FORCES ON INCLINED PLANES

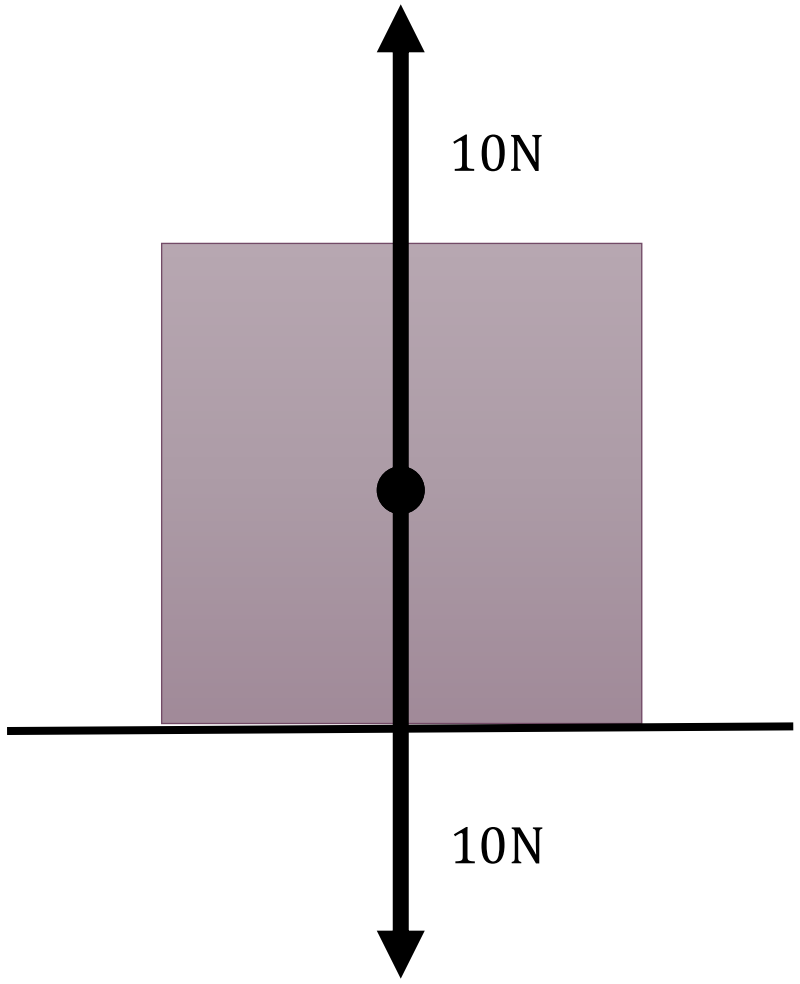
Unit 4: Dynamics

OBJECTS ON A FLAT PLANE

When an object lies on a **flat plane**, **normal force** and **weight** are parallel on the same line.

Thanks to **Newton's Third Law**, this means that the magnitude of the normal force is equal to the total magnitude of all downward forces.

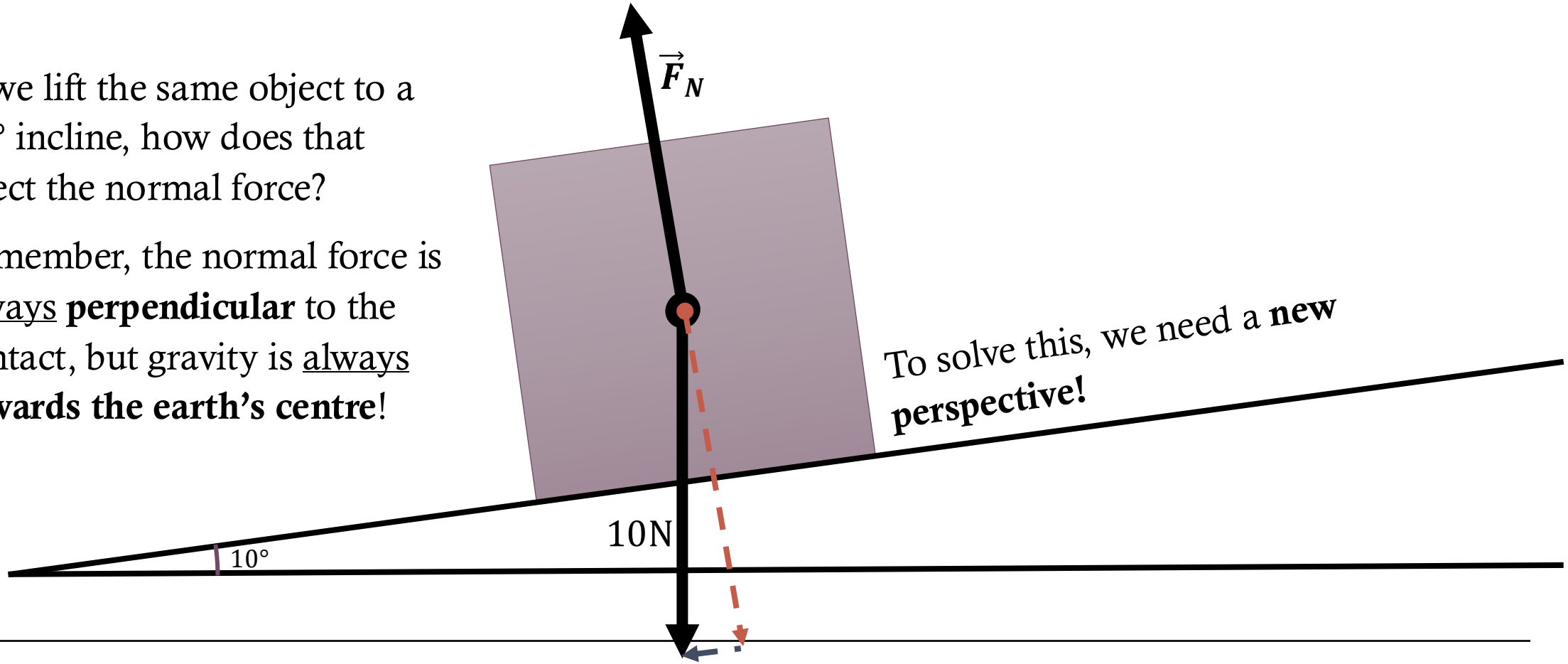
If this object has a weight of 10N and there are no other vertical forces, then the normal force from its contact with the surface must also be 10N.



OBJECTS ON AN INCLINED PLANE

If we lift the same object to a 10° incline, how does that affect the normal force?

Remember, the normal force is <u>always</u> **perpendicular** to the contact, but gravity is <u>always</u> **towards the earth's centre**!

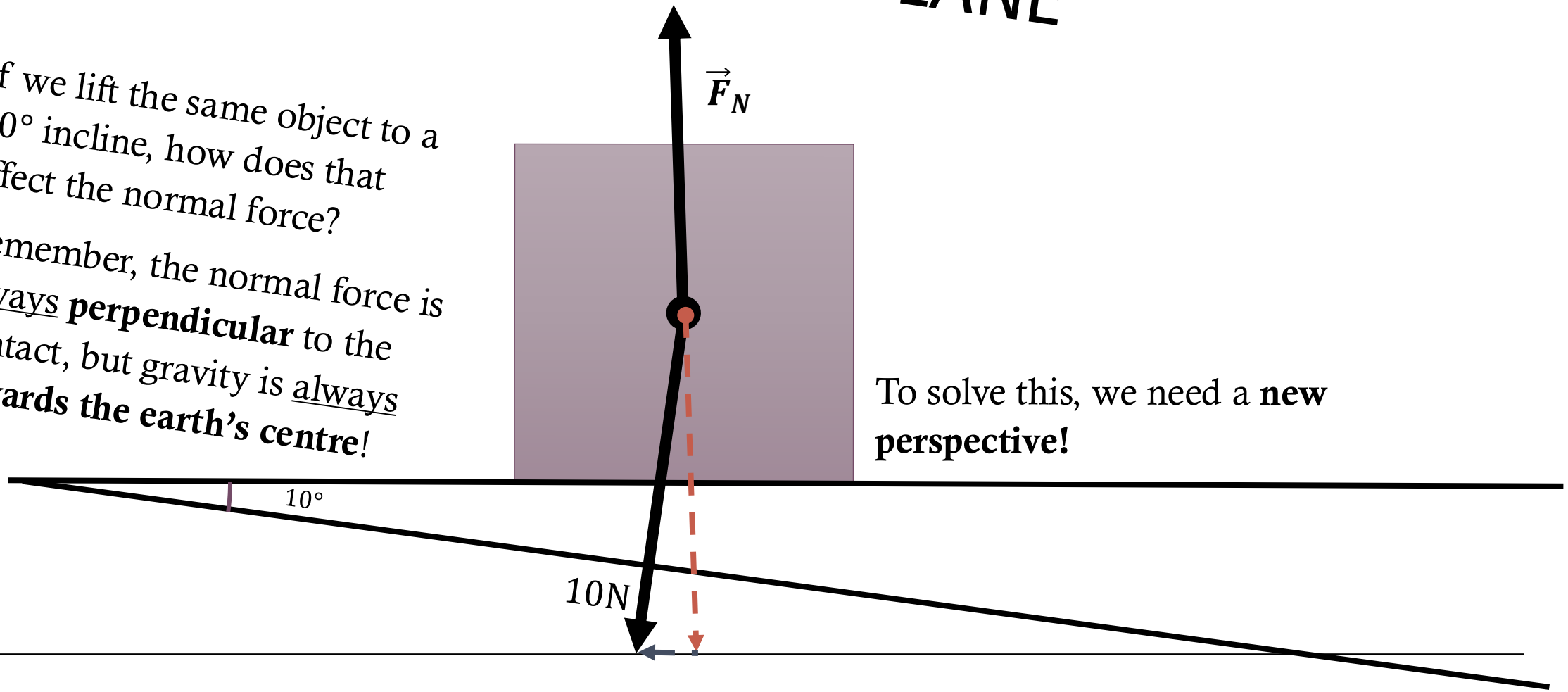


To solve this, we need a **new perspective!**

OBJECTS ON AN INCLINED PLANE

If we lift the same object to a 10° incline, how does that affect the normal force?

Remember, the normal force is always **perpendicular** to the contact, but gravity is always **towards the earth's centre**!

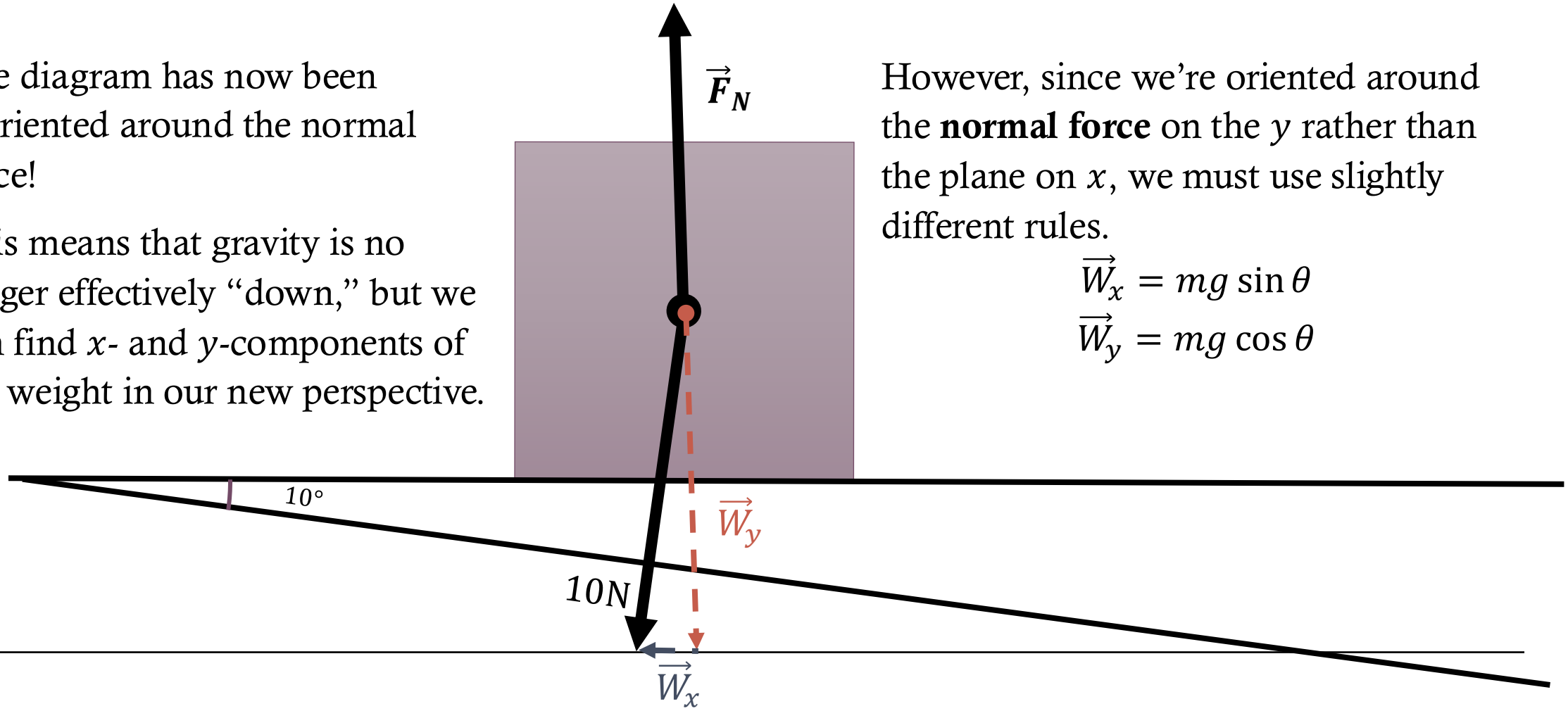


To solve this, we need a **new perspective!**

CHANGING PERSPECTIVE

The diagram has now been reoriented around the normal force!

This means that gravity is no longer effectively "down," but we can find x- and y-components of the weight in our new perspective.

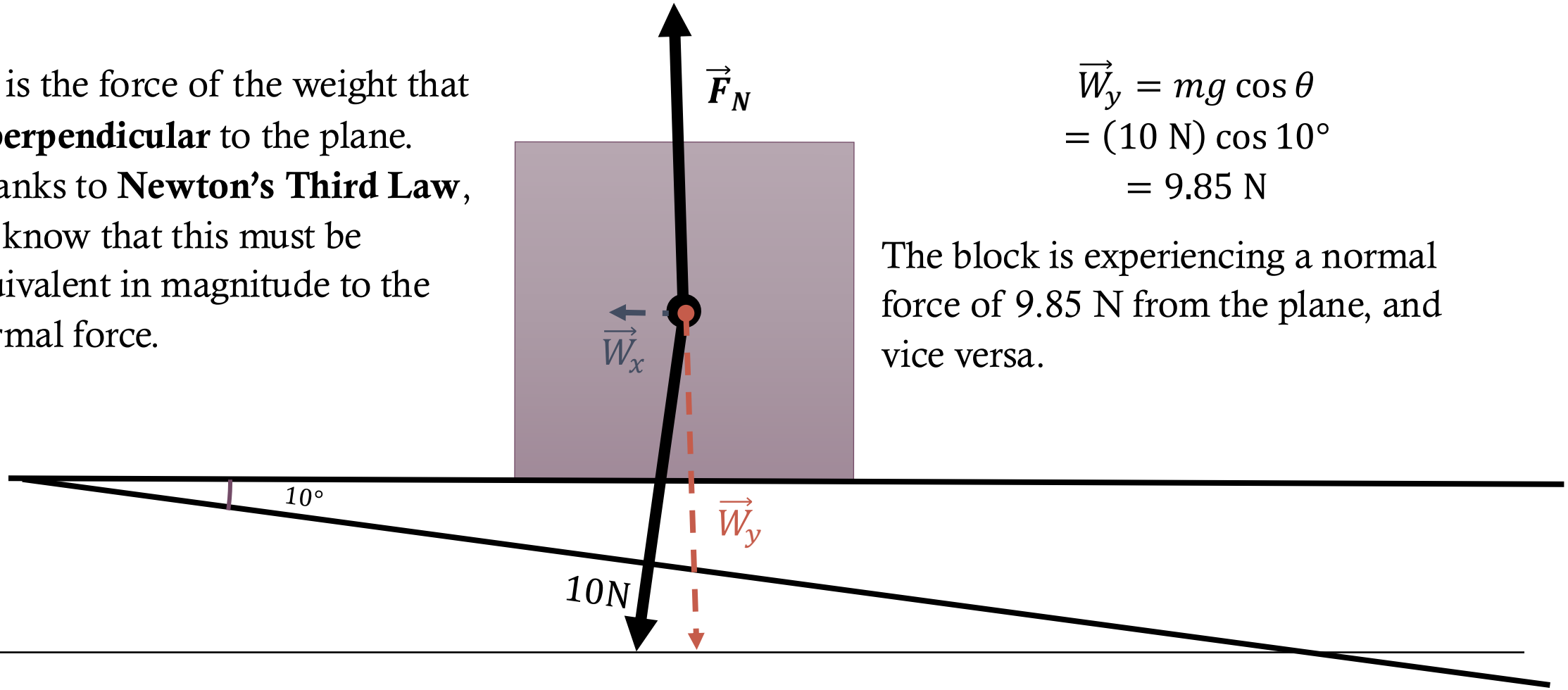


However, since we're oriented around the **normal force** on the y rather than the plane on x, we must use slightly different rules.

$$\vec{W}_x = mg \sin\theta$$
$$\vec{W}_y = mg \cos\theta$$

CHANGING PERSPECTIVE: THE Y COMPONENT

$\vec{W}_y$ is the force of the weight that is **perpendicular** to the plane. Thanks to **Newton's Third Law**, we know that this must be equivalent in magnitude to the normal force.



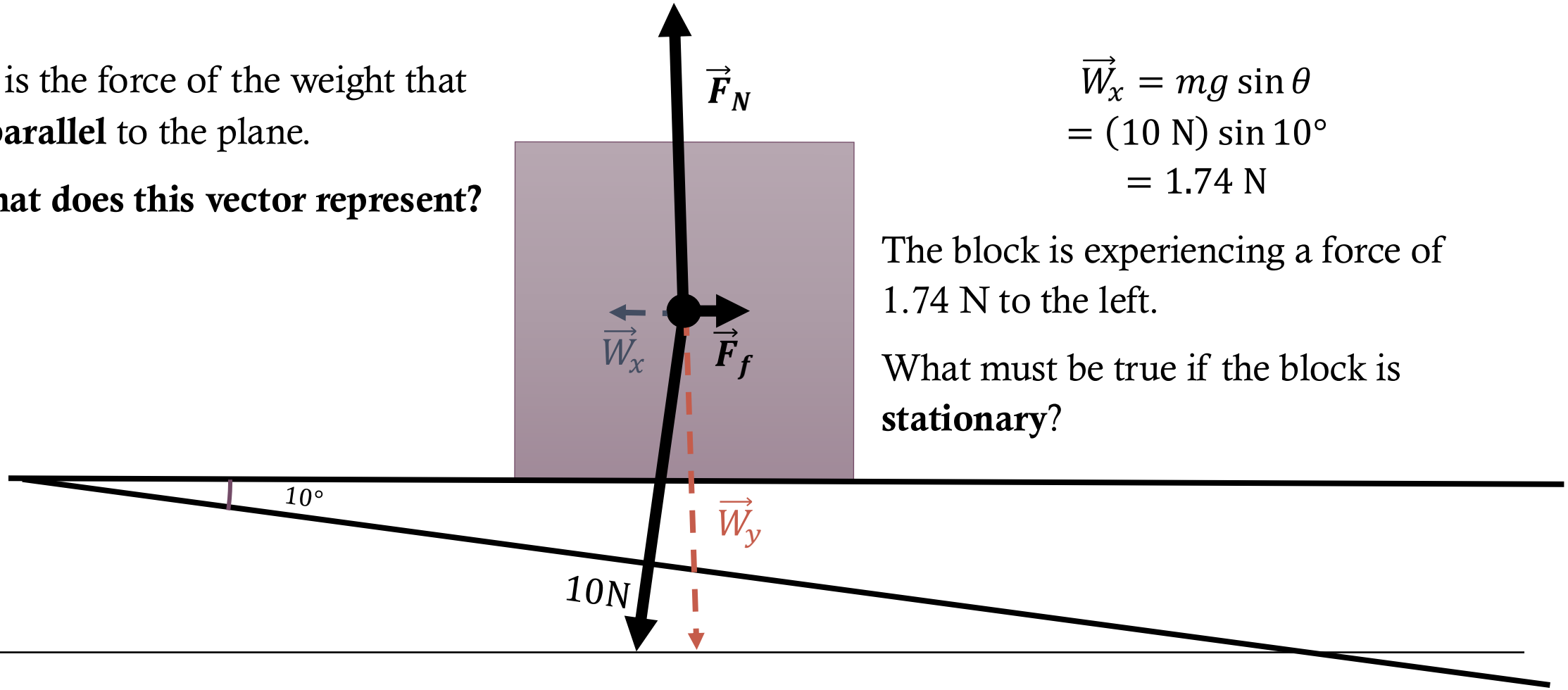
$$\vec{W}_y = mg\cos\theta$$
$$= (10\text{ N})\cos 10°$$
$$= 9.85\text{ N}$$

The block is experiencing a normal force of 9.85 N from the plane, and vice versa.

CHANGING PERSPECTIVE: THE X COMPONENT

$\vec{W}_x$ is the force of the weight that is **parallel** to the plane.

What does this vector represent?



$$\vec{W}_x = mg \sin\theta$$
$$= (10\ \text{N}) \sin 10°$$
$$= 1.74\ \text{N}$$

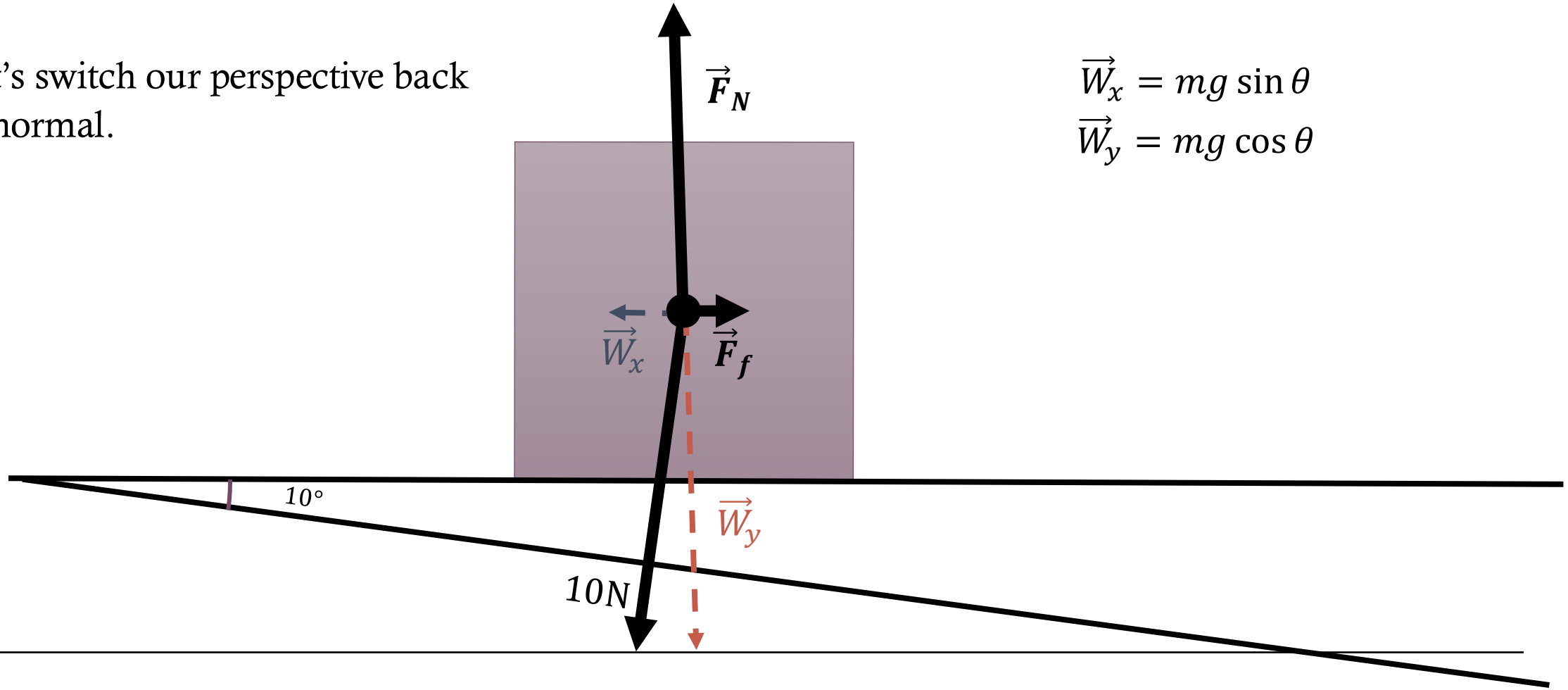
The block is experiencing a force of 1.74 N to the left.

What must be true if the block is **stationary**?

CHANGING PERSPECTIVE

Let's switch our perspective back to normal.

$$\vec{W}_x = mg\sin\theta$$
$$\vec{W}_y = mg\cos\theta$$



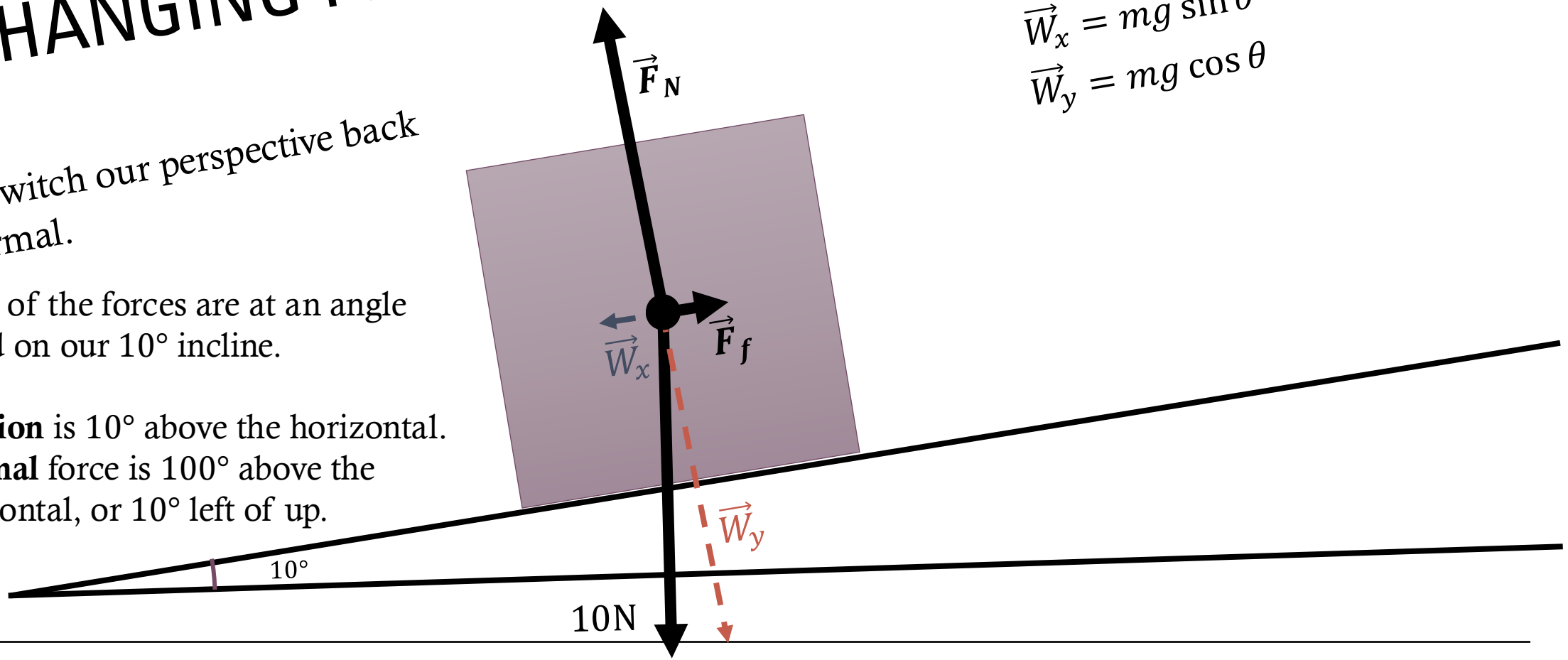
CHANGING PERSPECTIVE

Let's switch our perspective back to normal.

Most of the forces are at an angle based on our 10° incline.

Friction is 10° above the horizontal.
Normal force is 100° above the horizontal, or 10° left of up.

$$\vec{W}_x = mg\sin\theta$$
$$\vec{W}_y = mg\cos\theta$$



FRICTION AND INCLINES

While static friction ($\vec{F}_f \leq \mu_s \vec{F}_N$) can hold an object in place, there is **minimum angle** at which the force of gravity overcomes this force of friction.

$$\vec{F}_{gx} = mg \sin\theta$$

$$\vec{F}_{f\,\max} = \mu_s \vec{F}_N = \mu_s (mg \cos\theta)$$

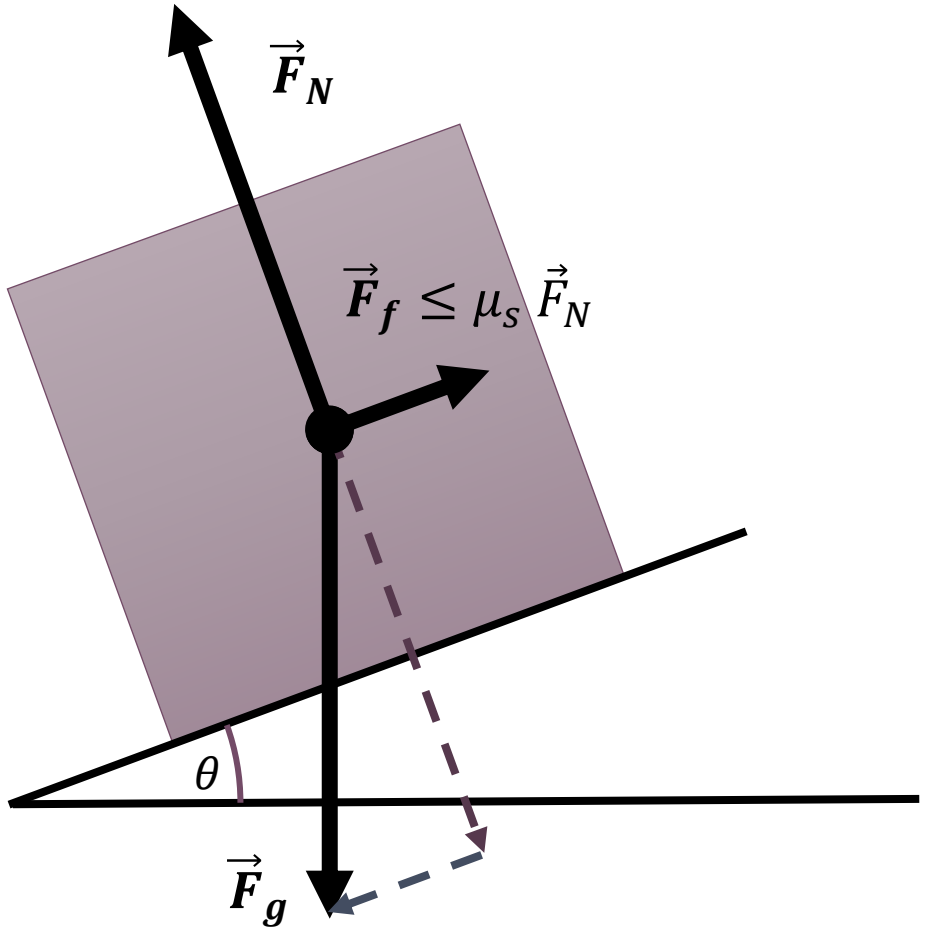
When gravity overcomes friction,

$$\vec{F}_{gx} = \vec{F}_{f\,\max}$$

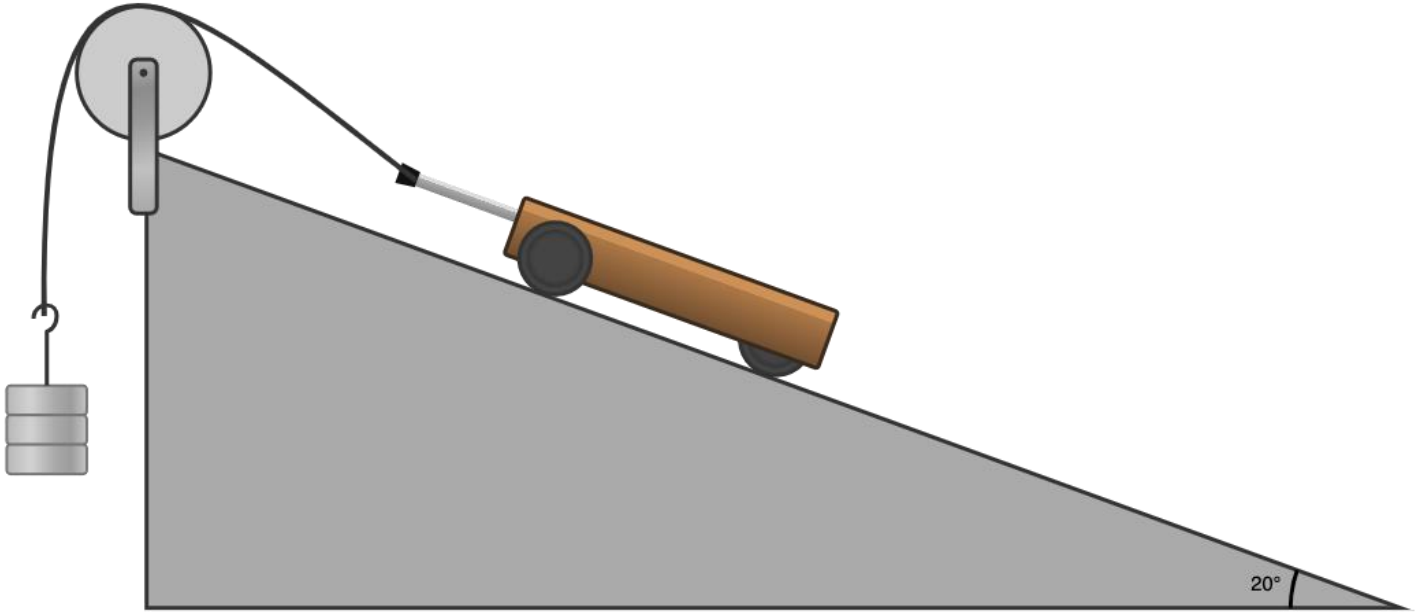
$$mg \sin\theta = \mu_s mg \cos\theta$$

$$\therefore \frac{\sin\theta}{\cos\theta} = \mu_s \implies \mu_s = \tan\theta$$

This means that we can use an incline to measure μ_s!



EXAMPLE

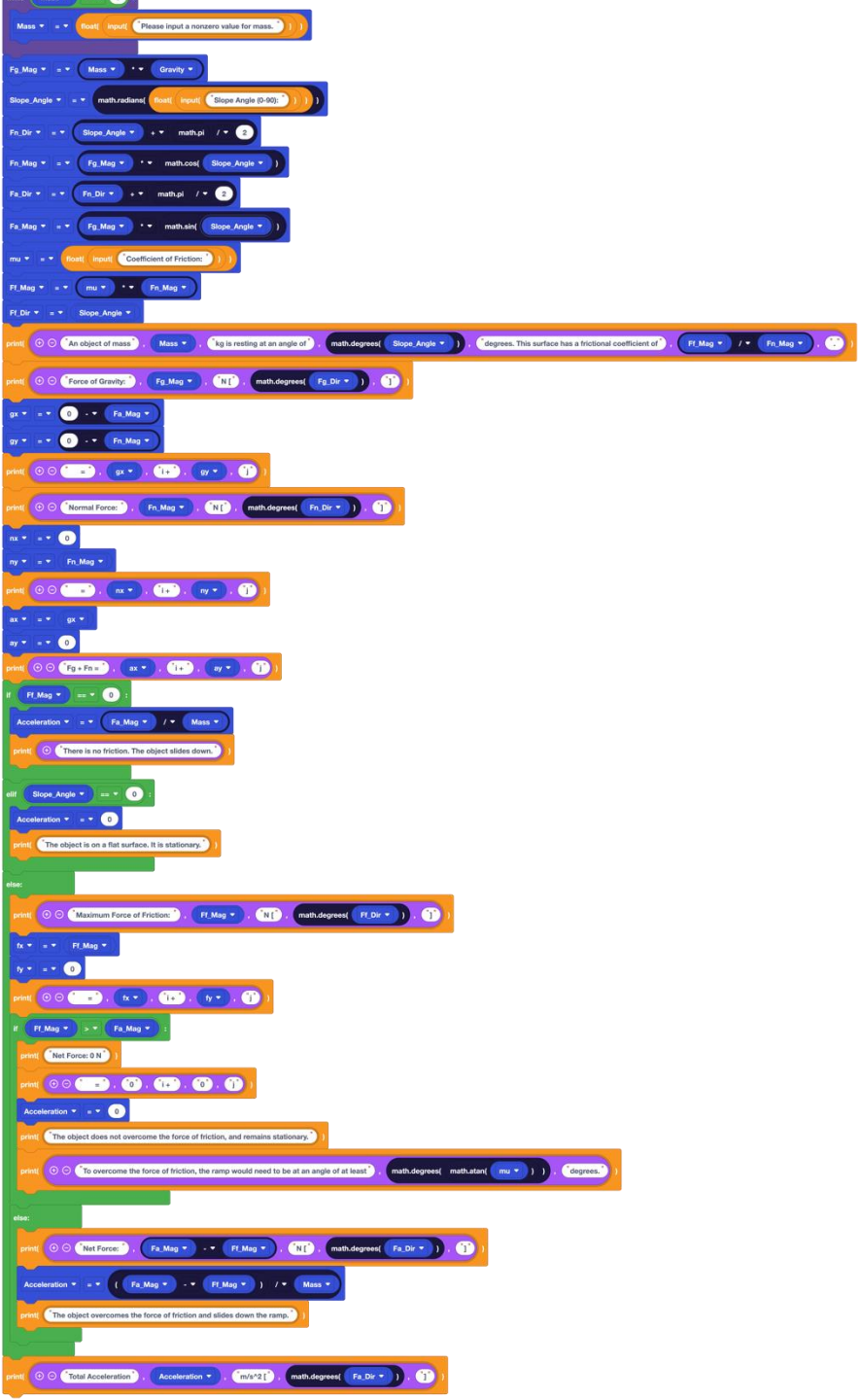


If the angle of the ramp is 20° and the suspended mass is 60 g, what is the **maximum** that the cart can weigh if they are both stationary?

Assume a μ_s of 0.2.

PROJECT

Build your own
RAMP
CALCULATOR

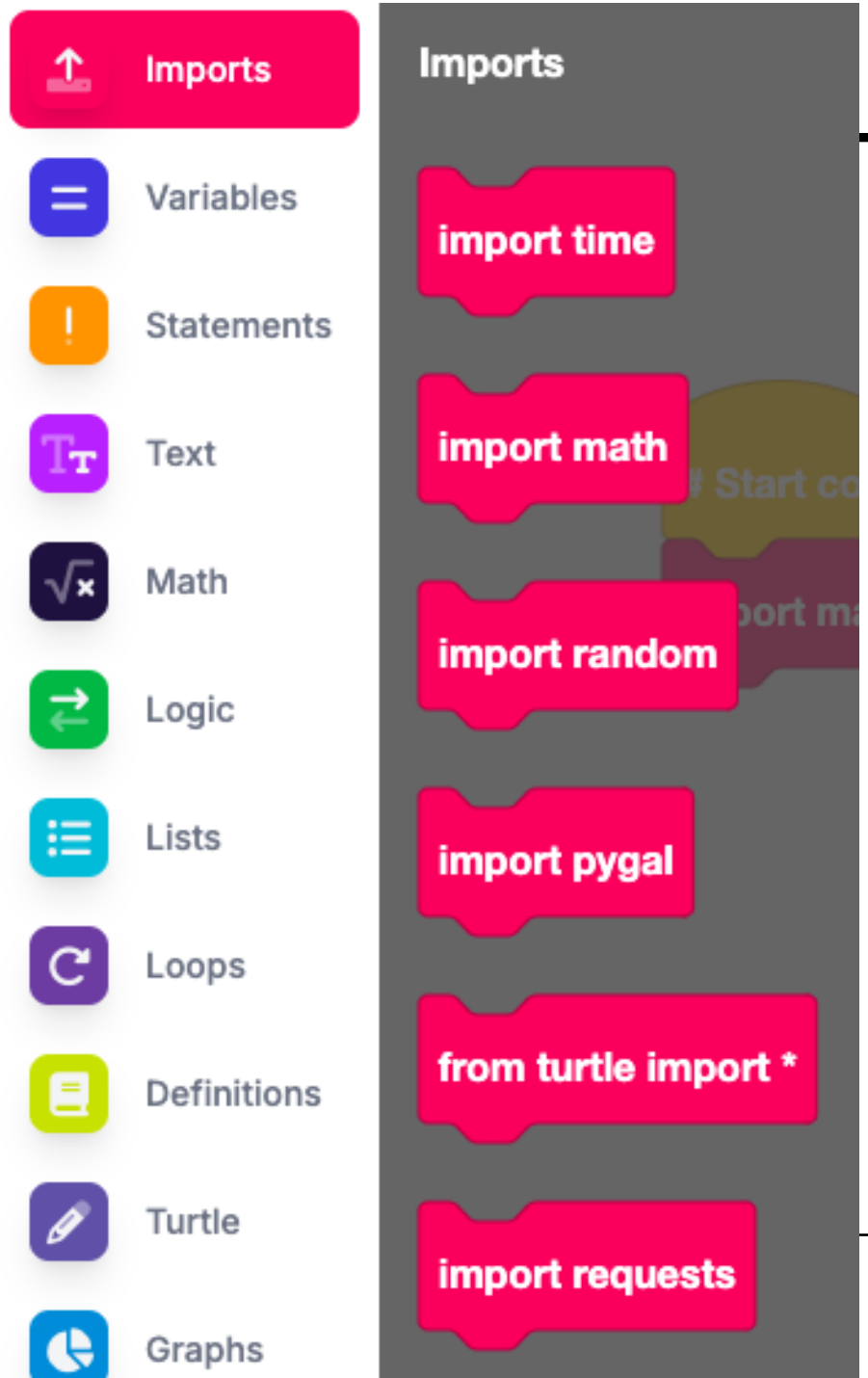


BLOCK CODING

You are going to be building a tool to help you solve inclined plane problems. To do this, you're going to need to learn a **programming language.**

EduBlocks is a site that allows users to create powerful **Python** code using what are called "**blocks**" of code.

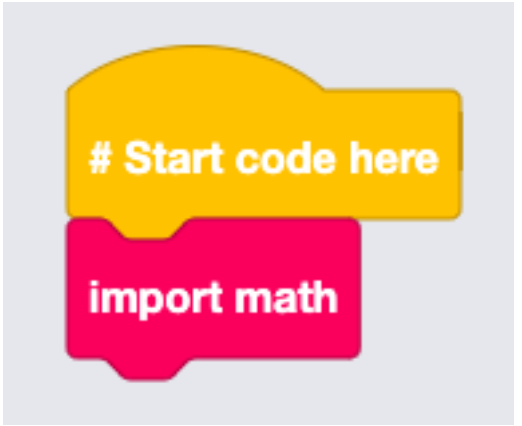
Blocks are a visual representation of a code's structure which can be dragged and dropped from a menu into place. The site then translates these blocks into actual Python.

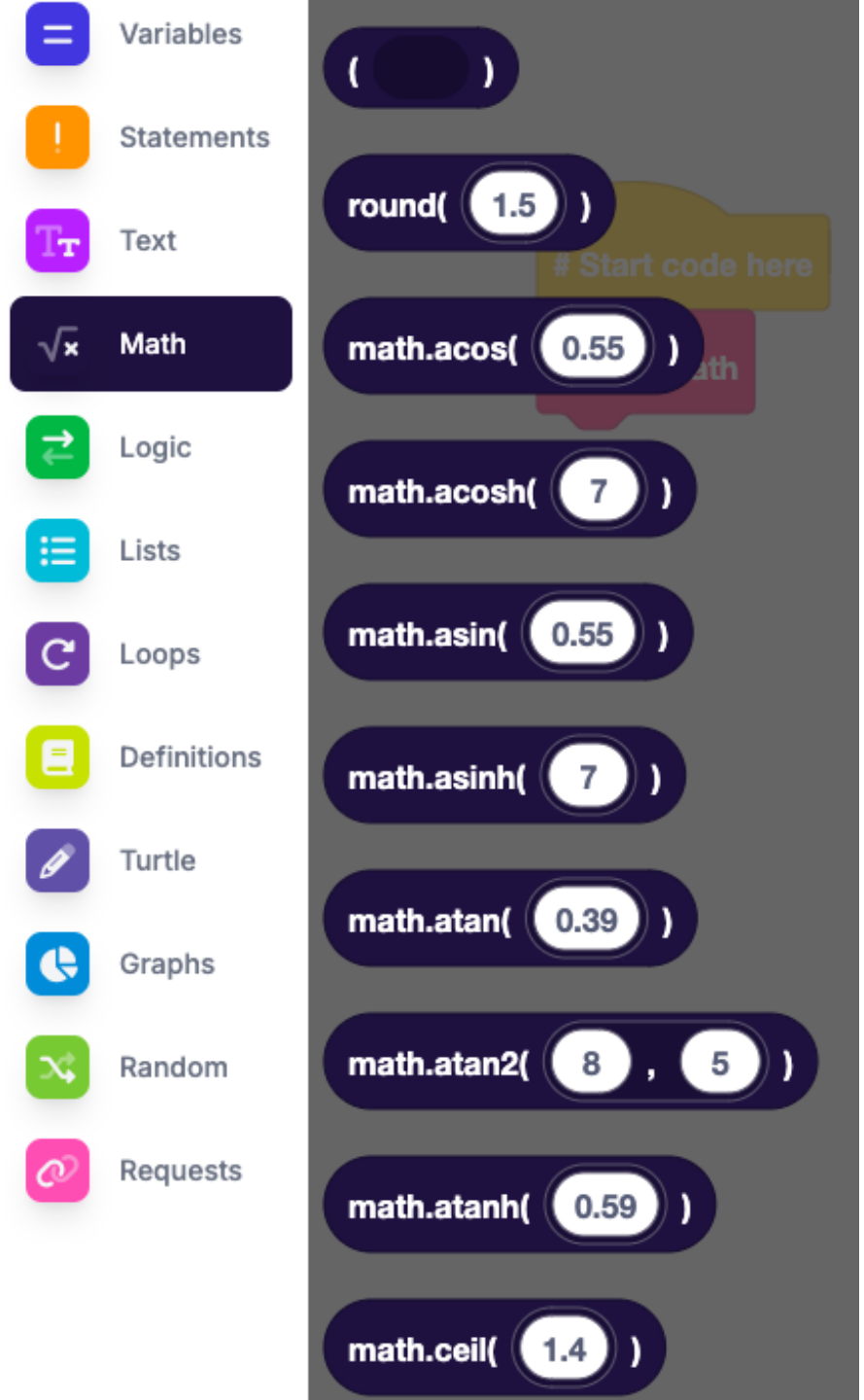


LIBRARIES

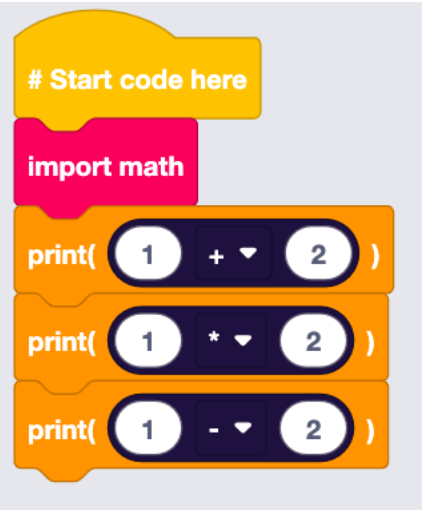
Right after the **# Start code here** block, we're going to need to start with a **library**. Libraries can contain useful commands that you can use in your code.

Since we're going to need a lot of **trigonometry**, we're going to need to **import** the **math** library!





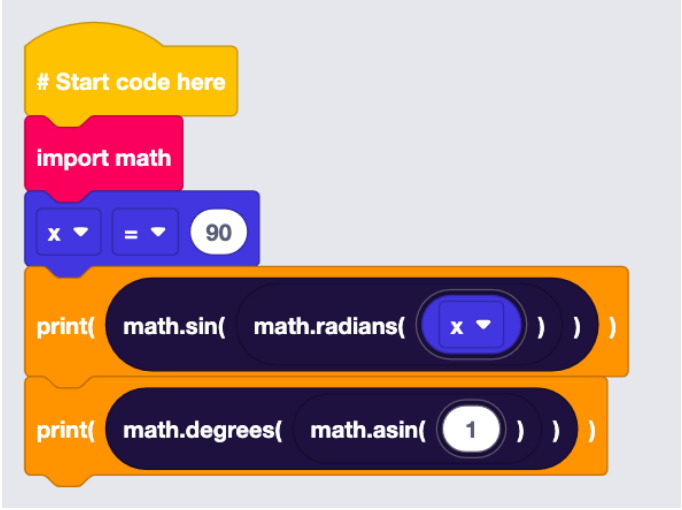
MATH



The **math** library contains many of the commands you are going to need in this project.

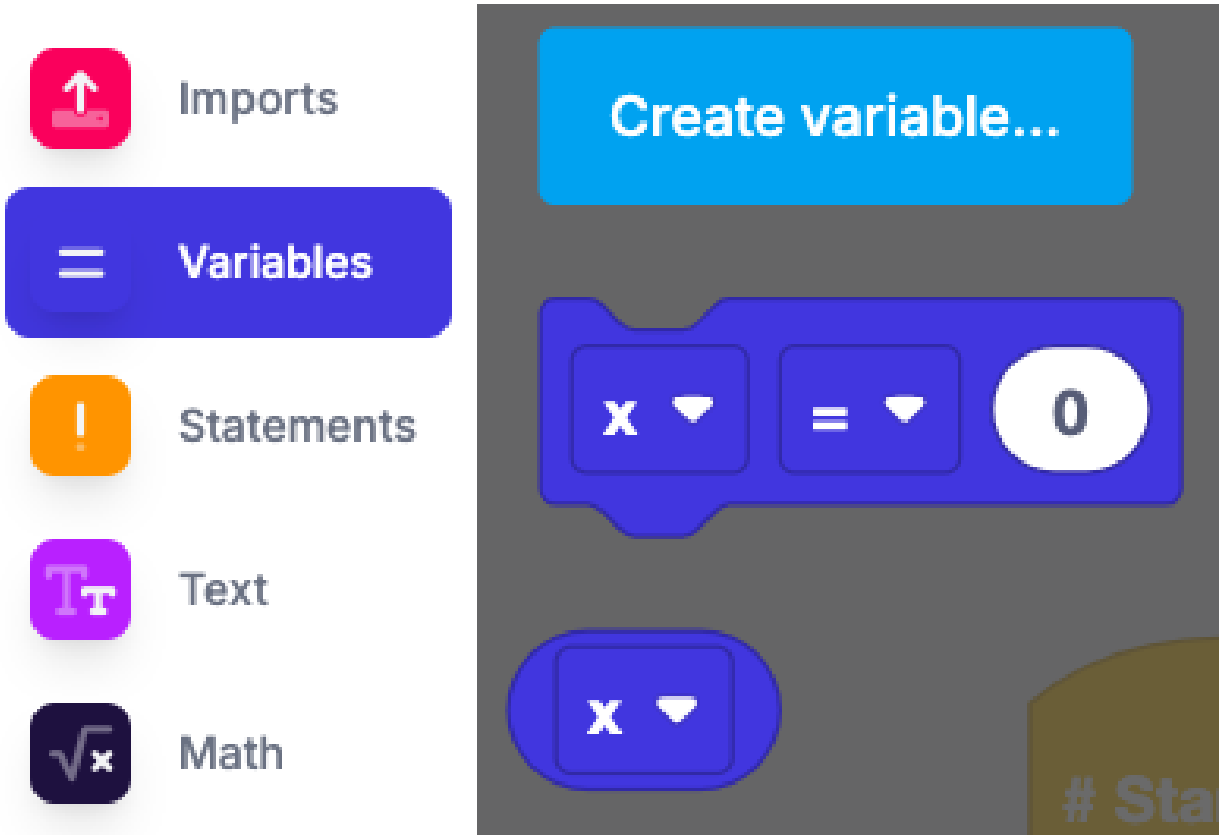
While simple arithmetic functions like **addition**, **multiplication**, and **subtraction** can be performed without the math library, functions like **degree/radian conversion** and **trigonometry** are going to need to import math first.

Important: Trig functions in Python only use radians!



Notice the different shapes of blocks! "Puzzle piece" blocks are the functions, while round blocks go inside of them. You can also put round blocks inside of other round blocks!

VARIABLES



Just like in mathematical equations, **variables** are essential to programming.

To make sense of your code and keep it clean and neat, you're going to need to use variables to represent values.

What variables are you going to need for forces on an inclined plane?

INPUT AND OUTPUT

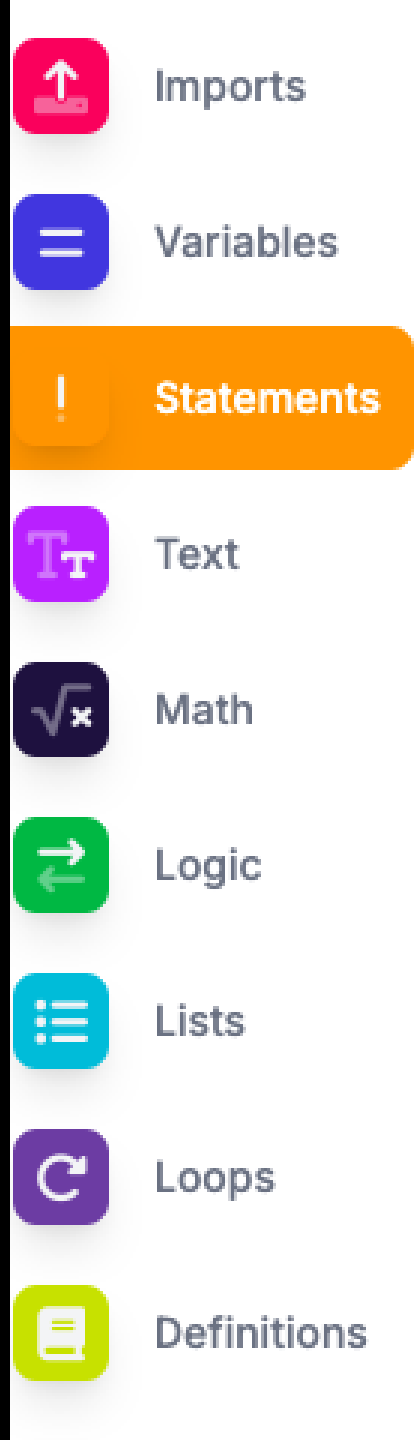
Equally important for programming is the **input/output** of the program.

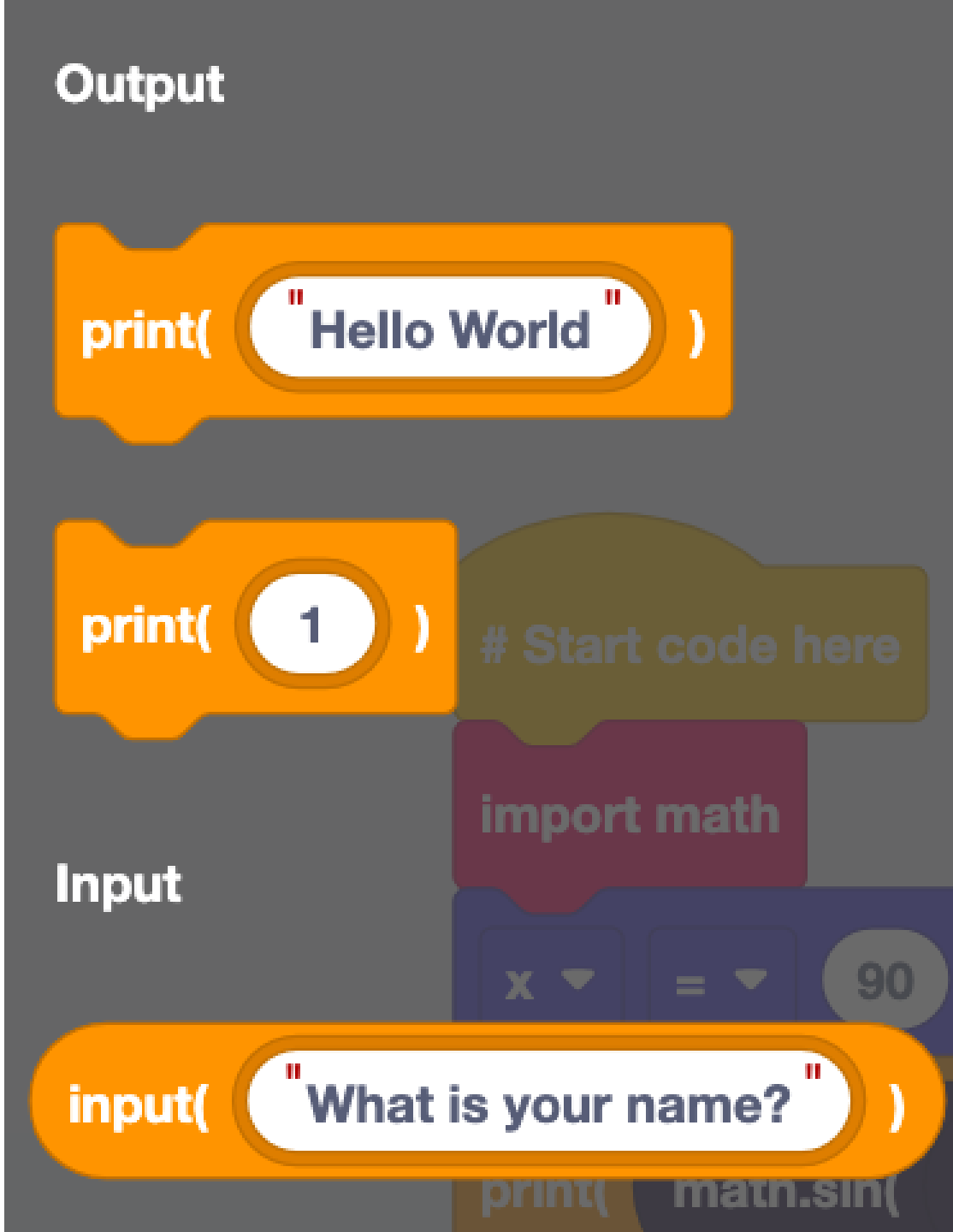
Output allows the program to provide information to the user, printing out text or numbers to the display.

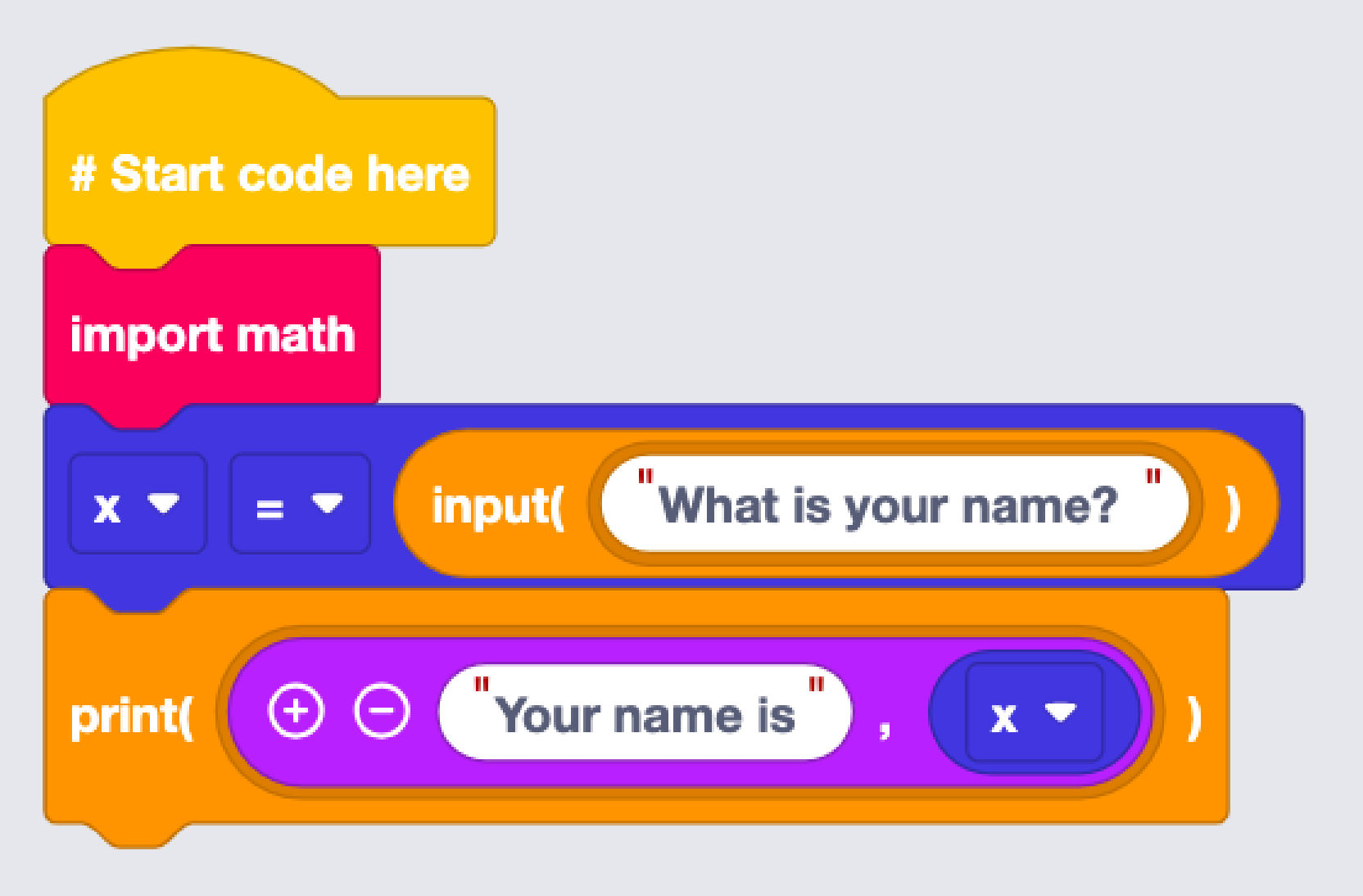
Input allows the user to provide information to the program to allow it to run with the desired values or information.

Your code is going to need **three inputs**:

- The mass of the object
- The angle of the ramp
- The coefficient of friction between the ramp and the object (simplified to one value)



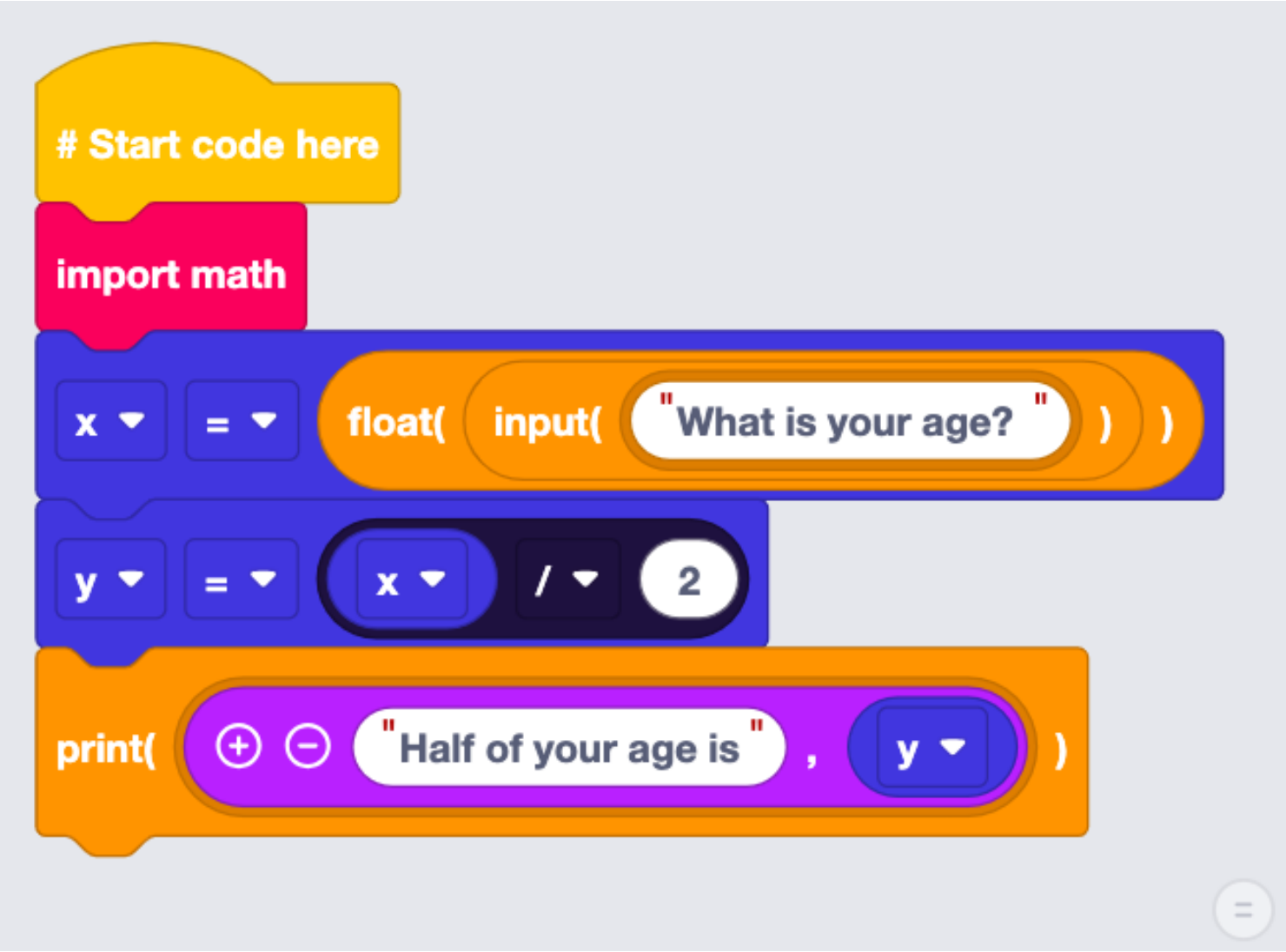




INPUT, OUTPUT, AND VARIABLES

The block code above will take your name as **input**, then **output** to repeat your name back to you.

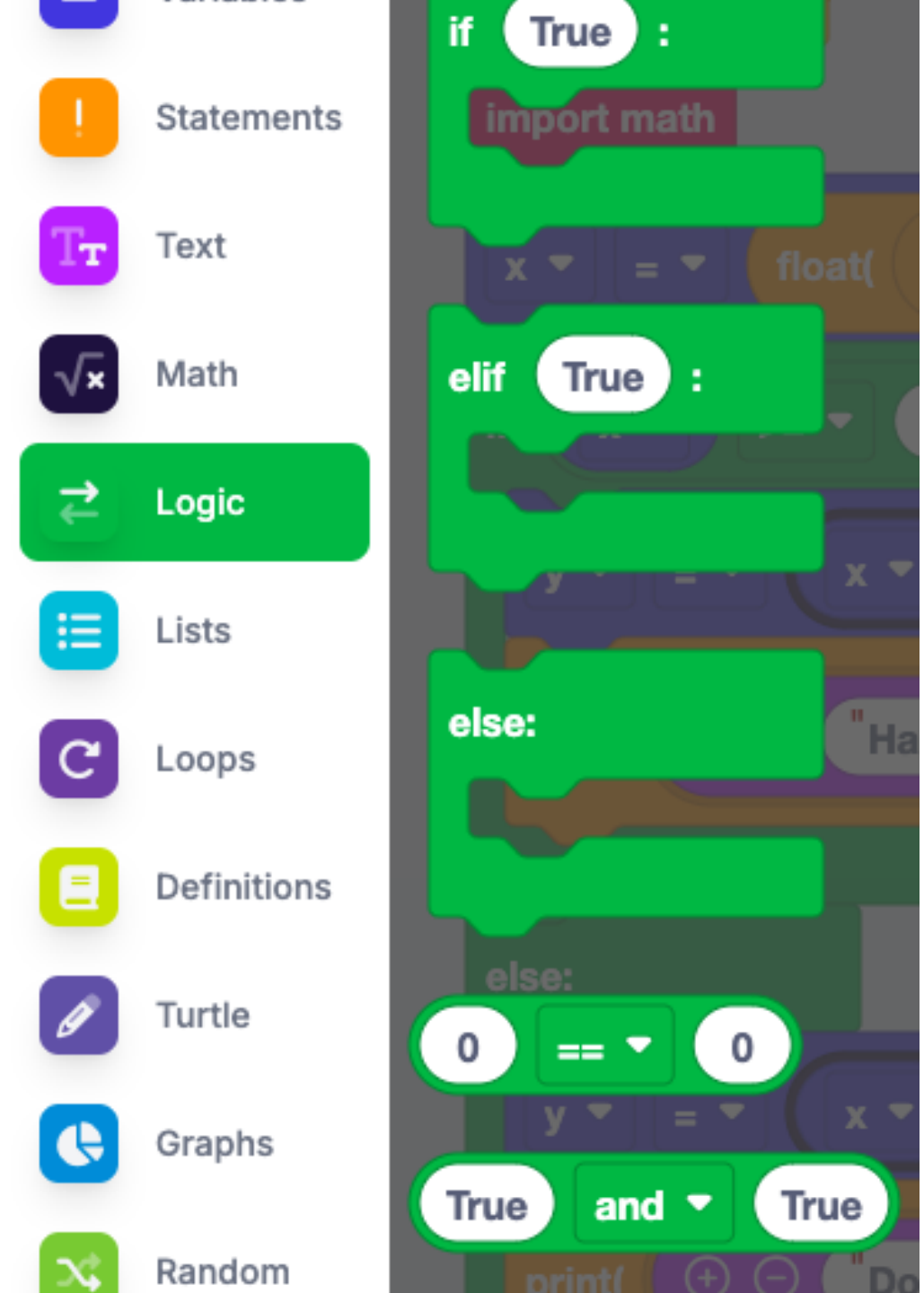
It is **assigning** the value of the variable **x** to become your name.



INPUT, OUTPUT, AND VARIABLES

Now the code is expecting a **float** input.

A **float** is a type of value that allows numbers with decimal points. You will want your variables in the form of **floats** to do trig to them!



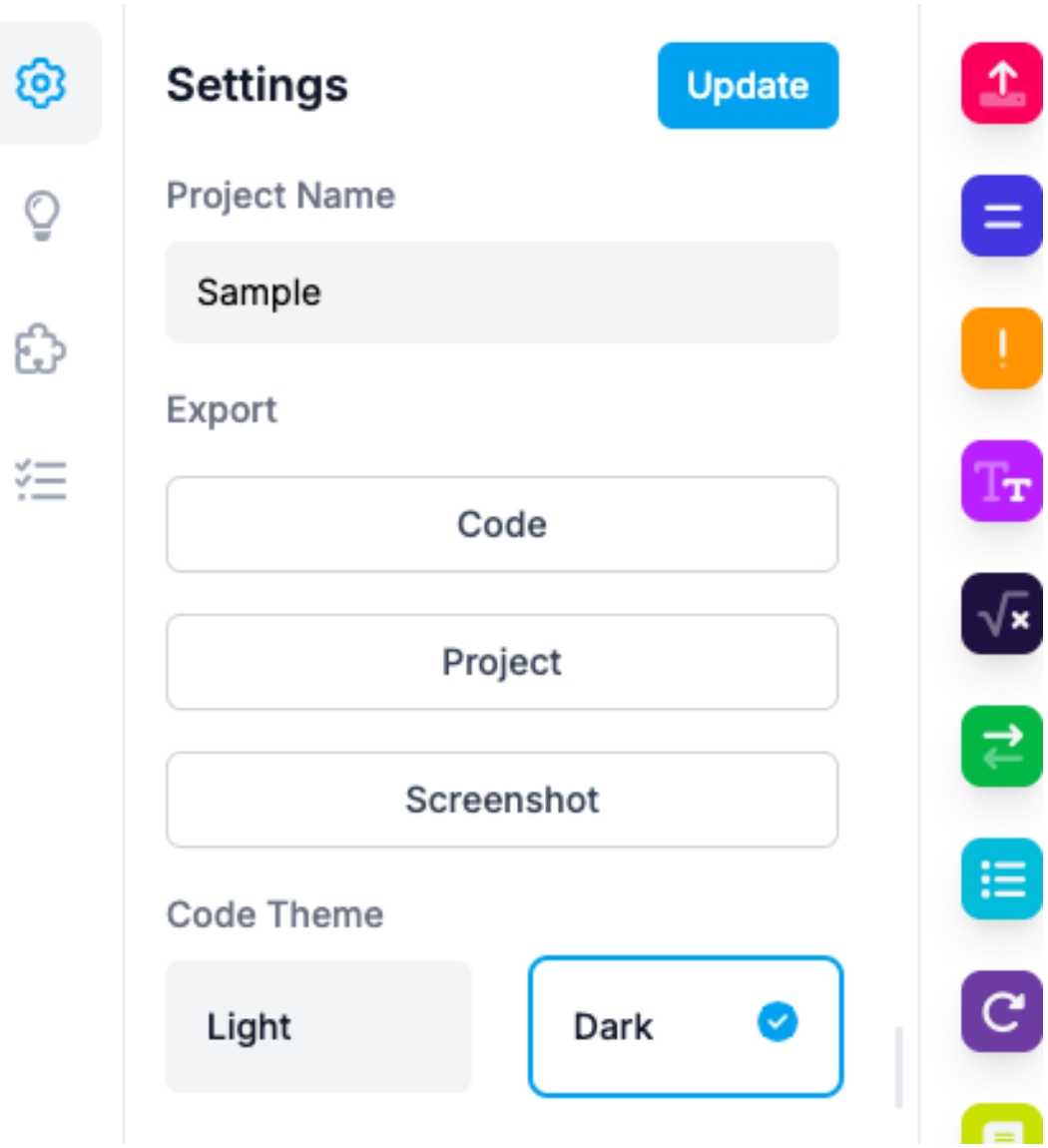
CONDITIONS AND LOGIC

Sometimes when programming, you need the code to sometimes do one thing, and sometimes do another. This is when we use **conditions**.

In this case, **<u>if</u>** the age is 17+, then we halve it. If it's anything **<u>else</u>**, then we double it.

Remember: If $mg \sin\theta \leq \mu_s F_N$, then the object has **no net force**!

```
# Start code here
import math
x = float( input( "What is your age?" ) )
if x >= 17 :
    y = x / 2
    print( "Half of your age is" , y )
else:
    y = x * 2
    print( "Double your age is" , y )
```



ASSIGNMENT

- Create a program that can take as inputs the **mass** of an object (in kg), the **angle** of an incline (in degrees), and the **coefficient of friction** (in this case, we are assuming that $\mu_d = \mu_s$). It should, as an output, provide the user with the **net force** and **acceleration** of the object.
- Submit both the **.py code file** and the **screenshot** of the block code.
- The code must be able to properly generate correct results for the following test cases:

m / kg	3	4	0.5	100
θ / °	15	40	0	20
μ	0.1	1	0.2	0

SAMPLE CODE (FOR INSPIRATION)

```
#Start code here
import math
Gravity = 9.81
Fg_Dir = math.radians(270)
Mass = float(input("Object Mass (kg): "))
while Mass == 0:
  Mass = float(input("Please input a nonzero value for mass. "))
Fg_Mag = Mass * Gravity
Slope_Angle = math.radians(float(input("Slope Angle (0-90): ")))
Fn_Dir = Slope_Angle + math.pi / 2
Fn_Mag = Fg_Mag * math.cos(Slope_Angle)
Fa_Dir = Fn_Dir + math.pi / 2
Fa_Mag = Fg_Mag * math.sin(Slope_Angle)
mu = float(input("Coefficient of Friction: "))
Ff_Mag = mu * Fn_Mag
Ff_Dir = Slope_Angle
print("An object of mass", Mass, "kg is resting at an angle of", math.degrees(Slope_Angle), "degrees. This surface has a frictional coefficient of", Ff_Mag / Fn_Mag, ".")
print("Force of Gravity: ", Fg_Mag, "N [", math.degrees(Fg_Dir), "]")
gx = 0 - Fa_Mag
gy = 0 - Fn_Mag
print("      = ", gx, "i + ", gy, "j")
print("Normal Force: ", Fn_Mag, "N [", math.degrees(Fn_Dir), "]")
nx = 0
ny = Fn_Mag
print("      = ", nx, "i + ", ny, "j")
ax = gx
ay = 0
print("Fg + Fn = ", ax, "i + ", ay, "j")
if Ff_Mag == 0:
  Acceleration = Fa_Mag / Mass
  print("There is no friction. The object slides down.")
elif Slope_Angle == 0:
  Acceleration = 0
  print("The object is on a flat surface. It is stationary.")
else:
  print("Maximum Force of Friction: ", Ff_Mag, "N [", math.degrees(Ff_Dir), "]")
  fx = Ff_Mag
  fy = 0
  print("      = ", fx, "i + ", fy, "j")
  if Ff_Mag > Fa_Mag:
    print("Net Force: 0 N")
    print("      = ", "0", "i + ", "0", "j")
    Acceleration = 0
    print("The object does not overcome the force of friction, and remains stationary.")
    print("To overcome the force of friction, the ramp would need to be at an angle of at least", math.degrees(math.atan(mu)), "degrees.")
  else:
    print("Net Force: ", Fa_Mag - Ff_Mag, "N [", math.degrees(Fa_Dir), "]")
    Acceleration = (Fa_Mag - Ff_Mag) / Mass
    print("The object overcomes the force of friction and slides down the ramp.")
print("Total Acceleration", Acceleration, "m/s^2 [", math.degrees(Fa_Dir), "]")
```