Let’s get a “taste” of programming using Scratch.

Bring up the browser and go to: www.scratch.mit.edu

And click on create

We can write programs – called scripts by chaining together blocks that perform various actions.

**Input:** get input from the environment
- Sensing
- Events

**Output:** make something happen in the environment
- Looks
- Sound
- Motion
- Pen (used in conjunction with Motion)

**Memory:** save a value for later use
- Answer (Sensing)
- Data

**Processing:** do arithmetic and symbolic processing (+,-, concatenate …)
- Operators

**Control:** Control the order of the operations, do conditional processing (if), repetition (loops) …
- Control
**Input:** get input from ("sense") the environment. Click on “Sensing”.

You can ask for any single input value with this block: (This is also an example of “output” because it prints the “prompt” “What’s your name?”.)

![Ask Block](image)

and the answer is stored in the variable

![Answer Block](image)

This is an example of the use of “memory”.

So, if you click on the ask block above, when you type “Bob” in the input bar and press the enter key, you see

![Input Block](image)

If you click on “Sensing” in the “Scripts” tab, you get a list of all the things that you can “sense”.

You can also get input from “Events”. Note: here too you are getting “input: about the state of the environment.

![Events Block](image)
**Output:** Change something in the environment. In Scratch its under “**Looks**” because it changes how the environment “looks”. Click on “Looks”.

Here are some of the things that you can do:

### Processing:  
Lets you do arithmetic, ask “yes/no” questions using “logic”, and “join”ing (combining) various values.

### Processing:  
- Arithmetic: `+`, `-`, `*`, `/`
- Comparison: `<`, `>`, `=`, `<=`, `>=`
- Logic: `and`, `or`, `not`
- Join: `join`
- String manipulation: `length_of`, `letter_1_of`, `word`
- Mathematical functions: `mod`, `round`, `sqrt`,...
Example: Write a Scratch program to ask the user for their name and greet them by name.

Problem: Modify the program so that we get an exclamation point after the name:

Problem: Write a program to ask the user their age in years and have the program tell them their age in minutes. Don’t worry about leap years.

Problem: write a program to ask the user for the temperature in Fahrenheit and print out the resulting temperature in centigrade. The formula is:

<table>
<thead>
<tr>
<th>How to convert Fahrenheit to Celsius</th>
</tr>
</thead>
</table>
| The temperature \( T \) in degrees Celsius \( (^\circ C) \) is equal to the temperature \( T \) in degrees Fahrenheit \( (^\circ F) \) minus 32, times \( \frac{5}{9} \):
| \[ T(\circ C) = \left( T(\circ F) - 32 \right) \times \frac{5}{9} \] or
| \[ T(\circ C) = \left( T(\circ F) - 32 \right) / \left( \frac{9}{5} \right) \] or
| \[ T(\circ C) = \left( T(\circ F) - 32 \right) / 1.8 \] Example
| Convert 68 degrees Fahrenheit to degrees Celsius:
| \[ T(\circ C) = (68^\circ F - 32) \times \frac{5}{9} = 20 \, ^\circ C \]
Let’s draw some pictures! For this we will use the blocks from “Pen” and “Motion”.

Problem: Have the sprite draw a square with each side equal to 100 steps. Do this with nine blocks. One from Pen and eight from Motion.

**Control**: All the programs that we have written so far executed in a straight line, block after block. We can alter the order by using some of the blocks from the “Control” group.
Problem: Have the sprite draw a triangle with each side of length 100. It should look like this:

![Triangle](image)

Problem: Have the sprite draw a hexagon with each side of length 100. It should look like this:

![Hexagon](image)
Problem: Have the sprite draw a circle. It should look like this:

![Circle](image)

Problem: Have the sprite draw a 5 pointed star. It should look like this:

![Star](image)

Let’s get scratch to add two numbers.

![Add Two Numbers](image)

Here is the name program you wrote above, but in a loop.

![Name Program Loop](image)
We can make our own blocks.

When we click on “Make a Block” we get to name it:

And the when we use the name our block will be executed. Had we called our new block square we could define it with the blocks on the left and when we click the new block name on the right the sprite will draw a square.

We can even pass “parameters” into our block so that it modifies its behavior based on the value that we pass in.

For example, our square above will always be 100X100. We can make a more flexible square that lets us specify the length of a side. When we define our new block, we click on “Options” and choose "Add number input:"

Edit Block

Options

- Add number input:
- Add string input:
- Add boolean input:
- Add label text:
- Run without screen refresh

OK Cancel
Here is one possible block for a square with a parameter to allow for a variable length side:

Here is a block to reset the sprite to its original position in the center of the screen:

Let's redo some of the above examples in Python!

When you click on it, it will bring up the “Python Shell” called IDLE that you see below.

If you don’t see the icon, you can type idle into the search Windows box:
Now click on the IDLE shortcut and Python will start up in the IDLE shell:

We are now ready to write some Python programs!

A program is a sequence of Python statements. We will first create a file to hold the program, then will enter the Python statements. Finally we will instruct Python to run our program.

To create the file, click on the “File” button in the IDLE shell:

Now enter the program below in the file you just created.

```python
name=input("What is your name? ")
print("hello", name)
```

Before we run the program we need to save the file. Press F5:
Press OK and give the file a name and press save. The program will run.

Here is a sample run.

```python
>>> What is your name? Jerry
hello Jerry
>>> |
```

Here is an example of looping in Python:

```python
for i in range(5):
    print("Hello")
```

prints:

```python
>>> Hello
Hello
Hello
Hello
Hello
>>> |
```

Let’s do some simple graphics with Python! It’s called “Turtle Graphics” because we will issue commands to a “turtle”.

```python
# the following statement makes the turtle commands available to us
from turtle import *
```

Here are some turtle commands:

penup()
pendown()  
forward(n) - move n positions in the direction the turtle is pointing  
back(n) - move n positions in the direction opposite the one that the turtle is pointing  
right(n) – rotate the turtle n degrees in a clockwise direction  
left(n) – rotate the turtle n degrees in a counter-clockwise direction

Example:

What will this do?

```python
for i in range(10):
    pendown()
    forward(15)
    penup()
    forward(15)
```

Think first, write your answer below then run the program to check your answer.

Let’s write the following programs:

1. Write a Python program to draw a square with each side of length 100.

2. Write a Python program to draw a triangle with each side of length 100. It should look like this:
3. Write a Python program to draw a triangle with each side of length 100. It should look like this:

![Triangle]

4. Write a Python program to draw a hexagon with each side of length 100. It should look like this:

![Hexagon]

5. Write a Python program to draw a circle. It should look like this:

![Circle]

6. Write a Python program to draw a 5 pointed star. It should look like this:

![Star]

7. What do you think that this one does? How/why?

```python
for i in range(100):
    forward(10+5*i)
    right(120)
```
Remember, a computer can:

Some Python equivalents …

<table>
<thead>
<tr>
<th>input</th>
<th>x=eval(input(&quot;Please enter a number&quot;))</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>print(&quot;You entered&quot;, x)</td>
</tr>
<tr>
<td>memory</td>
<td>y= … something ....</td>
</tr>
<tr>
<td>process</td>
<td>x=y+1,  y=5*x,  y=x/15+9 …</td>
</tr>
<tr>
<td>control</td>
<td>for, if, while .. these statement alter the program flow</td>
</tr>
</tbody>
</table>

8. Write a program to ask the “user” how old they are in years. The program should print out how many days they have lived. Assume there are no leap years.

9. Write a program to ask the user for two numbers. Your program should print the following:

```python
>>> Please enter the first number: 3
>>> Please enter the second number: 5
>>> The sum of 3 and 5 is 8
```
Here is a more general version. It keeps looping until the user inputs ‘done’.

# The Python “sum” program

print("Welcome to the addition program.")

print("You can enter values for x and y and I will calculate ")

print("and display the sum.")
print() # prints a blank line

x=input("Please enter a value for x (entering 'done' terminates the program): ")

while x !='done': # this a ‘while loop’. It’s an example of “control”.
    x=int(x)
    y=int(input("Please enter a value for y: "))
    sum=x+y
    print("The sum of ",x," and ",y," is: ",sum)
    print()
    x=input("Please enter a value for x (entering 'done' terminates the program): ")

print()
print("Thanks for trying our program!")