Burazin, A., Gula, T., & Lovric, M. (2023). Using coding to enhance numeracy teaching and learning. In Online Proceedings of the *Coding, Computational Modeling, and Equity in Mathematics Education Symposium*, St. Catharines (Canada), April 2023.

Using coding to enhance numeracy teaching and learning

Andrijana Burazin[1], Taras Gula[2], Miroslav Lovric[3]
[1] University of Toronto, Mississauga; [2] George Brown College;
[3] McMaster University, lovric@mcmaster.ca

The purpose of this paper is to suggest coding as a means of working on *numeracy tasks*, which, in turn, can be used to enrich mathematics instruction. We believe that the non-traditional approaches and intuitive reasoning that are natural components of a numeracy task should establish their place in solving problems in mathematics.

What is numeracy? There is no consensus on the definition of numeracy (Geiger, Goos & Forgasz, 2015), and its relation to quantitative literacy, quantitative reasoning, and mathematical literacy remains ill defined. As well, mathematics-related terms, such as number sense, sense making, mathematics in context, and even modelling, are sometimes used to characterize or describe numeracy. The view that numeracy and mathematics are two different entities has been gaining attention and influence. Steen (1997) writes: "Numeracy is not the same as mathematics, nor is it an alternative to mathematics. Today's students need both mathematics and numeracy. Whereas mathematics asks students to rise above context, quantitative literacy is anchored in real data that reflect engagement with life's diverse contexts and situations."

Perhaps the most illustrative way of conceptualizing numeracy is to view it as living in the space between the school mathematics and the mathematics that is used in real-life situations. The transfer from the abstract (as done in a math classroom) to the concrete (math in the real world), once thought to be automatic and unproblematic, turned out to be anything but. This transfer is the key defining feature of numeracy; in other words, numeracy is about linking concrete quantitative situations to abstract mathematics (specifically arithmetic); see Gula & Lovric (2023) for a discussion about numeracy and its development.

Rather than trying to say what numeracy *is*, by presenting case studies of numeracy tasks, we discuss what numeracy is *about*. Hence, our focus, and this contribution, are on the teaching of numeracy. Teaching numeracy at various levels of education - from elementary, to tertiary, to adult - remains a challenge that has been addressed by (in our view, still incomplete) research and debate. Geiger, Goos & Dole (2011) write: "far less is known about how teachers [of numeracy] learn about, appropriate and then create effective mathematics teaching practices."

What is a numeracy task? Unlike a mathematics task, which is about mathematical ideas and objects and their relations (and which lives in the abstract), a numeracy task is about concrete, context-centred phenomena (using the abstract to make sense of the concrete). It is different from a typical word problem in mathematics: although a math word problem might sound like it is about the real world, deep down it is about some math concept and often requires a particular mathematical technique (see the first example in our discussion). A numeracy task is about a problem involving quantities (numbers) that a real person would indeed meet in their professional or personal life, and whose solution is obtained using the means that the actual person in that situation would, or might, use. In Gula & Lovric (2023) and Burazin, Gula & Lovric (2023) we conceptualize numeracy tasks and discuss ways of judging whether a given task is a good numeracy task. Borrowing from the Wolfram's (2016) computational thinking model, we view a numeracy task as having four essential components: define – abstract – compute – interpret and communicate.

We start by analyzing a sample mathematics textbook word problem, to serve as a contrast to numeracy tasks that we discuss.

A Math Textbook Word Problem (and a bit more)

The following mathematical task, or its variations, can be found in every calculus textbook:

> *Find the dimensions of a rectangle with perimeter 100 m whose area is as large as possible.*

This particular version is from Stewart (2021; page 332); sometimes, the problem is presented as a word problem, for instance about a farmer who has a fixed amount of fencing, and wants to fence off the largest possible rectangular piece of land. (Note: although this might sound like a real life situation, it is certainly not – when buying a piece of land, are we ever told: "Here is a rope, whatever you can enclose with it is yours.")

As this problem appears in the optimization section in the textbook, students will, most likely, solve it by writing equations. Denoting the length of the field by l and its width by w, the perimeter satisfies $2l + 2w = 100$, and one has to maximize the area $A = lw$. By combining the equations A is reduced to a function one variable, and students will follow the usual (expected!) strategy: differentiate and find critical points. Although it will likely yield a correct answer, this method does not give any insights about why the answer makes sense.

An alternative approach is to experiment: assume that the length is $30m$, and the width is $20m$ (so that the perimeter is $100m$). The area is $20 \cdot 30 = 600m^2$. If we increase the length, say to $35m$, and decrease the width to $15m$ (keeping the perimeter at $100m$), the area is $525m^2$. So that does not work. Let's try to make the longer side a bit shorter: if we decrease the length to $28m$ and increase the width to $22m$, the area is $28 \cdot 22 = 616m^2$, which is larger than the original area! If we further decrease the longer side, say, to $27m$, then the shorter side is $23m$, and the area further increases to $621m^2$. So it seems that if we take away from the longer side and give to a shorter side, we can increase the area. We can do this as long as there *is* a longer side! Thus, when both sides are equal (and equal to $25m$ in this case), we get the largest area of $625m^2$.The answer is the most symmetric of all rectangles – the square.

Now let's use coding. This short code in Python

```python
length=0.0
print("  length","    width","      area")

while length<50:
    width=50-length  # 2length+2width=100
    area=length*width
    print(f"{length:8.3f}{'':2}{width:8.3f}{'':2}{area:9.3f}")
    plt.scatter(length,area,color='b')
    length+=1
```

runs, in a loop, the length of the rectangle from 0 to 50, calculates the corresponding width and the area, and prints in a table, parts of which are below:

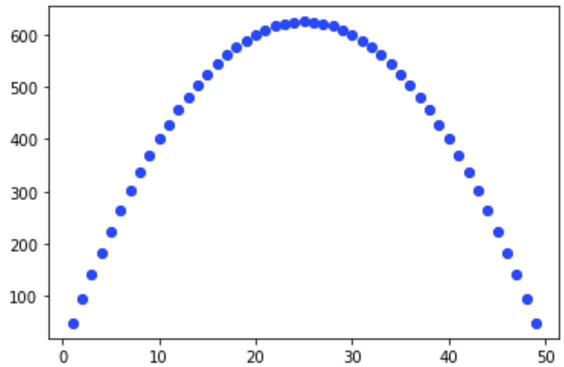
```
  length     width       area
   0.000    50.000      0.000
   1.000    49.000     49.000
   2.000    48.000     96.000
   3.000    47.000    141.000
   4.000    46.000    184.000
   5.000    45.000    225.000
```

…

```
22.000    28.000    616.000
23.000    27.000    621.000
24.000    26.000    624.000
25.000    25.000    625.000
26.000    24.000    624.000
27.000    23.000    621.000
28.000    22.000    616.000
29.000    21.000    609.000
```

…

In our case, the length starts at 0 and is incremented by 1 (of course, the increment can be changed to non-integer values). As well, the code produces a graph of the area as a function of the length, which shows an obvious symmetry (with additional lines of code we could annotate the graph, change the domain of the function, etc.):



This example illustrates that even a mundane math word problem can be repackaged into a rich experience. Non-traditional approaches (informal, or not), typical for numeracy tasks (as we will see shortly), stimulate creativity and enhance understanding. This is our point: numeracy tasks and approaches can enhance and enrich mathematics teaching and learning.

In the remaining part of this paper we discuss several numeracy tasks where one can use coding. To make the presentation shorter, we avoid giving a full, detailed real-life context. As in the symposium workshop, a reader is encouraged to look at the code, copy into a Python notebook, run it, and then modify and experiment.

Four Case Studies of Numeracy Tasks

Example 1 Good decision, or not? (Modified from a post on the Personal Finance subreddit).

> *There was a good opportunity, so I bought a $6000 leather couch on a sale for $5000. I paid with my visa, because I had no cash to spend. I can afford about $200 a month to pay off my credit card debt. Am I being reasonable?*

Of course, there is a formula somewhere that could answer this question. But that's not the point (and, quite likely, the person in this story would not be able to find it and use it). Here is the approach we suggest: we will follow what happens to the $5000 over time, neglecting all other charges that could have been made on the credit card. From the cardholder agreement or a monthly statement we learn that the annual percentage rate (APR) is 19.99%.

The cyclic nature of the problem:

> balance at the start of the month → payment of $200, assumed to be done on the first day of the month → interest accrued over the month → balance at the end of the month, equal to the balance at the start of the next month

suggests algorithmic approach, so this is a native coding problem.

The first step in approaching a numeracy task is to abstract it, i.e., strip of all inessential information. Hence the assumption that the payment is applied on the first day of the month, and (to simplify code) taking a month to have 30 days, and a year to have 365 days. (Challenge: modify the code to eliminate these assumptions.) From the Cardholder Agreement we read:

> The amount of interest we charge you on each account statement is calculated as follows:
>
> - first, we determine your average daily balance by adding the interest-bearing amount you owe each day and dividing that total by the number of days in the statement period
> - next, we determine the daily interest rate by dividing the annual interest rate by the number of days in a year.
>
> Your interest charge is then calculated by multiplying the average daily balance by the daily interest rate by the number of days in a statement period.

Now we have all information we need to code:

```
apr=0.199
balance=5000
monhtly_payment=200
month=0
total_interest=0

while balance>0:
    month+=1
    balance=balance-monhtly_payment
    interest=(apr/365)*30*balance
    if balance>0:
        total_interest+=interest
    balance+=interest

print("Debt paid after",month,"months.",f" Total interest paid={total_interest:6.2f}")
```

```
Debt paid after 32 months.  Total interest paid=1345.38
```

Using a while loop we are able to control the iterative process, so that it stops when the balance reaches zero. Because of our simplifying assumption, the balance is the same for all days in a given month, and hence it is the average daily balance. The calculation shows that the total interest paid ($1345.38) exceeds the sale difference ($1000); so in reality, the person will have to pay more that the full price for the couch. As well, it will take 32 months to bring the balance (for that purchase only!) to zero.

By adding a print statement, we obtain a listing which shows the dynamics of the interest accrued and the remaining balance:

```
month=  1   start balance= 4800.00   interest= 78.51   end balance=4878.51
month=  2   start balance= 4678.51   interest= 76.52   end balance=4755.03
month=  3   start balance= 4555.03   interest= 74.50   end balance=4629.53
month=  4   start balance= 4429.53   interest= 72.45   end balance=4501.99
month=  5   start balance= 4301.99   interest= 70.36   end balance=4372.35
month=  6   start balance= 4172.35   interest= 68.24   end balance=4240.59
month=  7   start balance= 4040.59   interest= 66.09   end balance=4106.68
month=  8   start balance= 3906.68   interest= 63.90   end balance=3970.58
month=  9   start balance= 3770.58   interest= 61.67   end balance=3832.25
```

Having a functioning code means that we can experiment. For instance, by changing the monthly payment to $500, we realize that the total interest the person would pay is a bit over $400, and the balance will be paid off fully in 11 months.

```
apr=0.199
balance=5000
monhtly_payment=500
month=0
total_interest=0

while balance>0:
    month+=1
    balance=balance-monhtly_payment
    interest=(apr/365)*30*balance
    if balance>0:
        total_interest+=interest
    balance+=interest

print("Debt paid after",month,"months.",f" Total interest paid={total_interest:6.2f}")
```

```
Debt paid after 11 months.  Total interest paid=408.10
```

Example 2 Simulating rare events (communicated by a colleague physician; to protect privacy, we do not identify the source, nor the actual disease or the school board in question)

> *In the case of an identified epidemic in a school, school district board administrators need to decide on the course of action, which might include closing all district schools. On average, every year 2 students in a large district school board in Ontario are diagnosed with a medical condition (we call it MC), which could cause an epidemic (such as meningitis). In the first eleven months in 2022, four children in that board have been diagnosed with MC. What should school administrators do? Declare an epidemic and close schools, or keep them open? The school board in question has about 200,000 students. The known incidence of MC is 1 in 100,000 per year.*

The number of students diagnosed with MC changes from year to year: there are years when no students are diagnosed, and there could be years when 2 or 3, or more students are diagnosed with MC.

No one can know for sure whether or not an epidemic is about to break. Math cannot help us answer that question either. But what can it do? A good question to ask is: what is the chance (i.e., how likely is it) that 4, or more, students being diagnosed with MC in one year happened just by chance, rather than being due to an actual epidemic?

We model this situation using probability. One simulation (lines 9-11 in the code below) checks each student every day of the year for the presence of MC (in order to speed up calculations, we used the units of thousands; hence 200 represents 200,000 students). It does so by picking a random number between 0 and 100*365-1, thus implementing the probability 1/100*365 that a randomly selected student has been diagnosed with MC on a given day (if the randomly selected number is 0, the student has MC, otherwise not).

This loop is then run 500 times (so performing 500 repeated experiments), and the number of times the number of students with MC equals, or exceeds, 4 is recorded.

In this particular run of the code, 4 or more cases occurred 71 times, and so we conclude that 4 or more cases of MC occurred by pure chance (i.e., not due to the epidemic) is 71/500, or 14.2%.

```python
num_students=200  # units are thousands
day_value=100*365  # incidence 1/100*365/day; need to account for the avg of 2
num_simulations=500  # number of simulations
count_4ormore=0   # number of simulations that ended with 4 or more cases

for k in range (num_simulations):
    meningitis_cases=0
    for i in range(365):
        for j in range(num_students):
            if random.randrange(0,day_value)==0: # positive case identified
                meningitis_cases+=1
    if meningitis_cases>=4:
        count_4ormore+=1

print("In",num_simulations,"simulations, 4 or more cases occured",count_4ormore,"time(s)")
print("The chance of 4 or more cases occurring is",count_4ormore/num_simulations)
```

```
In 500 simulations, 4 or more cases occured 71 time(s)
The chance of 4 or more cases occurring is 0.142
```

This is how far quantitative thinking goes. Now, with this useful piece of information, it is up to the school administrators and health care professionals to decide on the course of action.

Note: Using statistics, we can model this situation with a Poisson distribution with the parameter $\lambda = 2$. In that case, we compute the chance to be

$$1-\frac{e^{-2}\cdot 2^0}{0!}-\frac{e^{-2}\cdot 2^1}{1!}-\frac{e^{-2}\cdot 2^2}{2!}-\frac{e^{-2}\cdot 2^3}{3!}=0.14288$$

(see Lovric (2012) for details) i.e., a about 14.3%. Thus, our simulation was pretty close! (Note: the second time we run it, we obtained 14.8%).

Example 3 Understanding the Mean (What to do when a TA is late grading their papers?)

There are four sections in a large university course. The course instructor, about to start their lecture, wishes to discuss the test that the students wrote two weeks ago. However, one of their TAs has not finished marking. Based on the 300 marked tests, the instructor calculated the mean to be 67.6%. Rather than waiting for the TA to finish marking the remaining 100 tests, the instructor decides to report the average of 67.6% for the entire class. Find an argument to support their decision.

Obviously, the issue is the following: if we know something about a sample (i.e., a subset) of a population, what (if anything) can we say about the population itself?

Suppose that a population consists of five numbers, of which we know four: 5, 6, 8, and 17 (hence, this is a sample). Is there anything we can say, with some certainty, about the mean of the population of all five numbers, if the mean of this sample is 9? If the unknown number is 1, the mean is 37/5=5.4. If the unknown number is 9, the mean is 9, and if the unknown number is 27, the mean is 63/5=12.6. So, the mean of the five numbers (population mean) could be (lot) smaller, equal or (lot) larger than the mean (sample mean) of the known four numbers, and we cannot claim anything, with certainty, about the relation between the two means.

But in this case, the sample of 300 is much larger than 4. Would that change anything? To figure it out, we write a Python code to generate 400 random test grades (assumed to be normally distributed with a mean of 67 and a standard deviation of 30) and store them in a list (if the grade randomly generated is smaller than zero, we change it to zero). Then we calculate the means of the first 300 grades in the list, and compare it with the mean of all 400 grades. Here is the code:

```
import numpy as np
import numpy.random as nrand
from statistics import mean

grades=np.round(nrand.normal(67,30,400))
for i in range(len(grades)):
    if grades[i]<0:
        grades[i]=0
print(grades)

print("The average grade of 300 tests is",mean(grades[:300]))
print("The average grade of all 400 tests is",mean(grades[:400]))
```

To illustrate, we show a part of the list that contains students' test grades, and then the desired output:

```
[ 91.  75.  57.  98.  13.  63.  65.  33.  83.
  32. 138.  79.  53. 103.  90. 101.  25. 110.
 129.  40.  96. 116.  36.   0.  83.  52.  70.
  56.  56.  85.  38.  52.  62.  49.  97.  58.
  67.  29.  54.  62. 127.  88.  84.  56.  65.
  29.  17.  69.   7.  98.  51.  93. 108. 110.
  39.  14.  91.  69.   0.  61.  44.  22.  98.
The average grade of 300 tests is 67.59666666666666
The average grade of all 400 tests is 67.98
```

So, the two averages are close to each other. Running the code over and over again, we obtain different means, but in all cases the sample average (300 grades) is close to the population average (all 400 grades). In conclusion, the course instructor can report, with confidence, the average of 300 tests as being close to the entire class average.

The fact that this is so is not a coincidence, but a reflection an important fact from statistics. To get a bit better feel about it, we ask the following question: How does the class mean behave if we compute the mean of the first two tests in the list, then the mean of the first three tests, then the mean of the first 4 tests, and so on? Again, a code will help us figure it out: we generate a list of 400 random grades, and then, in a loop compute the means:

```
import numpy as np
import numpy.random as nrand
import matplotlib.pyplot as plt
%matplotlib inline
from statistics import mean

grades=np.round(nrand.normal(67,30,400))
for i in range(len(grades)):
    if grades[i]<0:
        grades[i]=0
print(grades)

averages=[]
for i in range(400):
    averages.append(mean(grades[:i+1]))

plt.plot(averages)
plt.show()
```

Here are the first few grades from the list:

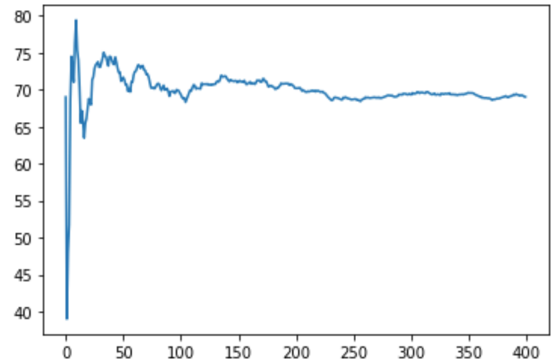
```
[ 69.   9.  66.  63. 135. 105.  64.  57. 119.
```

and their averages below: 69 is the average of the first grade, 69; the next term is (69+9)/2=39, followed by (69+9+66)/3=48, (69+9+66+63)/4=51.75, and so on.

```
print(averages)
```

```
[69.0, 39.0, 48.0, 51.75, 68.4, 74.5, 73.0,
```

The plot (the number of test grades on the horizontal axis and the corresponding mean on the vertical axis) reveals what is going on: after initial large fluctuations, the means stabilize, and approach the mean of the entire population:



In conclusion, if a sample is large enough, its mean is a good approximation of the mean of the entire population (in statistics, this is called the Law of Large Numbers).

Example 4 Understanding growth: size and volume (or, how mixing up the meaning of the two lead to a major error in a research paper)

> *In the article "Is clinical breast examination an acceptable alternative to mammographic screening?" published in the British Medical Journal (Mittra et al, 2000), we read:*
>
> > *[...] if you consider the exponential growth rate and doubling time of breast cancer you find that a single breast cancer cell has to undergo 30 doublings to reach a size of 1 cm, when it will contain 10^9 cells and be clinically palpable. Since the average size of a non-palpable, mammographically detected cancer can be assumed to be about 0.5 cm, the lead time gained by mammography over clinical breast examination would be of the order of only one doubling. Whether this lead time equivalent of one doubling in the natural course of 30 doublings would lead to a significantly greater reduction in mortality is questionable.*
>
> *Find a major flaw in this reasoning, thus alerting the physicians not to undervalue the significance of the lead time gained by mammography.*

Note that the term *size* in the article refers to a linear dimension (units are cm). Doubling time is the time needed for cells to double in number, and hence to double the volume. Mixing these two concepts is what lead to the error in this paper.

We write code to understand the exponential dynamics of the cell growth better. We assume (as is standard in practice) that a single cell generates two offspring cells, and that the cells accumulate in the form of a sphere. In the code we track the generation, i.e., count the doublings, and record the number of cells, the length and the volume of the cancer:

```
nocells=1
length=1/1024 #size (diameter) of a cell in cm

print("generation","  number of cells","  length (cm)","    volume (cm^3)")

for generation in range(1,31):
    nocells*=2
    length*=2**(1/3)
    volume=4*np.pi*(length/2)**3/3  #assume spherical
    print(f"{generation:5}{nocells:21}\t{length:.7f}\t{volume:0.10f}")
```

The starting lines of the output are:

generation	number of cells	length (cm)	volume (cm^3)
1	2	0.0012304	0.0000000010
2	4	0.0015502	0.0000000020
3	8	0.0019531	0.0000000039
4	16	0.0024608	0.0000000078
5	32	0.0031004	0.0000000156
6	64	0.0039062	0.0000000312
7	128	0.0049216	0.0000000624
8	256	0.0062008	0.0000001248
9	512	0.0078125	0.0000002497
10	1024	0.0098431	0.0000004993

and the last few lines, relevant for this discussion, are here:

25	33554432	0.3149803	0.0163624617
26	67108864	0.3968503	0.0327249235
27	134217728	0.5000000	0.0654498469
28	268435456	0.6299605	0.1308996939
29	536870912	0.7937005	0.2617993878
30	1073741824	1.0000000	0.5235987756

The number of cells in the 30th generation (doubling) is a bit over one billion, and the size is 1cm (so, the claim in the paper "*…30 doublings to reach a size of 1 cm, when it will contain 10^9 cells…" is correct)*. Note that from one generation (one doubling) to the next, the volume doubles, but the length does not - the length increases by a factor of $\sqrt[3]{2}$.

So, for the length to double from 0.5cm to 1 cm, it takes three doublings, not one (thus the claim "*…the lead time gained by mammography over clinical breast examination would be of the order of only one doubling*" is incorrect).

The error in the paper is quite serious; some health authorities and administrators tried to argue, based on the claim about the lead time of one doubling, that a mammography (which is expensive) might not provide a benefit of early cancer detection compared to a less costly clinical breast examination.

Conclusion

In this paper we presented examples which illustrate how certain numeracy tasks can be solved using coding. In the conceptualization of a numeracy task based on Wolfram's computational model define – abstract – compute – interpret and communicate, coding is part of *compute* stage. Since they appear in authentic, real life contexts (rather than being artificial as many word problems in math textbooks are), numeracy tasks, besides having the value on their own, could enrich teaching mathematics. As well, the non-traditional approaches and intuitive reasoning that are natural components of a numeracy task should establish their place in solving problems in mathematics.

References

Burazin A., Gula, T. & Lovric, M. (2023). Numeracy course “Numbers for Life” at McMaster University: Research into the effectiveness of the course instruction, and quality numeracy tasks. In: TBA (Eds.). *Proceedings of the Adults Learning Mathematics (ALM 30) Conference, 3-5 July 2023, Limerick, Ireland*. (in preparation)

Geiger, V., Goos, M., & Dole, S. (2011). Teacher Professional Learning in Numeracy: Trajectories Through a Model for Numeracy in the 21St Century. In J. Clark, et al. (Eds.), *Mathematics: Traditions and [New] Practices. Proceedings of the 34th annual conference of the Mathematics Education Research Group of Australasia and the Australian Association of Mathematics Teachers*. Adelaide: AAMT and MERGA.

Geiger, V., Goos, M., & Forgasz, H. (2015). A rich interpretation of numeracy for the 21st century: A survey of the state of the field. *ZDM*, 47, 531-548.

Gula, T. & Lovric, M. (2023). Numeracy tasks: provoking transfer between concrete and abstract thinking spaces. Submitted to *Numeracy*.

Lovric, M. (2012). *Calculus for the Life Sciences: Modelling the Dynamics of Life. Probability and Statistics.* Toronto: Nelson.

Mittra I., Baum, N., Thornton, H., & Houghton H. (2000). Is clinical breast examination an acceptable alternative to mammographic screening? *BMJ* 2000; 321: 1071-1073. doi: 10.1136/bmj.321.7268.1071.

Steen, L. A. (Ed.). (1997). *Why numbers count: quantitative literacy for tomorrow's America.* New York: College Entrance Examination Board.

Stewart J., Clegg, D. K. & Watson, S. (2021). *Calculus: Early Transcendentals, 9th Edition.* Brooks Cole.

Wolfram, C. (2016). Anchoring Computational Thinking in today’s curriculum. Available at: https://www.conradwolfram.com/writings/anchoring-computational-thinking-in-todays-curriculum. Accessed 19 July 2023. See also https://www.computationalthinking.org