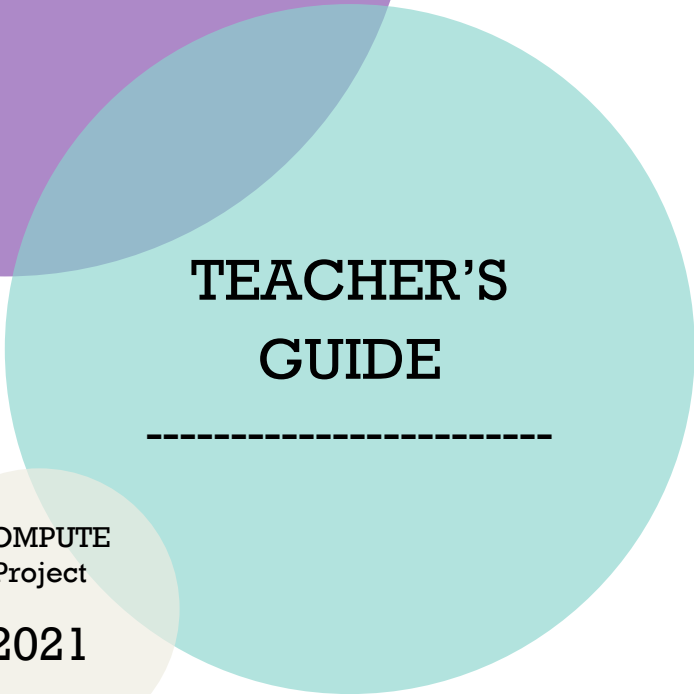




GUMMY BEARS

**For Computational
Thinking**



**TEACHER'S
GUIDE**



**COMPUTE
Project**

2021

GUMMY BEARS

FOR COMPUTATIONAL
THINKING^[1]

PROBLEM + SOLUTION ENGLISH & PSEUDOCODE

- The Problem
- The Solution
- The Solution in Pseudocode

The Problem

Your friend needs your help. He has three types of gummy bears:

- Mini bears (1.21 g each).
- Regular bears (2.31 g each).
- One super bear (2668 g).

He comes to you and says, “I have a certain quantity of *mini* bears. If I eat them all, how many more *regular* bears would I need to eat to match the super bear?”

The Solution

Let's list *all* the steps that you would need to take to help your friend.

Plain English

Fill in the Blanks

1. First you need to ask your friend, “How many **mini bears** do you have?”
2. Then your friend needs to answer. For this example, let's imagine your friend has forty mini bears, so they tell you: “I have 40 mini bears.”
3. Now you have to do some calculations. There are several methods to do this, but let's use the arithmetical method.

(Be sure to include *all* steps of your calculations.)

3.1. Weight of 40 mini bears = **$40 * 1.21 \text{ g.} = 48.4 \text{ g.}$**

3.2. Weight of regular bears needed = **$2668 \text{ g.} - 48.4 \text{ g.} = 2619.6 \text{ g.}$**

3.3. Regular bears needed = $2619.6 \text{ g.} / 2.31 \text{ g.} = 1134.026$.

4. Finally, now that you have the result of your calculations, you need to convey them to your friend: “You would need **1134.026 regular bears** to match the super bear!”

Generalized Solution

Now let’s generalize these steps such that you are able to use them as a formula for *any* ‘x’ number of initial mini bears.

1. First you need to ask your friend, “How many **mini bears** do you have?”
2. Then your friend needs to answer: “I have ‘x’ mini bears.”
3. Now you have to do some calculations. There are several methods to do this, but let’s use the arithmetical method.

(Be sure to include *all* steps of your calculations.)

3.1. Weight of ‘x’ mini bears = $1.21x \text{ g.}$

3.2. Weight of regular bears needed = $2668 \text{ g.} - 1.21x \text{ g.}$

3.3. Regular bears needed = $(2668 \text{ g.} - 1.21x \text{ g.}) / (2.31 \text{ g.})$

4. Finally, now that you have the result of your calculations, you need to convey them to your friend: “You would need $(2668 \text{ g.} - 1.21x \text{ g.}) / (2.31 \text{ g.})$ more regular bears to match the super bear!”

Pseudocode

Now let's start translating our solution into a language that is *similar* to the languages that computers understand. On the limbo between plain English and programming languages, there is pseudocode. For this exercise, we will use a variation of the conventional pseudocode.

Variables

A computer doesn't know which variables you want it to use, unless you *create* these variables *in your code*. Thus, let's start by creating the variables that the computer will need to solve the Gummy Bears problem.

Refer back to your generalized solution. At first glance, it may seem that you have only one variable ('x'), but look closer. The weight of 'x' mini bears depends on which value 'x' takes. What about the weight of the total of regular bears needed? And the total of regular bears needed?

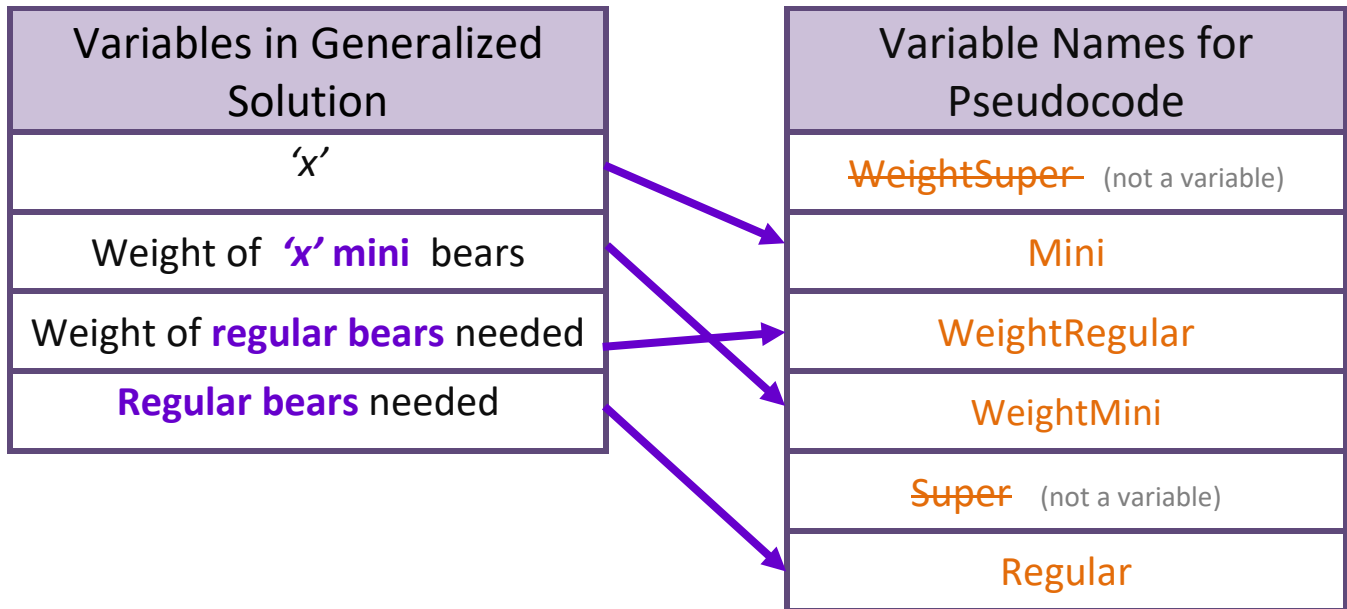
These are all variables!

In coding it is always best to give variables a distinct name that alludes to the variable's role. (e.g. "Weight of Regular Bears"). However, spaces between words are normally not allowed, and you also want to keep the name short. Thus, "Weight of Regular Bears" may become something like "Weight_Regular_Bears" or "Weight_Regular", "WRegular", etc.

Matching Pairs Game + Fill in the Blanks

Fill in the blanks of the first column. Then match each variable of the generalized solution to their most fitting name for pseudocode.

Note: Not all variable names on the right column will be used.



Now that we have our variables, let's work on the rest of the pseudocode.

Matching Pairs Game + Fill in the Blanks

Match each action of the generalized solution in plain English (left purple column) with its pseudocode translation (right purple column). Use the options on the green chart to fill in all the blanks.

Note: If possible, keep the color-coding of the options on the green chart (e.g. "answer" is turquoise; "Mini" is orange.)

TEACHERS

It is okay if students cannot keep the color-coding of the given words/phrases/symbols (e.g. they may not have color pencils at hand, or they may be filling out the worksheet digitally). However, when working on the Scratch part, make them refer to the next exercise to check what color each word/phrase/symbol is supposed to be. ("answer" is turquoise; "Mini" is orange, etc.)

Note: Some of these options will not be used, and some may be used more than once.

Choose from the following options to fill in the blanks:

WeightMini	2668	TotalSuper	"more regular bears to match the super bear!"
Weight of regular bears needed	Weight of mini bears	WeightRegular	"You would need"
answer	"How many mini bears do you have?"	Regular	2.31
40	Mini	1.21	WeightSuper

Generalized Solution (For computers)

Program asks how many mini bears the user has and waits for the answer.

User enters their answer to inform how many mini bears they have.
(Number of mini bears = User's answer)

Weight of mini bears = Number of mini bears * 1.21 g.

Weight of regular bears needed = 2668 g – Weight of mini bears

Number of regular bears needed = Weight of regular bears needed / 2.31

Program says how many regular bears the user would need to match the super bear.

Pseudocode

SET Regular TO (WeightRegular / 2.31)

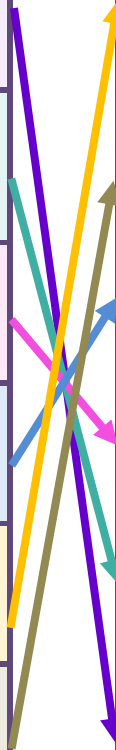
SAY "You would need" Regular "more regular bears to match the super bear!"

SET WeightRegular TO (2668 – WeightMini)

SET WeightMini TO (Mini * 1.21)

SET Mini TO answer

ASK "How many mini bears do you have?" AND WAIT



Hint: If the pseudocode name for the total number of regular bears needed were 'z', then the pseudocode line for the last step would be: SAY "You would need" z "more regular bears to match the super bear!"

Reorder the Pseudocode

Now that you have matched each step with its pseudocode translation, write the whole pseudocode below in the right order.

Remember to write each step in a different line. If possible, keep the color-coding of the pseudocode lines above. (E.g. "ASK" is turquoise; "SET" is orange; operators are green.)

TEACHERS

It is okay if students cannot keep the color-coding of the given words/phrases/symbols (e.g. they may not have color pencils at hand, or they may be filling out the worksheet digitally). However, when working on the Scratch part, make them refer to the previous exercise to check what color each word/phrase/symbol is supposed to be.

ASK "How many mini bears do you have?" AND WAIT

SET Mini TO answer

SET WeightMini TO (Mini * 1.21)

SET WeightRegular TO (2668 – WeightMini)

SET Regular TO (WeightRegular / 2.31)

SAY "You would need" Regular "more regular bears to match the super bear!"

TEACHERS

Revise the students' pseudocode. It is important that they have it right before they start coding on Scratch.

GUMMY BEARS

FOR COMPUTATIONAL
THINKING^[1]

Tab 2

SCRATCH

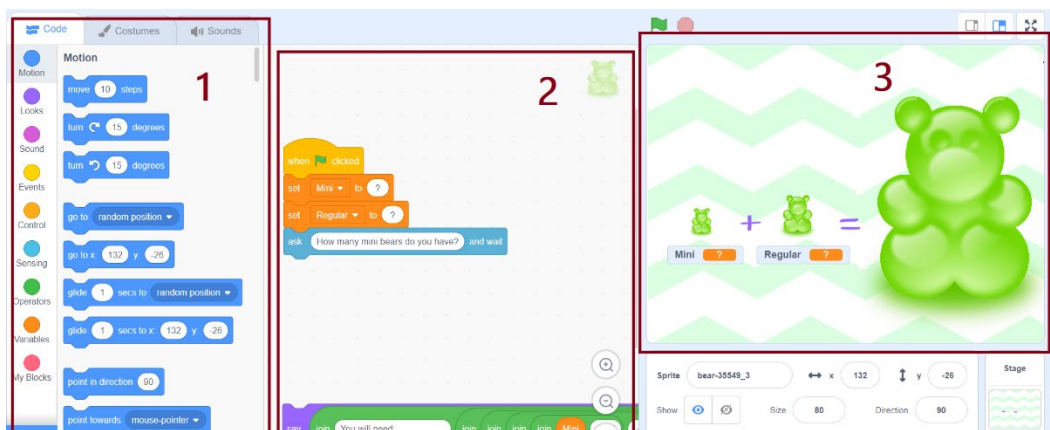
- Getting started
- Coding on Scratch

Getting Started

1. Open Scratch (<https://scratch.mit.edu/>) and create a free account.
2. Follow this link: <https://scratch.mit.edu/projects/556828524/editor> to open the Gummy Bears project template. **Do not edit this template!**
3. **Create a copy** of this template by clicking on File > Remix. Work on this copy (“remix”) from now on.

Coding on Scratch

1. Let’s start by familiarizing ourselves with Scratch:



- 1.1 The first section (left) contains all the possible coding blocks you can use on Scratch. The menu on the far left shows the different categories of coding blocks (“Motion”, “Looks”, “Sound”, etc.)
 - 1.2 The second section (middle) is “your workspace”. This is where you write your code by dragging your chosen coding blocks from the first section.

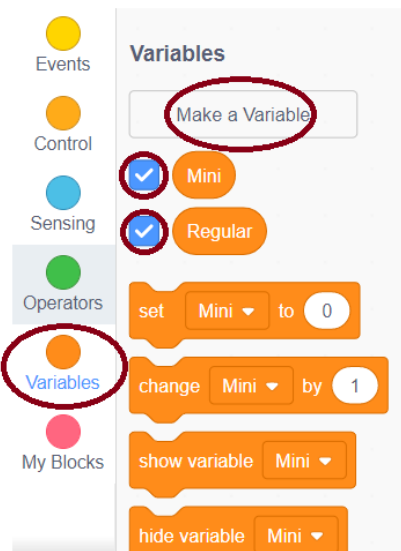
Right now, you should see four lines of code we have pre-written for you. Do you recognize the last line? (Ignore all those “joins”.) What about the first three lines—can you guess what they do?

1.3 The third section (right) will show the end result of your code.

2. Create your variables.

We have created the variables “Mini” and “Regular” for you. Refer to the previous exercises and identify which variables you still need to create.

You can create a variable by clicking on *Variables > Make a Variable*. Create the variables for all sprints. Next, uncheck their checkboxes, such that only “Mini” and “Regular” remain checked:



3. Now let's code!

Look at your workspace. We have helped you with the first three lines of code (which are not in your pseudocode) and the last line of code (which is also the last line in your pseudocode). Your job is to fill in the code in between.

To do this, look at all the possible coding blocks listed on the left side on Scratch, and then choose those that most resemble your pseudocode. Work line by line. Part by part. The Scratch translation of your pseudocode should go right after the first three lines of code we have helped you with.

Hint: Remember that the pseudocode lines in our previous exercises were color-coded. (e.g. “ASK” was turquoise, just as the “Sensing” category on Scratch!)

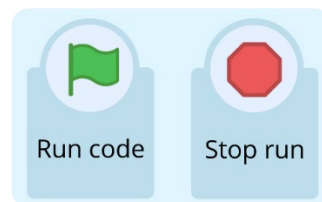
Teachers

The color-coding from the previous exercises is meant to guide students through the categories on Scratch.

All variables (“Mini”, “WeightMini”, “Regular”, “WeightRegular”) were orange, just as the “Variables” category on Scratch. Operators were green, mirroring the green “Operators” category on Scratch.

Some useful tips:

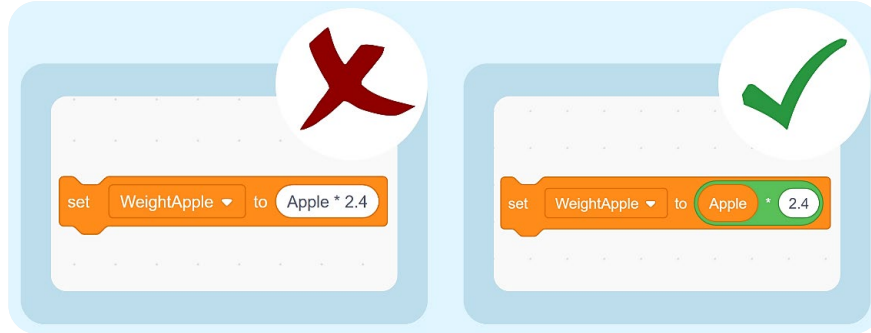
- To run and stop the code:



- To add new coding blocks:

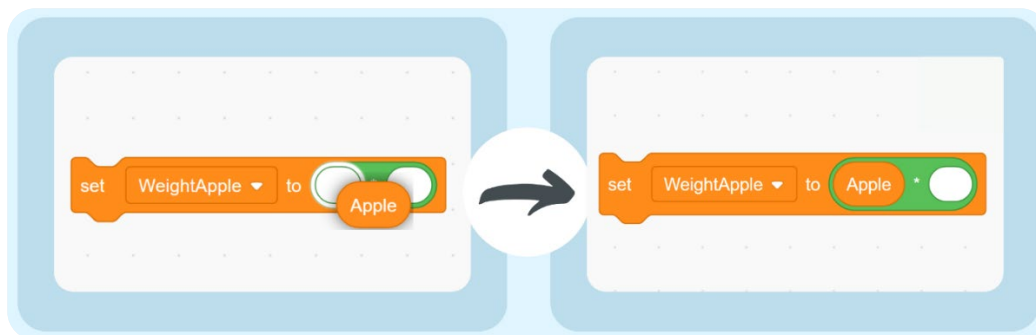


- To use variables and operators:

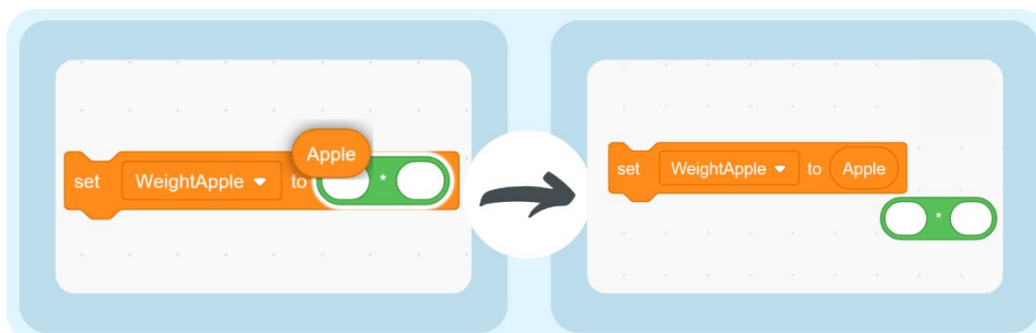


- To embed blocks:

- Drag a block towards the container block. Check the highlighted area before dropping it.
- The highlighted area tells you where the block will be placed:



The new block will be inserted into the first blank space.



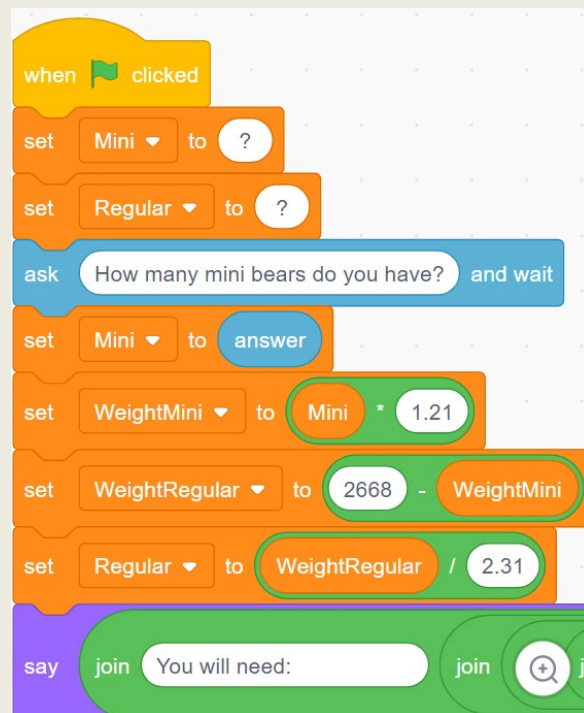
The whole green block will be replaced

Once you finish your code on Scratch, test it! Assume the role of “user” and enter any random number of mini bears. Is the program calculating correctly? Revise your code if necessary; test it again. Revise it until you get it right.

Happy coding!

Teachers

This is what the finished Scratch code looks like:



(The last line is incomplete here due to its length. This line was given to students, however, and they should not modify it.)

You can find the fully functional Gummy Bears Scratch code here:

<https://scratch.mit.edu/projects/553229937/editor/>

If clicking on the link above does not work, try copying it directly to the address bar.

GUMMY BEARS

FOR COMPUTATIONAL
THINKING^[1]

PYTHON

- Writing in Python
- Jupyter Notebook

Python

Matching Pairs Game + Fill in the Blanks

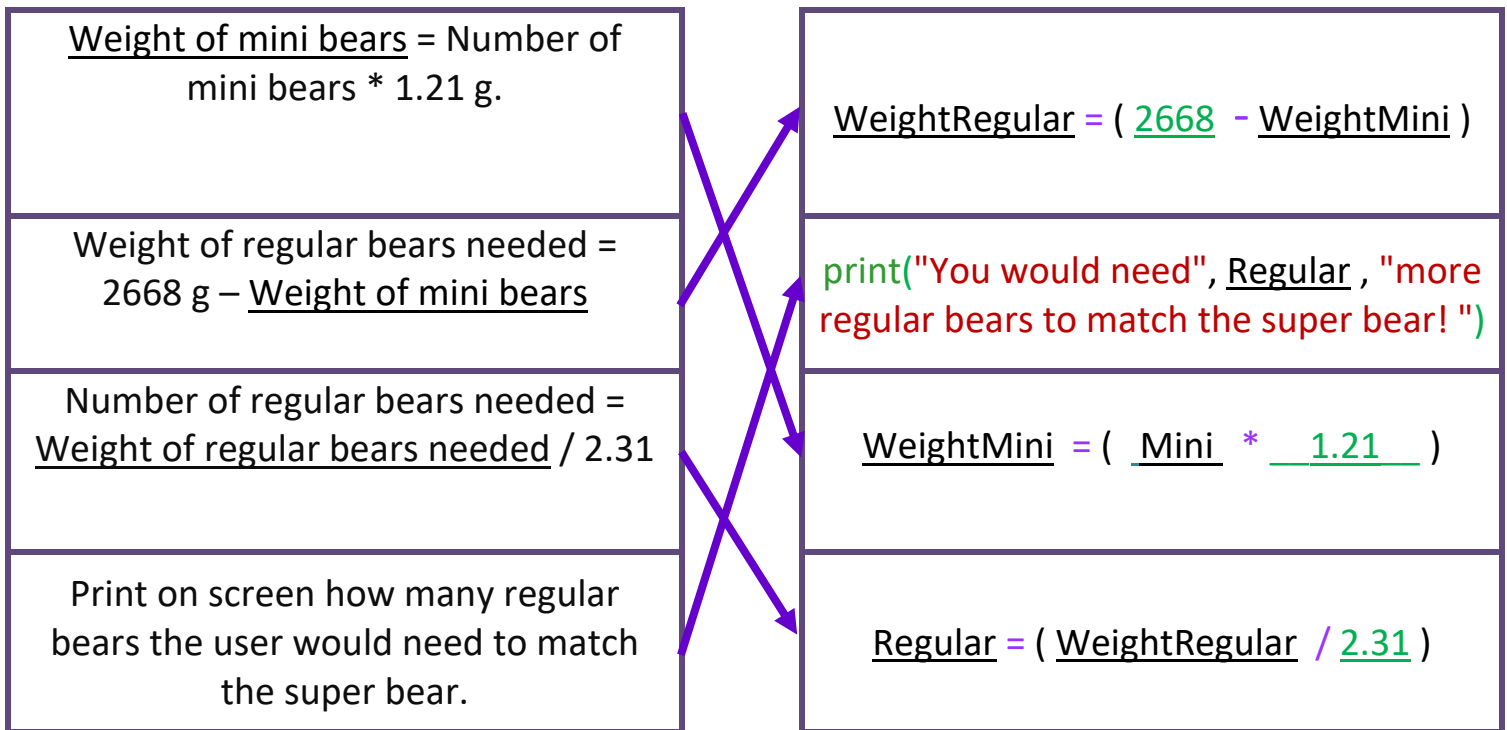
Match each action of the generalized solution in plain English (left purple column) with its Python equivalent (right purple column). Use the options on the green chart to fill in all the blanks.

Note: Some of these options will not be used, and some may be used more than once.

Choose from the following options to fill in the blanks:			
WeightMini	2668	TotalSuper	1.21
Weight of regular bears needed	Weight of mini bears	WeightRegular	WeightSuper
40	Mini	Regular	2.31

Note: Not all options on the right column will be used.

Generalized Solution (Adapted for Python)	Python
Ask how many mini bears the user has and store the user's input in a variable called "mini". Additionally, specify that the user's answer will be an integer number.	Regular = int(input("How many mini bears do you have?"))
	Mini = int(input("How many mini bears do you have?"))
	Mini = (input("How many mini bears do you have?"))



Hint: The program does not know whether the user's input will be text (which cannot be used for mathematical operations) or numbers (which can be used for mathematical operations). Thus, you need to specify what type of input (text, integers, decimals) the program should expect.

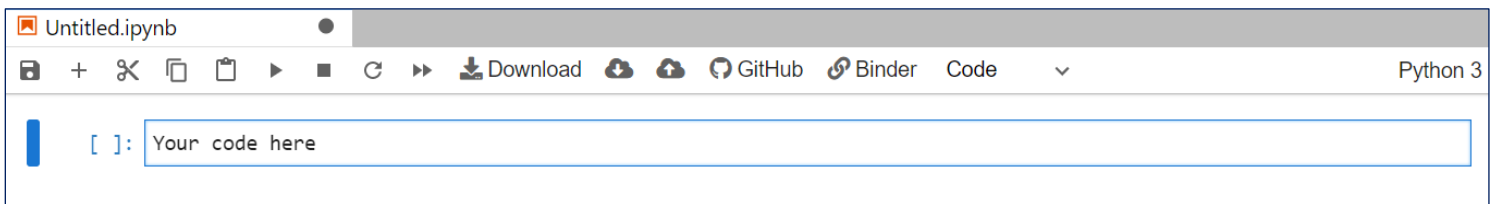
Reorder the Code

Now that you have matched each step with its Python equivalent, write the whole Python code below in the right order.

```
Mini = int(input("How many mini bears do you have?"))  
WeightMini = ( Mini * 1.21 )  
WeightRegular = ( 2668 - WeightMini )  
Regular = ( WeightRegular / 2.31 )  
print("You would need", Regular, "more regular bears to match the super bear!")
```

Run Your Code

1. Open Jupyter Notebook: <https://jupyter.org/try>
2. Open JupyterLab and wait a few seconds for it to load.
3. Click on “Python 3” under “Notebook”.
4. Type the Python code you wrote above into your workspace.



5. Click on the Play button to run your code and test it.
6. Congratulations! You just coded the Gummy Bears problem in Python.

Teachers

Here are some common typing errors in case you need to troubleshoot:

- **Forgetting** a comma, parenthesis, or some other **punctuation mark**.
- **Order of characters**. (e.g. Misplacing a comma inside of the quotation marks when it should be outside, or vice versa.)
- **Mismatched quotation marks**. (e.g. Combining single with double quotation marks, or straight with curved quotation marks.)
- **Inconsistent capitalization of variables**. (e.g. Naming a variable “Mini” but then trying to use it as “mini”.)

Note: Remind students that these are, not silly, but rather common mistakes even within the coding community. Even the most proficient coder can spend hours debugging his faulty program only to find that he had forgotten a comma!

References

- [1] *Module Adapted from* D. Meyer, “Super Bear,” Three Act Math, dy-dan – Dan Meyer., Oct. 30, 2011. [Online]. Available:
https://docs.google.com/spreadsheets/u/0/d/1jXSt_CoDzyDFeJimZxnhgwOVsWkTQEsfqouLWNNC6Z4/pub?output=html

