

Lab 02 Generating Random Numbers and Using Variables

Not to be redistributed
to Course Hero or any
other public websites

General Information

What you should do

- You should follow the instructions step by step especially if this is your first programming course.
- You should try to come up with the code yourself as much as possible. Do not be afraid of making mistakes, since debugging (finding out where your code goes wrong and fixing it) is part of the learning process.
- We do not give out model programs to the exercises. There can be multiple ways to write the code that solves the same problem. It is important that you build up the program logic yourself instead of merely looking at some code that you do not understand. At any time if you are lost or if you have any questions, feel free to ask the instructor, tutor, or teaching assistant and we will be very happy to help you.

Self-Discovery

- Most lab tasks are designed to be relatively simple such that you can take the time to think about the related underlying concepts. Besides, we also encourage you to discover things on your own which may not be specified in the tasks.

Task 1.1 Basics of Random Numbers

A computer program is often required to generate random numbers such as giving players a random set of cards in a game or simulating mathematical models for scientific experiments. Can you come up with some real-life examples of generating randomness such that the outcome is provided in random from a list of possible choices and no computer is used?

Note that the random numbers generated by computer are often not true random numbers but are called **pseudo-random numbers** instead. *True random numbers are non-deterministic*, meaning that there is no way for you to determine what number comes next even if you have the entire history of all the numbers generated in the past. On the other hand, pseudo-random numbers are generated by computer algorithms such that it is like showing numbers successively from a given list that repeats itself (thus you can determine the pattern if you wait long enough). Although the list may be very long for you to notice the pattern practically, it is still possible to determine the numbers that will show up if you know the algorithm thus *pseudo-random numbers are deterministic*.

Start the Chrome browser on your notebook and go to the webpage <https://scratch.mit.edu/> and sign in with the username and password of your account that you created in Lab 01.

In Scratch, a random number is generated by the following block:

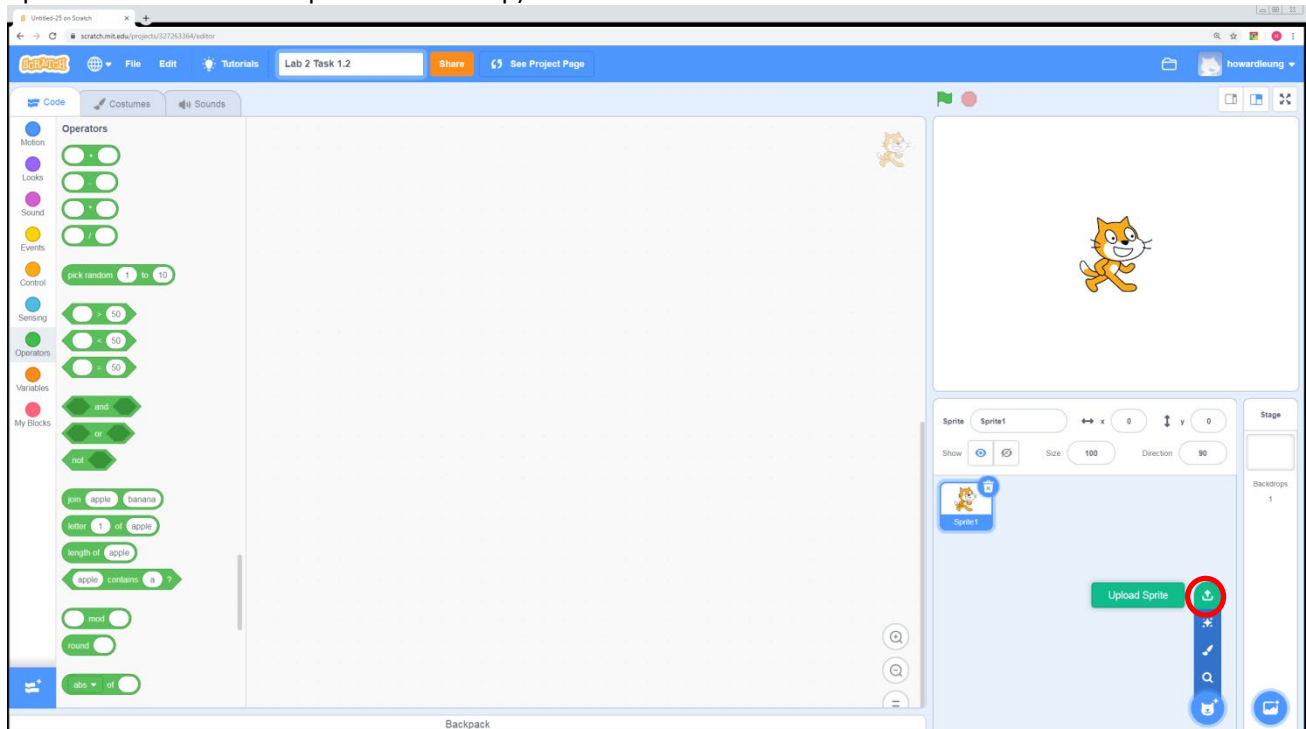


If you click the above block, then you will see that a random integer in the range of 1 to 10 (including both 1 and 10) is generated each time.

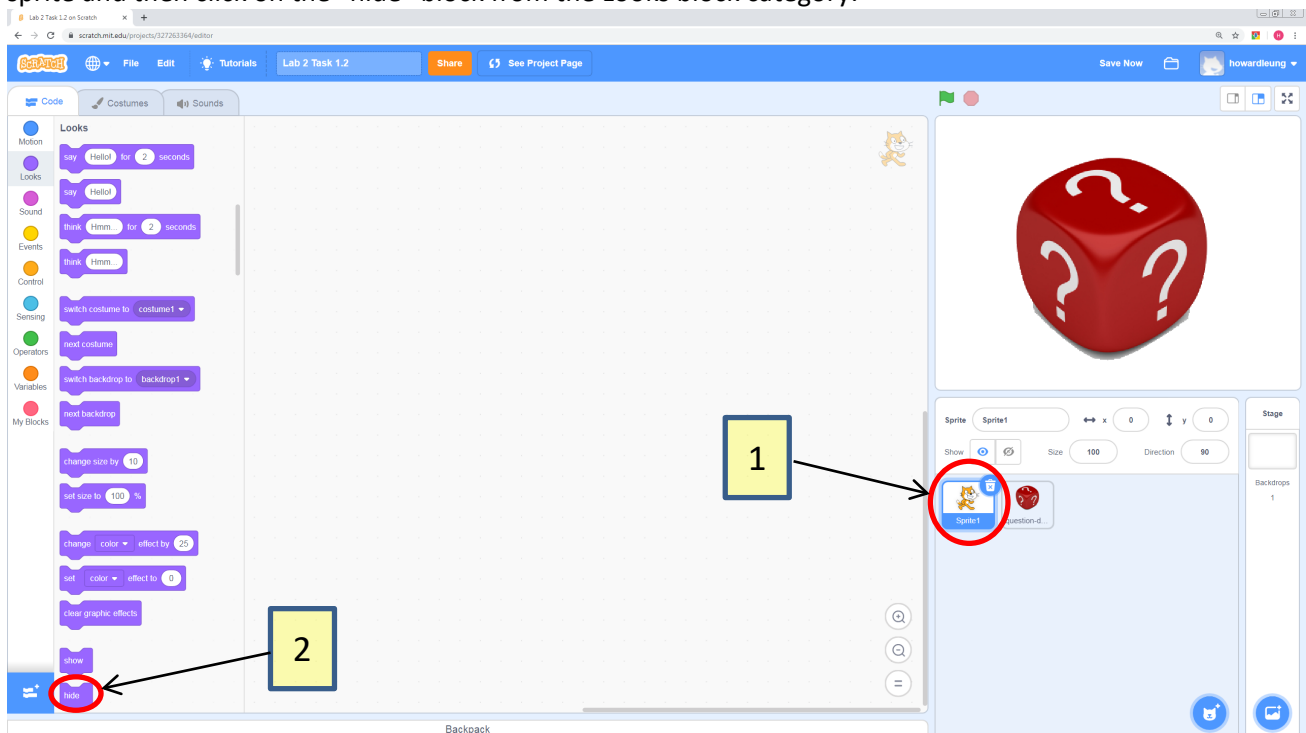
Task 1.2 Let Computer Throw Dice

In this task you will make a Scratch program to simulate dice throwing, i.e., it will generate a **discrete** random integer number from 1 to 6 when you click on the picture of dice. Download the sprite file `question-dice.sprite2` from the Canvas course page which contains such picture to the notebook.

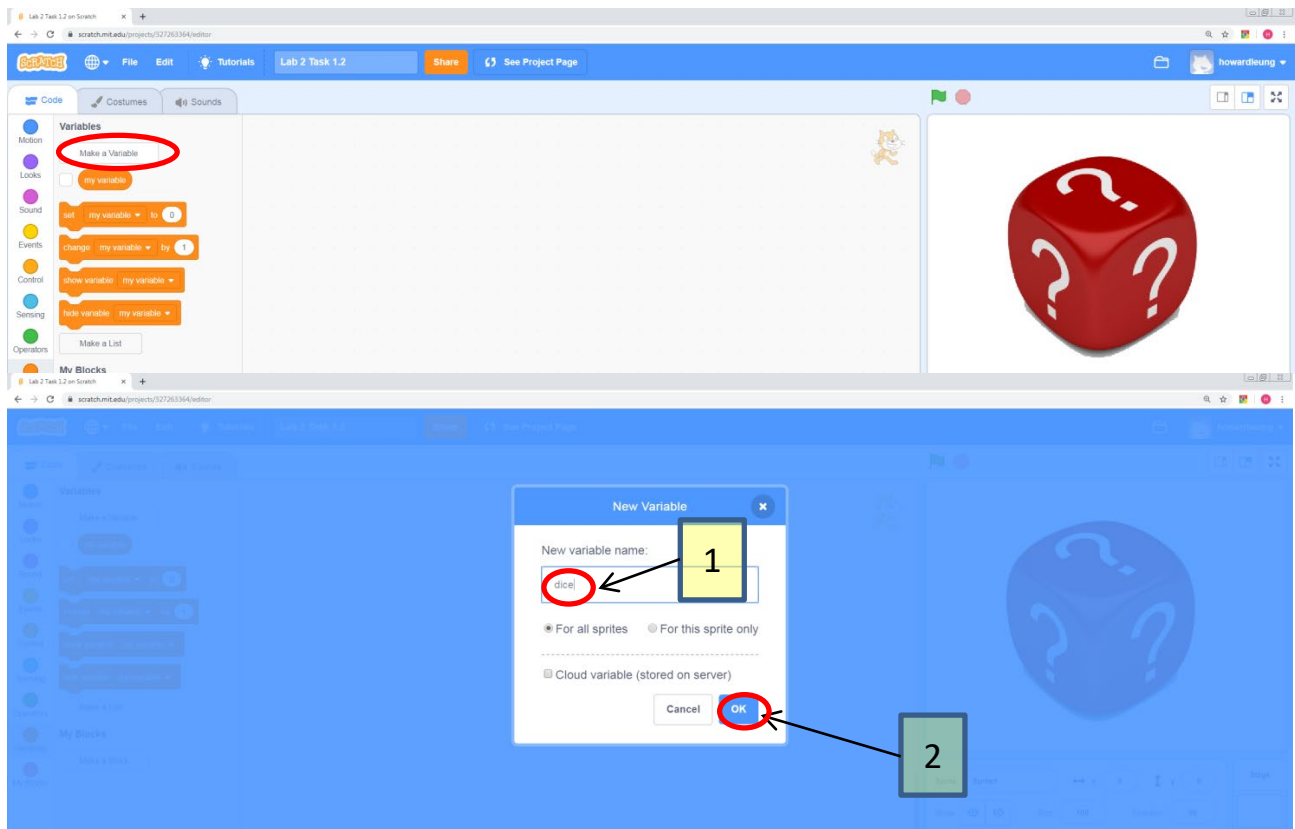
In Scratch, create a new project and put down Lab 2 Task 1.2 as the project name. Upload the sprite file `question-dice.sprite2` from the notebook to this new project by clicking on the button on the bottom right as shown in the following screenshot (hover the mouse over the blue cat icon to expand the options and choose the upload icon on top):



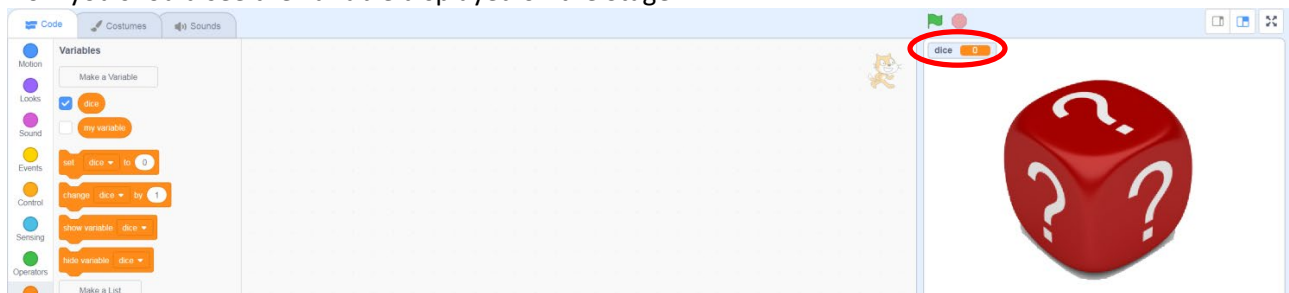
For this task we do not need the Scratch Cat so we can hide it. One way to hide a sprite is to click on that sprite and then click on the “hide” block from the Looks block category:



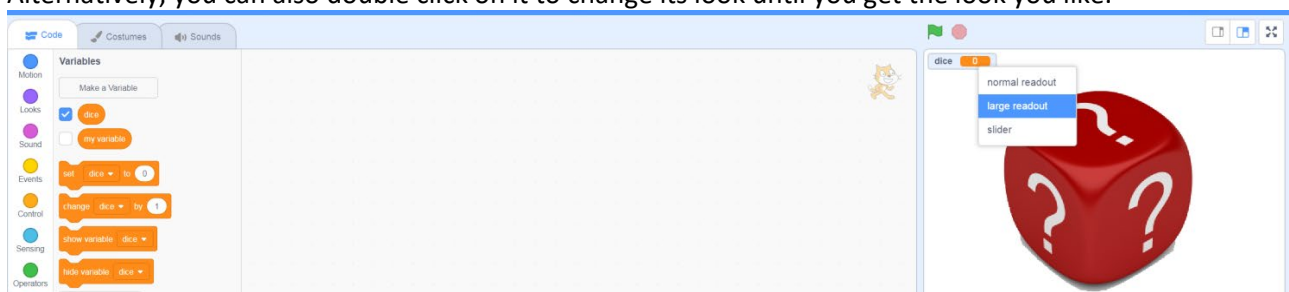
Now we want the program to display a number ranging from 1 to 6. For this, we create a variable to store this number. Click on the “Make a Variable” block from the Variables block category and put down *dice* as the variable name:



Now you should see the variable displayed on the Stage.



You can change the look of the variable. Now right click on the variable dice and choose large readout. Alternatively, you can also double click on it to change its look until you get the look you like.



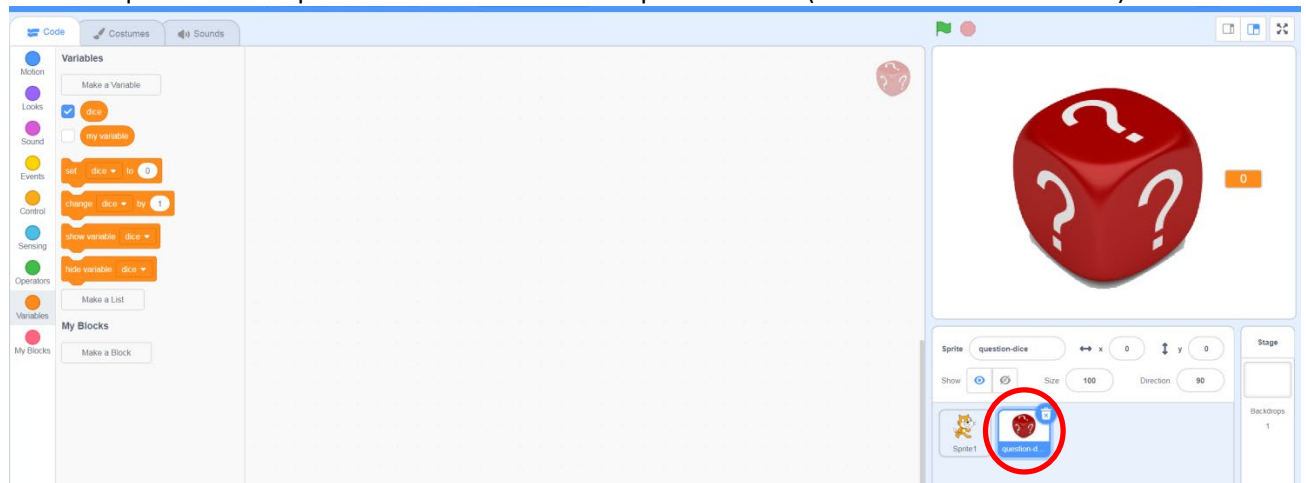
Drag this variable dice and move it next to the sprite.



Now we want to add the code to display a random number from 1 to 6 when the dice sprite is clicked.

To do:

- 1) Click the question-dice sprite to make sure that this sprite is active (instead of the Scratch cat)

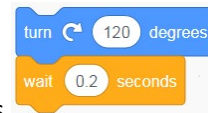


- 2) Add the block **when this sprite clicked** from the Events block category

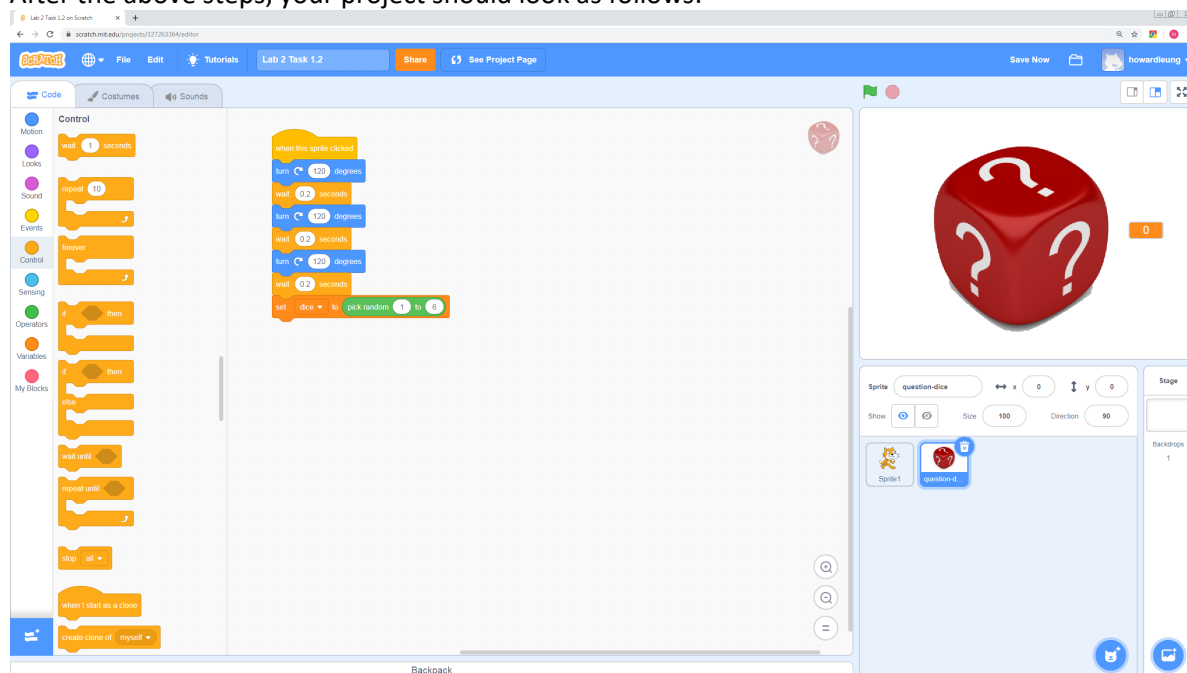
- 3) Add the block **set dice to 0** from the Variables block category

- 4) Add the block **pick random 1 to 6** inside the white square initially contains the value 0 from the previous block. Note that you need to change the numbers so that it is from 1 to 6 (instead of from 1 to 10).

- 5) To create the visual effect of the dice throwing, add the blocks **turn 120 degrees** and **wait 0.2 seconds** 3 times before the setting the value of the variable from the previous step.



After the above steps, your project should look as follows:



Now you can try your code by clicking on the dice sprite and see a new random number displayed after the dice movement.

Task 1.3 Generate Random Numbers from a Set of Non-Consecutive Integers

- 1) What are the possible values of x as a result of the following block?



- 2) How do you modify from the above block such that $x \in \{-1, 1\}$, i.e., the value of x is either -1 or 1?
- 3) How can you generate a random integer $x \in \{0, 1, 2\}$ but instead of equal probabilities, the probabilities of getting 0,1,2 are 0.25,0.5,0.25 respectively?

- 4) What are the possible values of y as a result of the following block?



- 5) How do you modify from the above block such that $y \in \{1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$?
- 6) What are the possible values of p as a result of the following block?



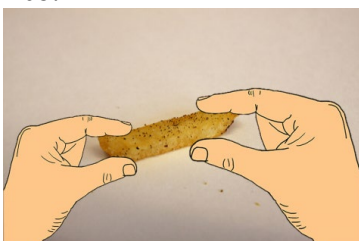
- 7) Does p from the above block have the same set of possible outcomes as q from the following block? If not, what are the differences?








Task 1.4 Random Floating-Point Numbers

In Task 1.2, your code generates random integer numbers. In this task, we would like the code to generate random floating-point numbers, e.g., 0.1, 0.234, 0.987654321, etc.

Consider the scenario in which 2 kids would like to get to the last piece of French fry which measures 1 inch long. Each kid is holding on one end of the fry and would eventually break the fry in a random position. Can you modify the code from Task 1.2 to simulate the random length of the fry segment obtained by one of the kids?



First we should determine the range of the random number to be generated. The original fry measures 1 inch so the kid may obtain at most 1 inch (the entire fry) if he is lucky. Alternatively, he may obtain 0 inch (nothing) if he is really unlucky. So the range of the random number is from 0 to 1. Besides, he may just get a fraction of the fry so he may get something like 0.5 inch (half), 0.25 inch (25%), 0.1234567 inch, etc. As a result, in the code, you need to generate a random **continuous** floating-point number (a number with decimal places) that ranges from 0 to 1. Note that if you represent all floating-point numbers on a number line, these numbers change continuously and given a certain number, it is not possible for you to tell what is the next number (as opposed to the case with integers where you can tell what is the next number).

If you change the numbers in the block  from "1 to 6" to "0 to 1", would it be able to generate the required random floating-point number? If you change these parameters to "0.0 to 1", would it be able to generate the required floating-point number? Note that as long as you put down a floating-point number in any of the 2 input parameters, a random floating-point number will be generated, with the range bounded by the 2 input numbers (inclusive). For example, while the block  gives you possible value of 1, 2 or 3, any of these blocks ,  or  would give you a floating-point random number x where $1 \leq x \leq 3$.

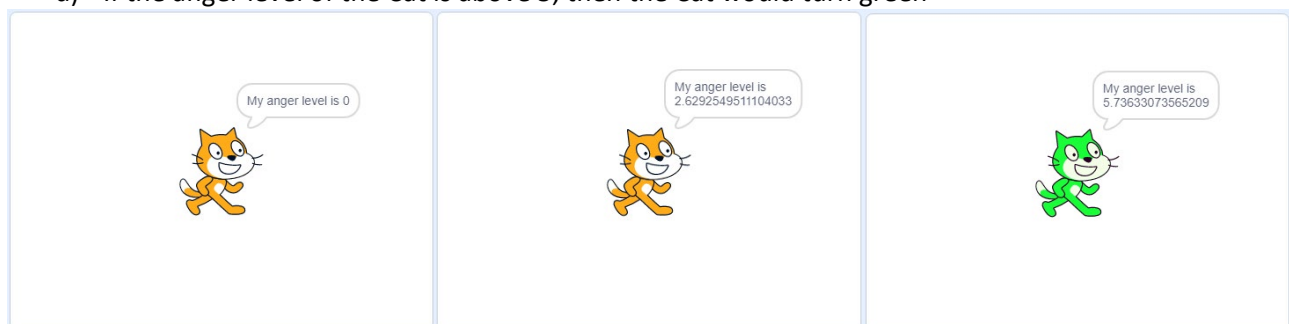
Exercise 1.4

- Generate a random floating-point number x such that $-3 \leq x \leq -1$ or $1 \leq x \leq 3$, i.e., $x \in [-3, -1] \cup [1, 3]$
Hint: refer to Task 1.3 step 2)
- Generate a random floating-point number x such that $0 \leq x \leq 2$ or $4 \leq x \leq 6$, i.e., $x \in [0, 2] \cup [4, 6]$
Hint: refer to a)

Task 1.5 Angry Cat

In this task, we will create the code with the following basic specifications:

- When the program starts, the anger level of the Scratch Cat (Cat) will be set to 0
- Each time when the user clicks the Cat, a random floating-point number in the range of 0.5 to 1 will be added to the anger level of the Cat
- The Cat will always say what is its current anger level
- If the anger level of the Cat is above 5, then the Cat would turn green



Create a new project and put down Lab 2 Task 1.5 as the project name. You can first try to write the program yourself according to the above specifications. If you are not able to do it, you can refer to the following step-by-step guide:


- Create 2 variables *anger* and x . The first variable *anger* is used to store the anger level of the Cat and x is used to store the random floating-point number to be added to the anger level of the Cat.

- 2) When the program starts, the variable *anger* needs to be initialized to 0. The color of the Cat should be reset to the original color (just in case previously it was set to other color). The Cat will say what is its anger level. This step can be accomplished by the following block:



- 3) The program also detects if the sprite Cat is clicked, then it will generate a random floating-point number x , which will then be added to the variable *anger*. The Cat will say what is its anger level. The program will further check if *anger* exceeds 5, then it will change the color of the cat to green. This step can be accomplished by the following block:



Note that the block  would take the value of x and then add it to *anger*, meaning that the value of *anger* will be increased by x . Another way of performing this operation is the following block:



Now try your program and click on the Cat until it turns green!

Exercise 1.5: Modify your program so that it satisfies the following additional specifications:

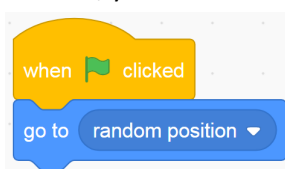
- e) Each time when the user clicks the space key, a random floating-point number in the range of 0.5 to 1 will be subtracted from the anger level of the Cat. Hint: Use the following block:



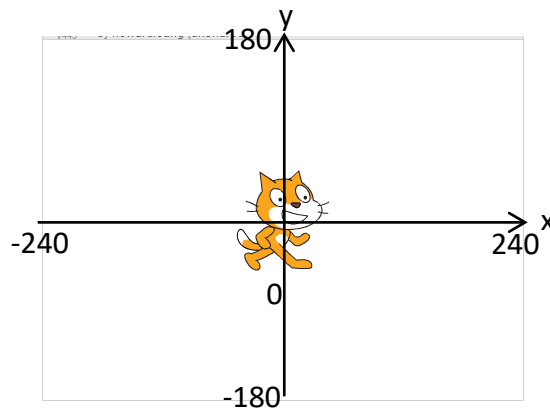
- f) The Cat will always say what is its current anger level
 g) If the anger level of the Cat is below 5, then the Cat would back to the original color (same color as when the program starts)

Task 1.6 Random Sprite Position

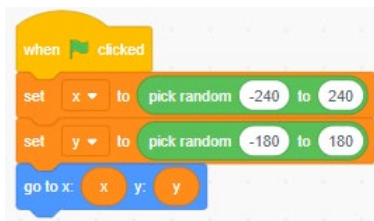
In Lab 1, you have used the following block to move the Cat and the Mouse sprites to a random position:



The x coordinate of the stage area ranges from -240 (left) to 240 (right) and the y coordinate ranges from -180 (bottom) to 180 (top). The center of the stage has the coordinates (0,0). The coordinate system of the stage area is illustrated below:

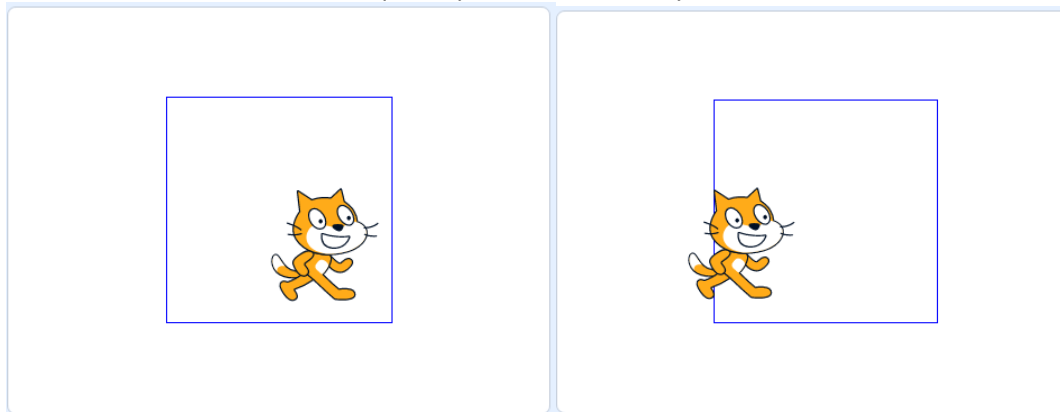


As a result, the above block would be equivalent to the following block for setting the sprite in a random position when the program starts:




Exercise 1.6: Write a program so that it satisfies the following specifications when the program starts:

- All the pen marks on the stage are cleared
- A square of size 200 centered at (0,0) is drawn using blocks from the Pen category
- The Cat moves to a random position with its center inside the drawn square. Note that when the center of the Cat is inside the square, part of the Cat may still fall outside it.

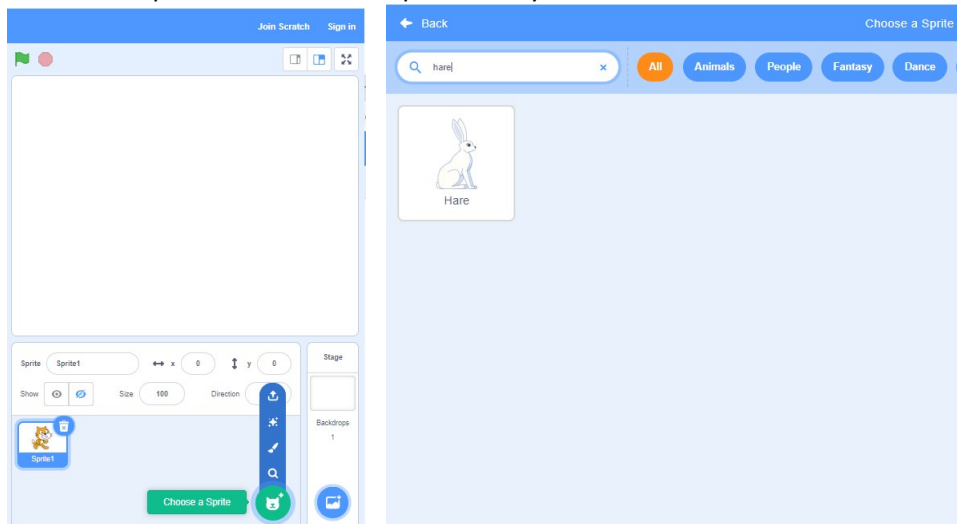


Task 1.7 The Tortoise and the Hare

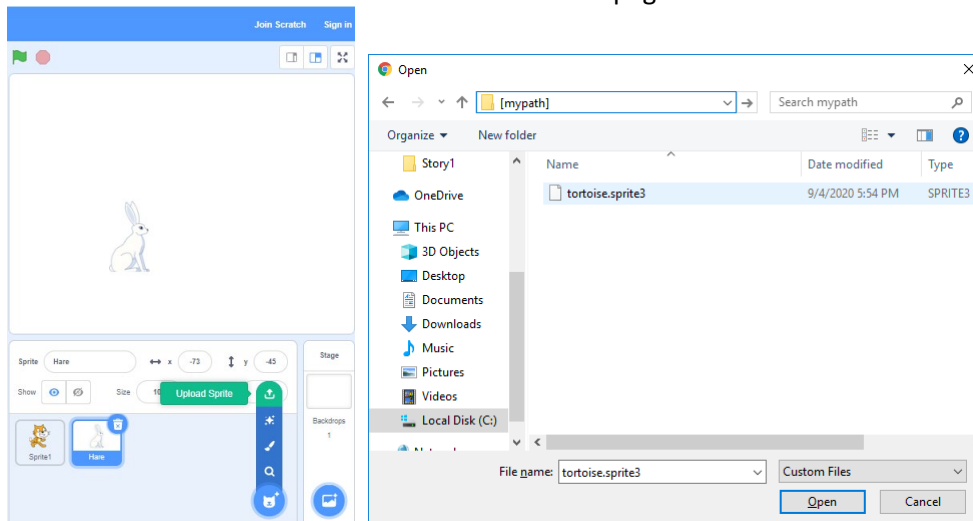
All of you must have heard the Aesop's fable about the tortoise and the hare: there is a race between the tortoise and the hare. The hare runs very fast at the beginning but takes a nap after a while. The tortoise keeps moving slowly but does not stop. The tortoise finally wins the race. We will recreate this race in this task.

Create a new project by selecting File > New from the menu. We do not need the Scratch Cat sprite so you can hide it (you can click on the block ).

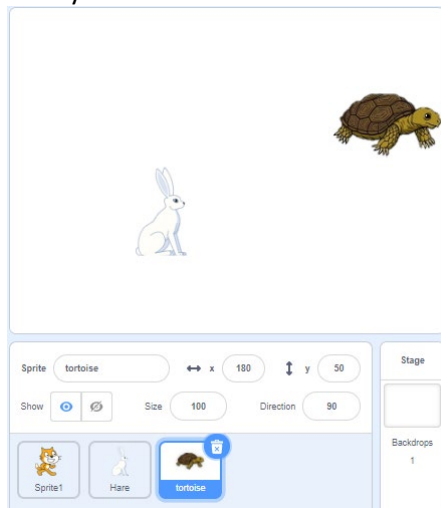
Now add a sprite Hare from the sprite library:



As we do not have the tortoise in the sprite library, we will upload the sprite file `tortoise.sprite3` which can be downloaded from the Canvas course page:



Now you should see both the Hare and the Tortoise sprites in your project:



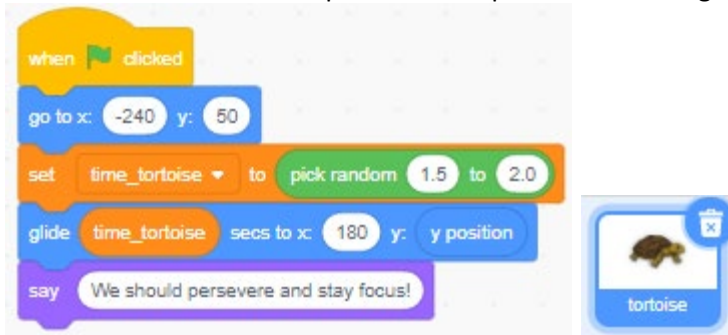
Now create 2 variables `time_hare` and `time_tortoise`, which are used to store the time for the hare to run before stopping to sleep and the time for the tortoise to move before reaching the destination.

Click on the Hare sprite and compose the following code:



The Hare will move from the left horizontally until $x=150$ after a random time from 0.5 to 1.0 second. Note that the block `y position` is the y-coordinate of that sprite (which will be equal to -100 as set by the earlier block). Since the y-coordinate never changes, it ensures that the sprite will move horizontally.

Now click on the Tortoise sprite and compose the following code:



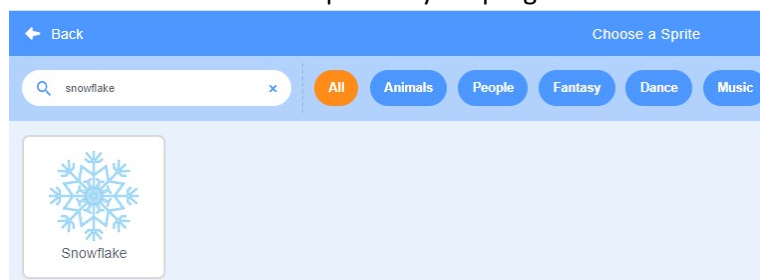
The Tortoise will move from the left horizontally until $x=180$ after a random time from 1.5 to 2.0 second. It is assumed that the Tortoise has then reached the destination on the right. It will then say the line to encourage people to persevere and stay focus.

Now run your program and observe the race between the hare and the tortoise!



Exercise 1.7: Write a program so that it satisfies the following specifications:

- You first add a snowflake sprite to your program



- b) When the program starts, the snowflake appears at position (x, y) , where x is a random integer with the range from -240 to 240, and y has a fixed starting value of 180.
- c) The snowflake then falls from top ($y=180$) to ($y=-180$) bottom with its horizontal position unchanged. The time it takes to fall from top to bottom is a random floating-point number between 3.0 to 5.0 seconds.

Run your program a few times to verify that it works.

Task 2 Complete the assessment from the Canvas course page

You should complete the [Lab 02 Assessment](#) from the Canvas course page before the posted deadline.

Task 3 Challenge your classmates

You can first reflect on what you have learnt in this lab, and then come up with problems to challenge your classmates. You can post your problem on the Canvas course page, under this [Discussion page](#). One should be able to solve your problem by using what he/she learns in Lab 02. You will not get extra marks by posting a challenging problem or solving a challenging problem posted by another student, but you will earn your fame so that you can impress the course leader, the lab tutors, and your classmates.
