy^2) + (4x^2y - 6xy^2)$], using a variety of tools (e.g., algebra tiles, computer algebra systems, paper and pencil).	- Students are required to use knowledge of how to add and subtract polynomials using paper and pencil to write code that will perform the same actions. - Students are required to build their knowledge of the different ways they can use Python to add and subtract polynomials. - Students are required to work with polynomials with one and two variables.
Multiply a polynomial by a monomial involving the same variable [e.g., $2x(x + 4)$, $2x^2(3x^2 - 2x + 1)$], using a variety of tools (e.g., algebra tiles, diagrams, computer algebra systems, paper and pencil).	- Students are required to use knowledge of the distributivity property to write code for multiplying a polynomial by a number. - Students are required to build their knowledge of how to use Python to multiply a polynomial by a monomial involving the same variable.
Solve first-degree equations, including equations with fractional coefficients, using a variety of tools (e.g., computer algebra systems, paper and pencil) and strategies (e.g., the balance analogy, algebraic strategies).	- Students are required to build their knowledge of how to solve (and check the solution of) a first-degree equation using Python.

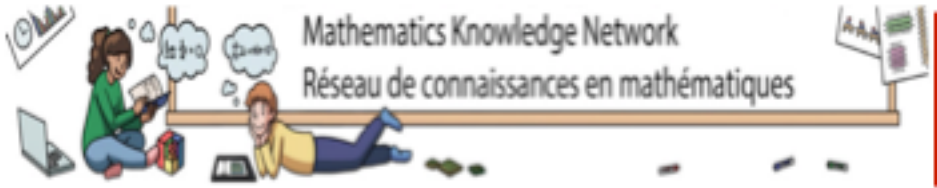


Discussion Points:

- How do we add polynomials by hand? How can we add polynomials using Python? What are the differences between the approaches?
- In *Example 1*, we were shown that the polynomial $x^2 + 2x - 3$ can be represented by $[1,2,-3]$ and the polynomial $x^2 + 6$ can be represented by $[1,0,6]$. Where does the 0 come from? Is it necessary? Could we have represented the polynomial $x^2 + 2x - 3$ by $[-3,2,1]$? Note: Representing a polynomial by an array is linked to a more advanced mathematical concept called a *vector*, which you may see in later studies of mathematics (e.g., in *Linear Algebra*)!
- How do we multiply a polynomial and a monomial by hand? Why can't we use loops to code this process in Python (like we did for adding polynomials)?
- What are the advantages or disadvantages of having Python perform our operations for us? For instance: Is it always faster to use Python? Should we always trust Python? …
- What does it mean to "solve" an equation? How does the *solve()* function in Python work?
- What are some ways we can check to see if we have correctly solved an equation?

Things to Note:

- This activity is geared towards students with very little coding knowledge and some basic mathematical knowledge related to the activity. You may want to have students working independently or in groups depending on their skill levels in Python and in mathematics.
- If students are having difficulty getting their code to work, ensure that they are reading the error message from Python and checking for syntax errors (e.g., a missing bracket, comma, colon, etc.).
- This activity uses many of the coding concepts that are introduced in the Python Basics activity. This said, there may also be some new elements of code introduced in this activity. We encourage you and your students to read the comments that are provided throughout the activity (in and around the coding cells), where we describe some of the coding elements used. If there is an element of code that you still do not understand, you may need to refer back to the Python Basics activity or try looking for information online (e.g., you can Google "numpy.ones() python" to learn more about what Python's *numpy.ones()* function does).



References

http://www.edu.gov.on.ca/eng/curriculum/secondary/math910curr.pdf
https://scratch.mit.edu/about/

The MKN is funded by the Ontario Ministry of Education. The MKN is a KNAER Project hosted by the Fields Institute for Research in Mathematical Sciences. The views expressed in this document belong to the authors and do not necessarily reflect the opinions of the Ministry of Education nor the Ontario government.

Activity created by Tyrell Nurdin, Lauren McCann, and Lauren May for MATH 3P41 at Brock University under the ***Math Knowledge Network.*** Coordinated by Dr. Chantal Buteau (Brock University). Amended by Lauren McCann and Tyrell Nurdin. June 2020.

Mathematics Knowledge Network
Fields Institute for Research in Mathematical Sciences
222 College Street, 2nd Floor, Toronto ON, M5T 3J1