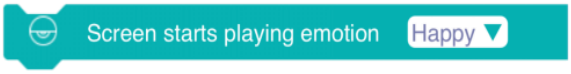
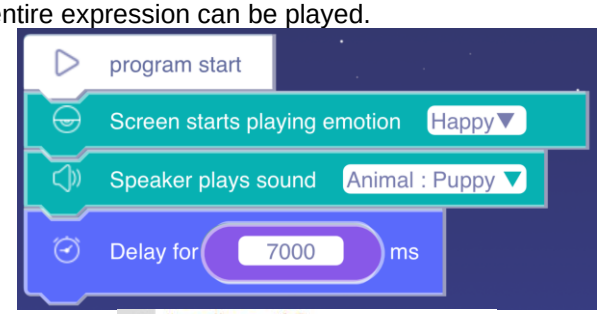
"Programming Blocks" help doc

The "Actuators" Library

1. "Screen Play Emotion" Programming Block

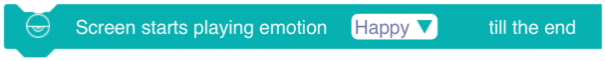


Name: The screen starts playing emotion
Function: The screen displays the built-in animated expressions of the system. When this programming block is executed, the screen starts playing the emotion, and then continues with the following procedure.
Parameters: Option box drop-down list currently has 6 options, name: happy, excited, curious, shy, angry, sad. Each animated expression length: about 6.5s.
Example: The robot barks while playing a happy emotion on the screen.
Note: In the program below the programming block is generally used with the "delay for" module, so that the entire expression can be played.




```
1  # coding=utf-8
2  import random
3  import sys
4  sys.path.append('/userdata/')
5  from clicbot import *
6  robot = Clicbot()
7  #program start
8  robot.act_face(0, 0)
9  robot.act_voice(10, 0)
10 sleep((7000))
```

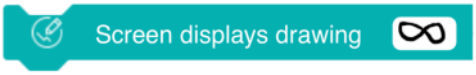
2. "Screen starts playing emotion till the end" Programming Block



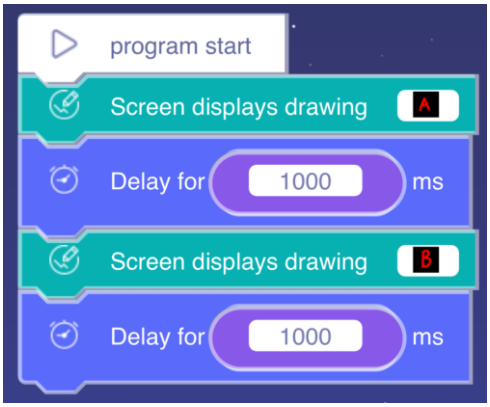
Name: Screen starts playing emotion till the end
Function: The screen displays the built-in animated expressions of the system. The screen starts to play the expressions. After the expression is played, continue to execute the following program.
Parameters: The option box drop-down list currently has 6 options, name: happy, excited, curious, shy, angry, and sad. The length of each animated expression: about 6.5s.
Example: The screen plays a happy expression, and after the expression is played, a dog bark.
Note: The last "Delay for" programming block is to complete "Dog Barking".


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_face(0, 1)
robot.act_voice(10, 0)
sleep((7000))
```

3. "Screen displays drawing" Programming Block

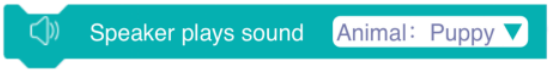


Name: Screen displays drawing
Function: The screen displays drawing. When this programming block is playing, the screen displays drawing, and then continues with the following program. If the following program does not consist other expressions, the drawing will be displayed until the end of the program.
How to use: Click on its option box to open the artboard, manually draw the picture. Below the artboard box, you can select brushes of different colors and lines of different sizes; above the artboard box, you can select Clear All and Return to the Previous step.
Example: The screen displays " Letter A" for 1 second, and then displays " Letter B" for 1 second, and the program ends.

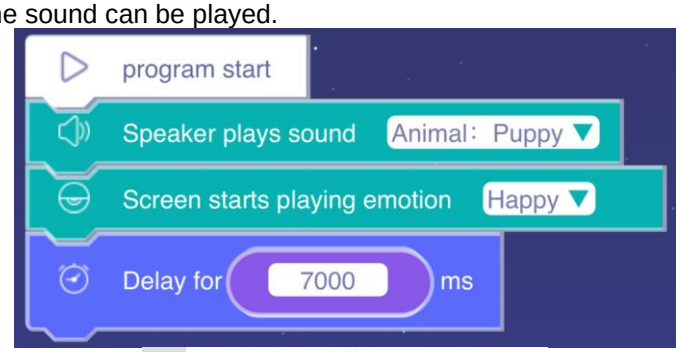


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_svg("../../source_tmp/1641352639.jpg")
sleep((1000))
robot.act_svg("../../source_tmp/1641352676.jpg")
sleep((1000))
```

4. "Speaker plays sound" Programming Block

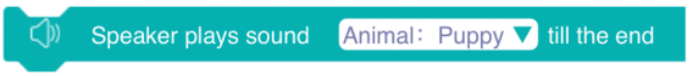


Name: Speaker plays sound
Function: Play system built-in sound. When this programming block is executed, the brain starts playing the sound and continues to execute the following program after it starts playing the sound.
Parameters: The drop-down list of this programming block has a total of 2 category options, name: animal sound and special effects sound. Animal sounds, 6 sounds: a sound of a dog, cat, chicken, pig, cow and sheep; There are 6 special effect sounds: start, brake, object detection, standby, alarm and task completion sound.
Example: The program playing a happy emotion while the dog barking.
Note: In the program below, this programming block is generally used with the "Delay for" module, so that the sound can be played.




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_voice(10, 0)
robot.act_face(0, 0)
sleep((7000))
```

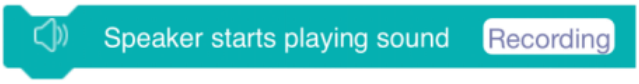
5. "Speakers plays sound till the end" Programming Block



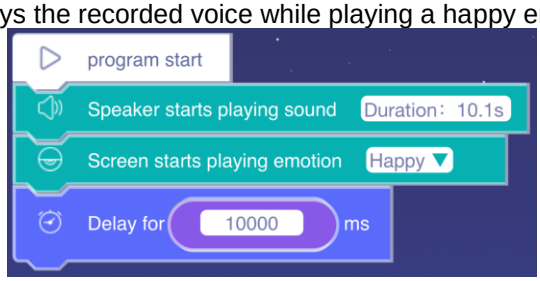
Name: Speakers plays sound till the end
Function: Play built-in sound. When this programming block is executed, the brain starts playing the sound, and continues to execute the following program after the sound playback ends.
Parameters: There are 2 category options in the drop-down list of this programming block: Animal sounds and special effect sounds.
Example: The speaker plays a dog barking sound. After the barking is over, the screen starts to play a happy expression. After the expression is played, the whole program ends.


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_voice(10, 1)
robot.act_face(0, 1)
```

6. "Speaker starts playing sound" Programming Block

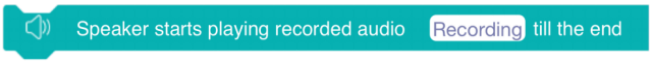


Name: Speaker starts playing sound
Function: Play recorded sound. When executing this programming block, program start playing the recorded sound, then continue with the following procedure.
How to use: Click the option box, in the pop-up window, press and hold the microphone button to record the sound, and release the button after the recording is over.
Parameters: This programming block has a built-in microphone button, which can record sound for about 10 seconds.
Example: The brain plays the recorded voice while playing a happy emotion.

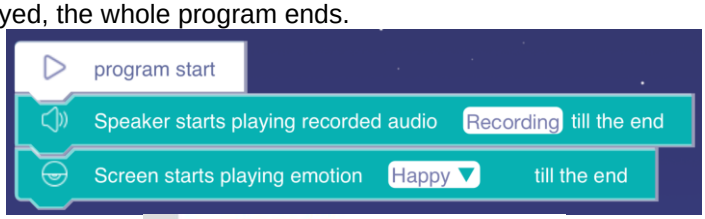


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_transcribe_voice("1641356483.aac", 0)
robot.act_face(0, 0)
sleep((10000))
```

7. "Speaker starts playing recorded audio till the end" Programming Block

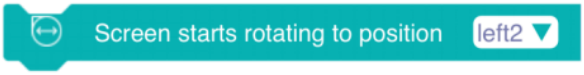


Name: Speaker starts playing recorded audio till the end
Function: Play the recorded sound. When executing this programming block, program start playing the recorded sound, and after the sound playback ends, execute the following program.
How to use: Click the option box, in the pop-up window, press and hold the microphone button to record the sound, and release the button after the recording is over.
Parameters: This programming block has a built-in microphone button, which can record sound for about 10 seconds.
Example: After the brain plays the recording, the screen plays a happy expression. After the expression is played, the whole program ends.



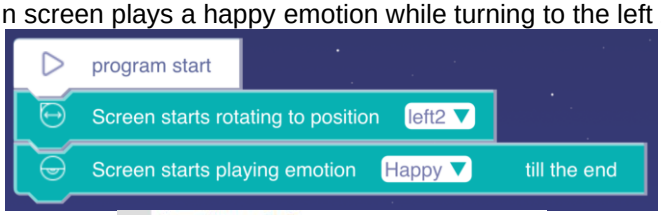

```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_transcribe_voice("-1", 1)
robot.act_face(0, 1)
```

8. "Screen starts rotating to position" Programming Block



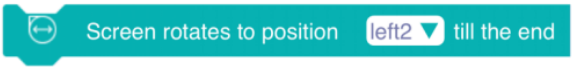
Name: Screen starts rotating to position
Function: It is used to control the brain screen to rotate left or right. After the system executes this programming block, the screen starts to rotate, and then continue to execute the following program.

Parameters: There are 5 options in the drop-down list of this programming block: left 2, left 1, center, right 1, right 2.
Example: The brain screen plays a happy emotion while turning to the left 2 position.

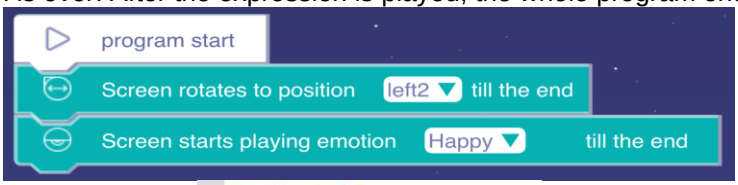



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_transcribe_brain(0, 0)
robot.act_face(0, 1)
```

9. " Screen rotates to position till the end" Programming Block



Name: Screen rotates to position till the end
Function: It is used to control the brain screen to rotate left or right. After the system executes this programming block, the screen starts to rotate. When the screen rotation is over continued to execute the following program.
Parameters: There are 5 options in the drop-down list of this programming block: left 2, left 1, center, right 1, and right 2.
Example: The brain screen is rotated to the left 2 positions, and a happy expression is played after the rotation is over. After the expression is played, the whole program ends.

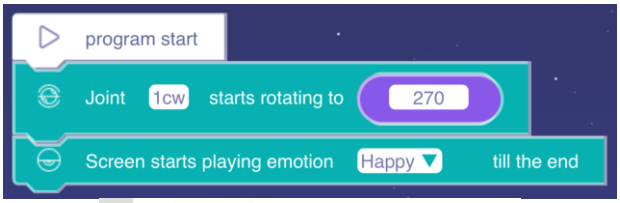



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_transcribe_brain(0, 1)
robot.act_face(0, 1)
```

10. "Joint X starts rotating to" Programming Block

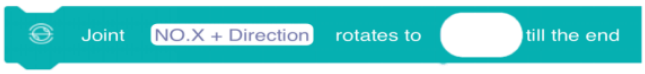
Name: Joint X starts rotating to

Function: It is used to control the joint to rotate a certain angle. After the system executes this programming block, the joint starts to rotate, and then continues to execute the following program.
Parameters: This programming block has two parameters, the first parameter is "select the number of the joint (only one joint can be selected), and set the rotating direction (clockwise, counterclockwise, nearest, cancel rotating) ", the second parameter is "Set the angle value (0-360 degrees) after the joint is rotated".
Example: Joint No. 1 rotates to 270 degrees clockwise, rotates while the screen displays a happy expression.

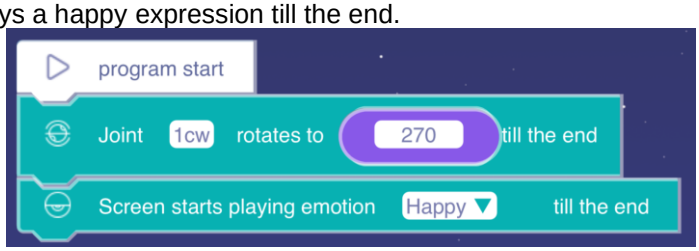



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_joint_servo("1cw", (270), 0)
robot.act_face(0, 1)
```

11. " Joint X rotates to … till the end" Programming Block

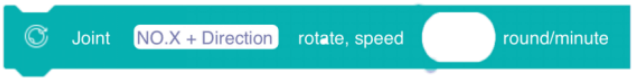


Name: Joint X rotates to … till the end
Function: It is used to control the joint to rotate in a certain angle. After the system executes this programming block, the joint starts to rotate. When the specified angle is reached, continue to execute the following program.
Parameters: This programming block has two parameters, the first parameter is "select the number of joint (only one joint can be selected), and set the rotating direction (clockwise, counterclockwise, nearest, cancel rotating) ", the second parameter is "Set the angle value (0-360 degrees) after the joint is rotated".
Example: No. 1 joint rotates to 270 degrees in a clockwise direction. After the rotation is over, the screen displays a happy expression till the end.

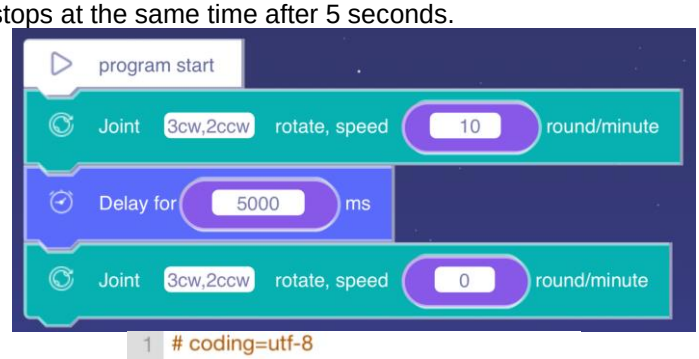


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_joint_servo("1cw", (270), 1)
robot.act_face(0, 1)
```

12. "Joint X rotate，speed … round/minute" Programming Block

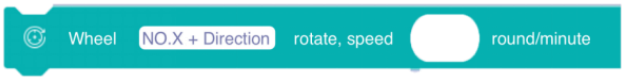


Name: Joint X rotate，speed … round/minute
Function: It is used to control the joint to rotate at different speeds, stop rotating when the speed is 0, and continue to rotate when the speed is not 0.
Parameters: This programming block has two parameters, the first parameter is "select the number of joint (multiple joints can be selected), and set the rotating direction (clockwise, counterclockwise, cancel) ", the second parameter is "Set the speed value at which the joint rotates (0-36 rpm) ".
Example: No. 2 joint rotates counterclockwise, No. 3 joint rotates clockwise at 10 r/min at the same time, and stops at the same time after 5 seconds.




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_joint_rotate("3cw,2ccw", (10))
sleep((5000))
robot.act_joint_rotate("3cw,2ccw", (0))
```

13. "Wheel X rotate, speed … round/minute" Programming Block



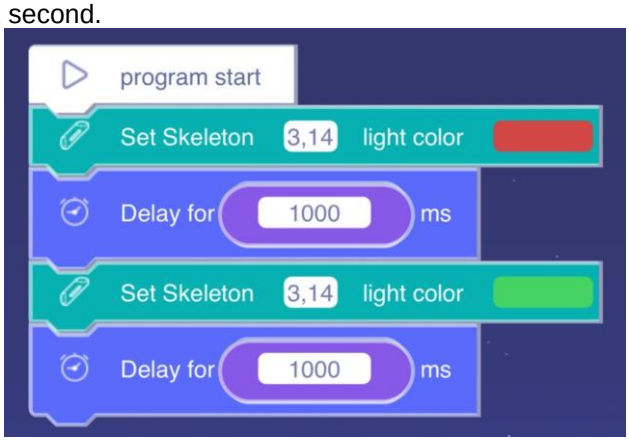
Name: Wheel X rotate, speed … round/minute

Function: It is used to control the wheels to rotate at different speeds, stop rotating when the speed is 0, and continue to rotate when the speed is not 0.
Parameters: This programming block has two parameters, the first parameter is "select the number of the target wheel (multiple wheels can be selected), and set the rotating direction (clockwise, counterclockwise, cancel rotating) ", the second parameter is "Set the speed value of the target wheel rotating (0-270 rpm) ".
Example: Wheels No. 6 and No. 5 rotate clockwise at 30 r/min at the same time, and stop at the same time after 3 seconds.


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_wheel_rotate("6cw,5cw", (30))
sleep((3000))
robot.act_wheel_rotate("6cw,5cw", (0))
```

14. "Set Skeleton X light color" Programming Block

Name: Set Skeleton X light color
Function: It is used to control the extension rod indicator light to display different colors. You can select any indicator light color through the color panel, and the indicator light can be continuously lit.
Parameters: This programming block has two parameters, the first parameter is "Select the number of the target extension rod (multiple extension rods can be selected) ", and the second parameter is "Set the color of the target extension rod (color panel) ".
Example: No. 3 and No. 14 extension bar indicators show red for 1 second at the same time, and then green for 1 second.



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_skele_color("3,14","#d34747")
sleep((1000))
robot.act_skele_color("3,14","#47d363")
sleep((1000))
```

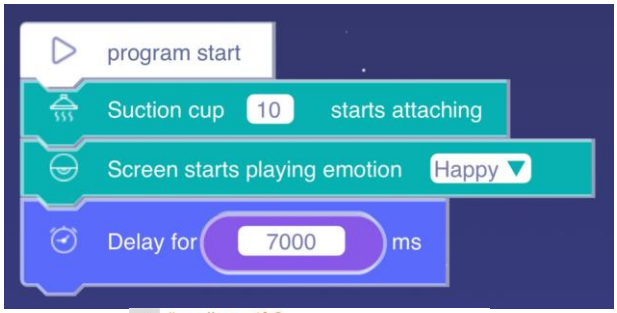
15. "Grasper" Programming Block

Name: Grasper
Function: used to control the grasping of the mechanical claw.
Parameters: This programming block has two parameters, the first parameter is "select the number of the grasper (only one grasper can be selected) ", and the second parameter is "set the state of the grasper (release, grab) ".
Example: No. 18 grasper is released, and the mechanical claw grabs after 2 seconds.


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_machinery("18",0)
sleep((2000))
robot.act_machinery("18",1)
```

16. "Suction cup X starts attaching" Programming Block

Name: Suction cup X starts attaching
Function: Used to control the attaching of the suction cup. When this programming block is executed, the suction cup starts to attaches itself, and then continues to execute the following program (note: the suction cup continues to attaches itself during this process).
Parameters: This programming block has a total of 1 parameter, which is "Select the number of the target suction cup (only one suction cup can be selected) "
Example: No. 10 suction cup starts attaching, the screen displays a happy expression, and the program ends after 7 seconds.



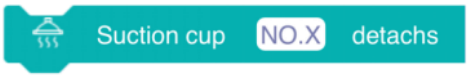

```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_transcribe_open("10", 0)
robot.act_face(0, 0)
sleep((7000))
```

17. "Suction cup attachs till success" Programming Block

Name: Suction cup attachs till success
Function: Used to control the attaching of the suction cup. When this programming block is executed, the suction cup starts to attach itself, and when the attach process is completed, continue to execute the following procedure.
Parameters: This programming block has a total of 1 parameter, which is "Select the number of the target suction cup (only one suction cup can be selected) ".
Example: After the No. 5 suction cup is successfully attached, the screen displays a happy expression till the end.

```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_transcribe_open("5", 1)
robot.act_face(0, 1)
```

18. "Suction cup detachs" Programming Block

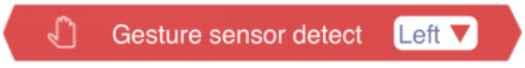


Name: Suction cup detachs
Function: Used to control the detach of the suction cup. When this programming block is executed, the suction cup starts to detach itself, and then the following program continues.
Parameters: This programming block has a total of 1 parameter, which is "Select the number of the target suction cup (only one suction cup can be selected) ".
Example: No. 10 suction cup starts to attach itself, and after 7 seconds, the suction cup is detachs.


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_transcribe_open("10", 0)
sleep((7000))
robot.act_transcribe_close("10")
```

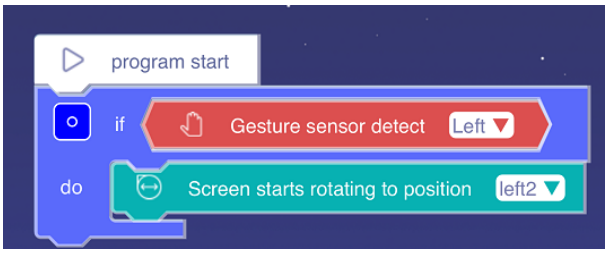
The "Sensors" Library

1. "Gesture sensor detect" Programming Block



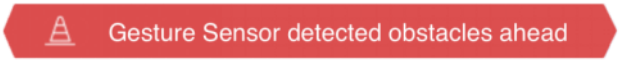
Name: Gesture sensor detect
Function: Used to detect gesture operations.

Parameters: This programming block has a total of 8 parameters, including ("Left", "Right", "Up", "Down", "Circle Clockwise", "Circle Counterclockwise", "Approach", "Away").
Example: When the gesture "Left" is detected, the screen starts to rotate towards position "Left 2".

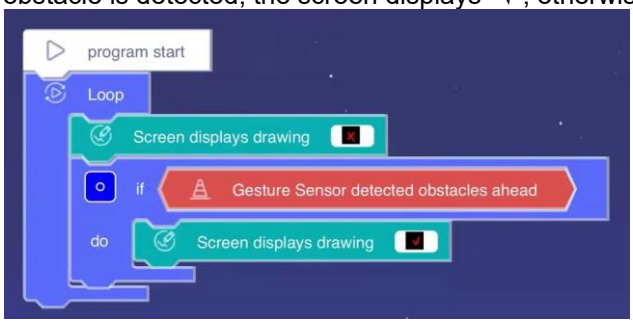



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#开始程序
if robot.percept_figer_location(0):
    robot.act_transcribe_brain(0, 0)
```

2. "Gesture sensor detected obstacles ahead" Programming Block



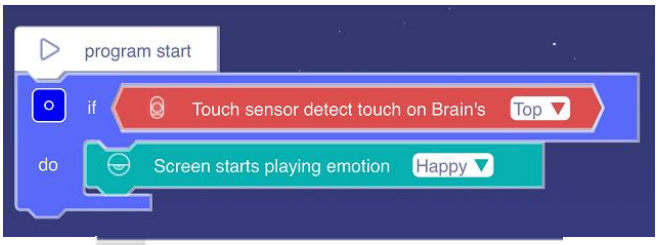
Name: Gesture sensor detected obstacles ahead
Function: This programming module is used to detect whether there is an obstacle in front of the eye.
Parameters: Gesture sensor detection range (0--5cm).
Example: When an obstacle is detected, the screen displays "√", otherwise "×".




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_svg("../../source_tmp/1641458400.jpg")
    if robot.percept_check_obstacle():
        robot.act_svg("../../source_tmp/1641458390.jpg")
```

3. "Touch sensor detect touch on Brain’s X" Programming Block

Name: Touch sensor detect touch on Brain’s X
Function: Used to detect whether the Brain is touched.
Parameters: This programming block has three parameters "top", "left", "right".
Example: When touching the top, the screen displays the expression "Happy".




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
if robot.percept_check_capacitance(0):
    robot.act_face(0, 0)
```

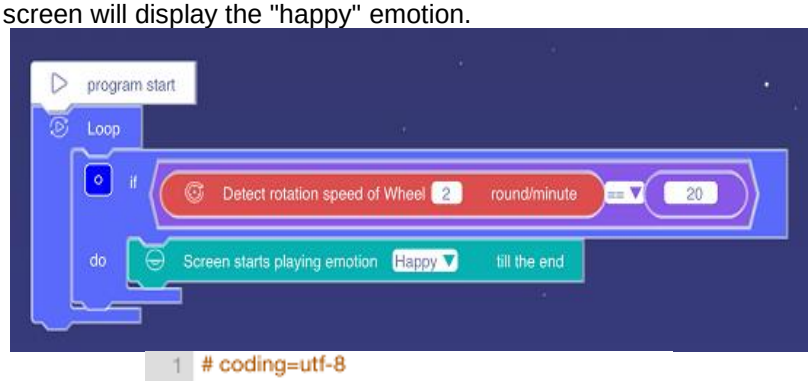
4. "Touch screen detect X" Programming Block

Name: Touch screen detect X
Function: This module monitors whether there are operation instructions on the brain screen during program operation.
Parameters: This programming block has two parameters, "click" and "slide".
Example: When the screen is tapped, the screen displays the "happy" emotion.

```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
if robot.percept_check_brain(0):
    robot.act_face(0, 1)
```

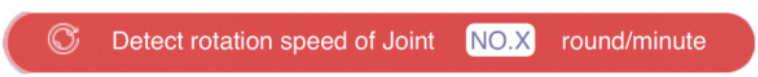
5. "Detect rotation speed of Wheel X round/minute" Programming Block

Name: Detect rotation speed of Wheel X round/minute
Function: When the wheel is turned, this programming block will monitor rotation speed of the wheel.
Parameters: This programming block detects the wheel speed range (0~ 270r/min).
Example: This programming block will detect wheel 2, if the speed of wheel 2 reaches 20r/min, the screen will display the "happy" emotion.



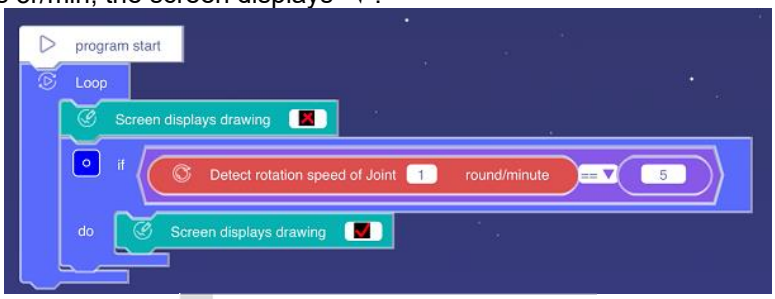

```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    if (robot.percept_wheel_rotate(2)) == (20):
        robot.act_face(0, 1)
```

6. "Detect rotation speed of Joint X round/minute" Programming Block



Name: Detect rotation speed of Joint X round/minute

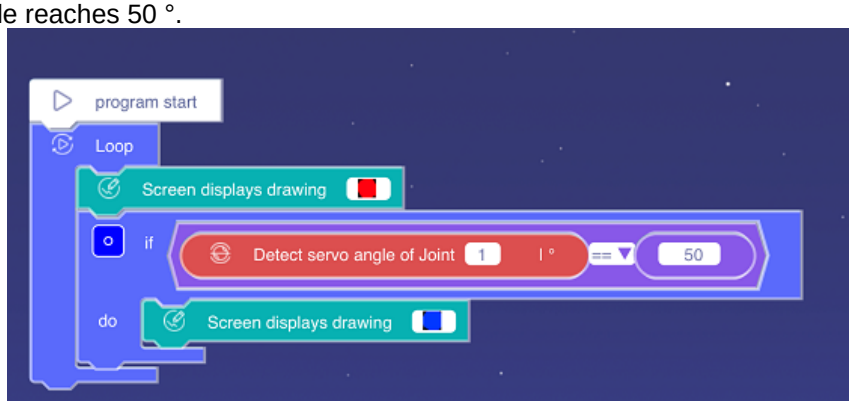
Function: This programming block is used to monitor the rotating speed of the joint as it rotates.
Parameters: This programming block detects the joint speed range (0~ 36r/min).
Example: When the program starts, the screen displays "×", and when it detect the joint 1 speed reaches 5r/min, the screen displays "√".




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_svg("../../source_tmp/1641524198.jpg")
    if (robot.percept_joint_rotate(1)) == (5):
        robot.act_svg("../../source_tmp/1641524210.jpg")
```

7. "Detect servo angle of Joint X" Programming Block

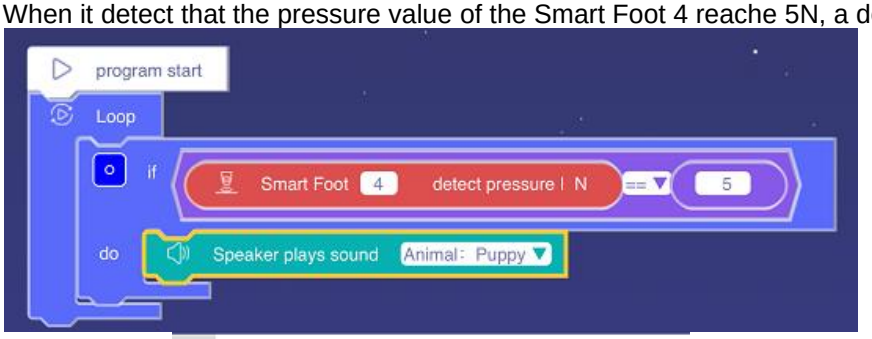
Name: Detect servo angle of Joint X
Function: This programming block is used to detect the current rotation angle of the joint.
Parameters: This programming block detects the joint angle range (0~ 360 °).
Example: The program starts with "red" on the screen, and "blue" when it detects that the joint 1 angle reaches 50 °.



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_svg("../../source_tmp/1641536830.jpg")
    if (robot.percept_joint_servo(1)) == (50):
        robot.act_svg("../../source_tmp/1641536849.jpg")
```

8. "Smart Foot X detect pressure" Programming Block

Name: Smart Foot X detect pressure
Function: This programming block is used to detect the pressure value of the Smart Foot.
Parameters: The programming block detects the pressure through the pressure sensor on the sole of the foot, and the detection range of the pressure sensor (0~ 20N, that is, the maximum is about 2kg).
Example: When it detect that the pressure value of the Smart Foot 4 reache 5N, a dog barks.

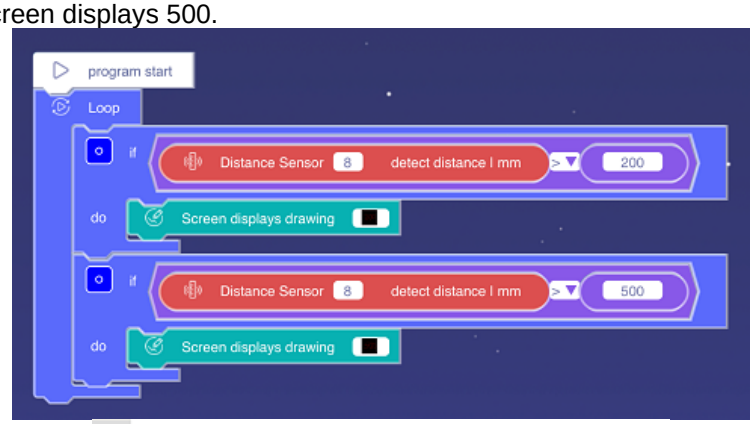



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    if (robot.percept_pressure_transducer(4)) == (5):
        robot.act_voice(10, 0)
```

9. "Distance Sensor X detect distance mm" Programming Block

Name: Distance Sensor X detect distance mm

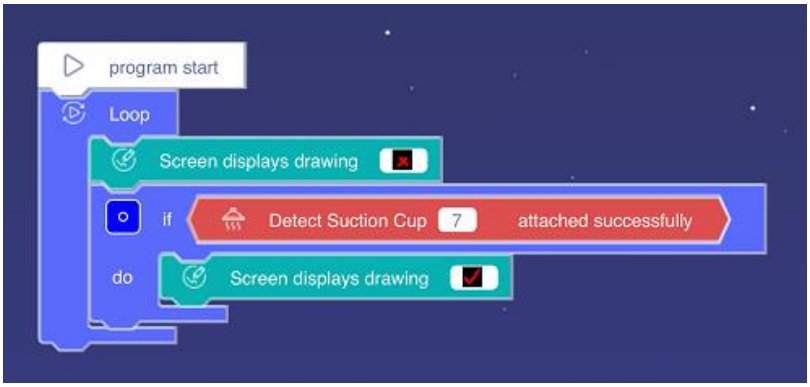
Function: This programming block is used to detect the distance between the object and the sensor.
Parameters: This programming block detects the distance between the object and the sensor through the distance sensor, and the sensor detection range (20~ 1000mm) (1cm = 10mm).
Example: When the distance sensor detects the object distance is more than 200mm, the screen displays 200, and when the distance sensor detects the object distance is more than 500mm, the screen displays 500.




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    if (robot.distance_sensor(8)) > (200):
        robot.act_svg("../../source_tmp/1641539755.jpg")
    if (robot.distance_sensor(8)) > (500):
        robot.act_svg("../../source_tmp/1641539763.jpg")
```

10. "Detect Suction Cup X attached successfully" Programming Block

Name: Detect Suction Cup X attached successfully
Function: Use to detect whether the Suction Cup attach is complete.
Parameters: none
Example: When the program starts, before the suction cup is successfully attached, the screen displays "X", and when the suction cup is successfully detached, the screen displays "√".

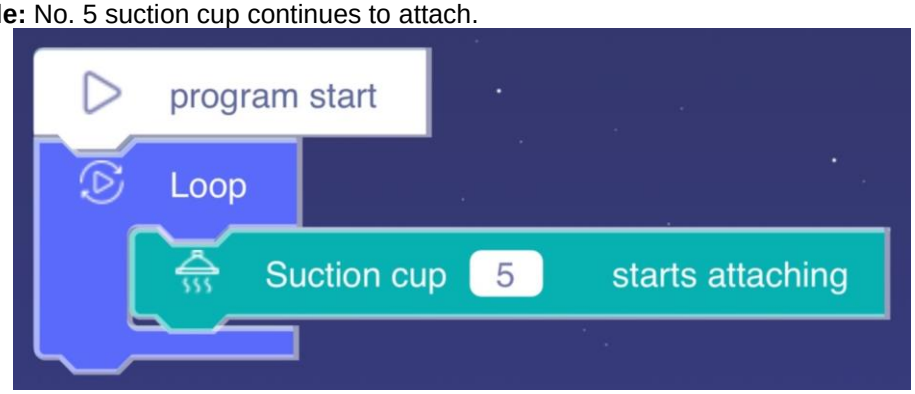


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_svg("../../source_tmp/1641543559.jpg")
    if robot.distance_sucker_adsorbent(7):
        robot.act_svg("../../source_tmp/1641543566.jpg")
```

The "Controls" Library

1. "Loop" Programming Block

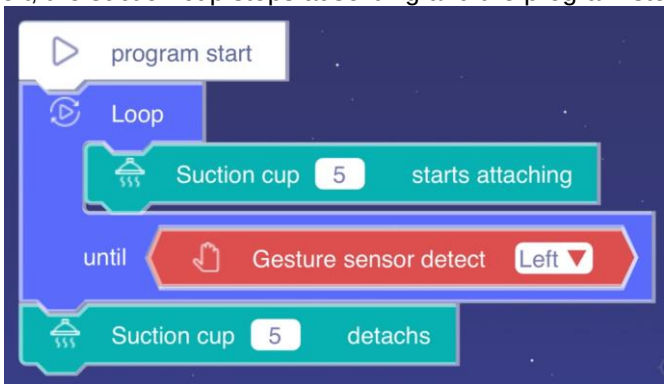
Name: Loop
Function: This programming block is used for the loop structure (while-type loop) in programming, which can realize the repeated execution of part or all parts of the program.
Parameters: None.
Example: No. 5 suction cup continues to attach.



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_transcribe_open("5", 0)
```

2. "Loop until" Programming Block

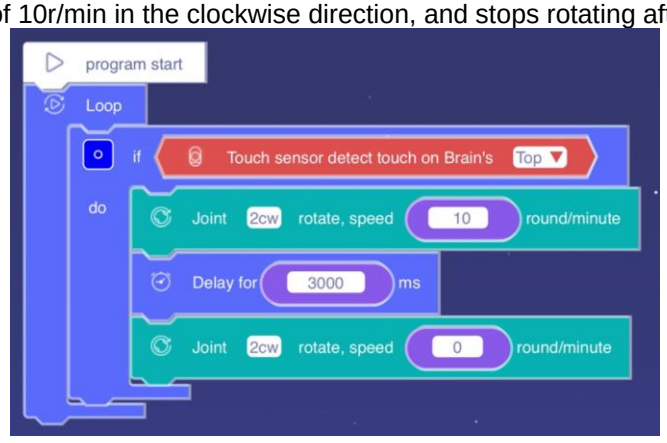
Name: Loop until
Function: This programming block is a loop structure (while-type loop) with its own conditional judgment, which can realize the repeated execution of some or all programs. When the program executes to this programming block, the loop action is executed first, and then the conditional judgment is performed. When the condition is complete, the loop is terminated, otherwise the loop continues.
Parameters: This programming block has 1 parameter; the hexagon box is the judgment condition. The conditional expression can be varied, but the result is only True and False. (Note: Hexagons represent -- expressions.)
Example: When the program starts, the suction cup starts to attach. When the screen detects the gesture to the left, the suction cup stops adsorbing and the program stops.



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_transcribe_open("5", 0)
    if((robot.percept_figer_location(0))):
        break
robot.act_transcribe_close("5")
```

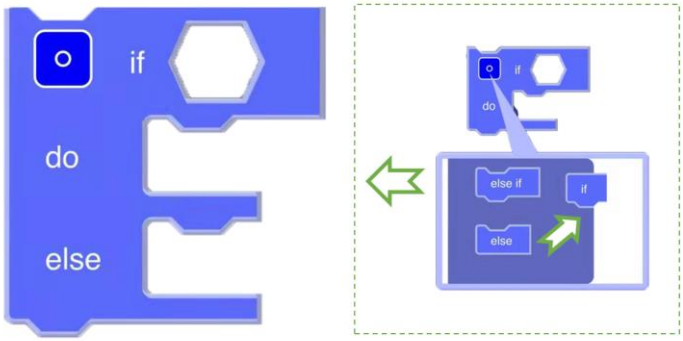
3. "If-do" Programming Block

Name: If-do
Function: This programming block is an if-type single-selection structure, which can realize conditional judgment. When the program executes to this programming block, the conditional judgment is performed first, and when the condition is complete, the rest of the program is executed; otherwise, it is not executed.
Parameters: This programming block has 1 parameter, that is, the hexagon box is the judgment condition. The conditional expression can be varied, but the result is only True and False. (Note: Hexagons represent -- expressions.)
Example: The program starts to loop. If touch the top of the brain, the No. 2 joint starts to rotate at a speed of 10r/min in the clockwise direction, and stops rotating after 3 seconds.

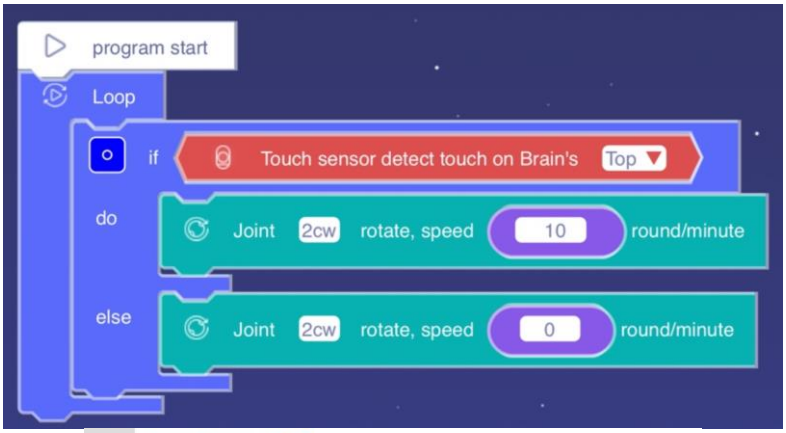


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    if robot.percept_check_capacitance(0):
        robot.act_joint_rotate("2cw", (10))
        sleep((3000))
        robot.act_joint_rotate("2cw", (0))
```

4. " If-do-else" Programming Block

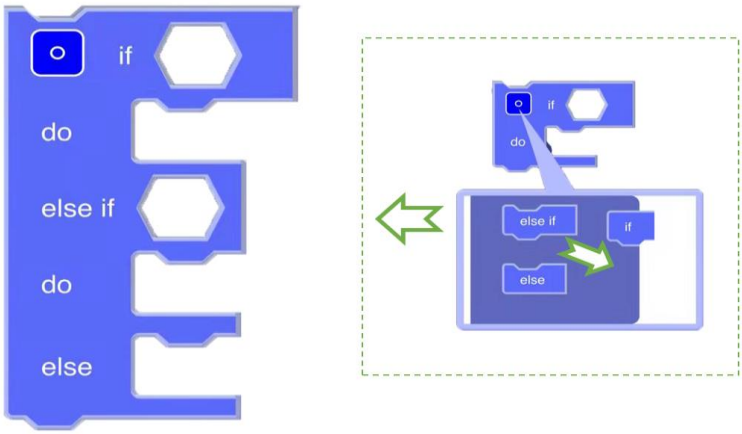


Name: if-do-else
Function: This programming block is an if-else type double-selection structure, which can realize conditional judgment. When the program executes to this programming block, first perform conditional judgment, and when the condition is complete, execute the upper "executor"; otherwise, execute the lower "executor".
Parameters: This programming block has 1 parameter; the hexagon box is the judgment condition. The conditional expression can be varied, but the result is only True and False. (Note: Hexagons represent -- expressions.)
Example: The program starts the loop. When stroking the top of the brain, the No. 2 joint starts to rotate at a speed of 10r/min in the clockwise direction; otherwise, the joint stop rotating.



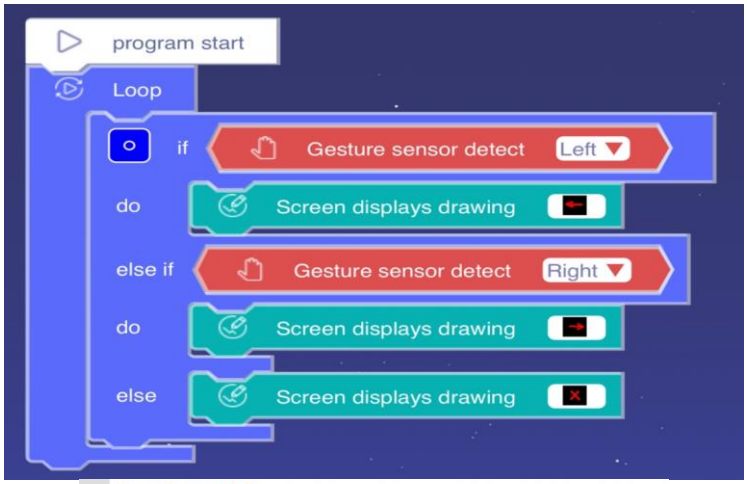

```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    if robot.percept_check_capacitance(0):
        robot.act_joint_rotate("2cw", (10))
    else:
        robot.act_joint_rotate("2cw", (0))
```

5. "If - do – else if – do - else" Programming Block



Name: If - do – else if – do - else
Function: This programming block is an if-else type multi-selection structure, which can realize multi-condition judgment. When the program executes this programming block, the first conditional judgment is performed first, and when the condition is complete, the first line of "executor" is executed; otherwise, the second conditional judgment is continued. When the condition is complete, the second line of "executor" is executed. Otherwise, the third line of "executor" is executed.

Parameters: This programming block can have multiple parameters; the hexagon box is the judgment condition, and the conditional expression can be various, but the result is only True and False. (Note: Hexagons represent -- expressions.)
Example: The program starts to loop. When the screen detects a gesture to the left, the screen displays a left arrow; otherwise, when the screen detects a gesture to the right, the screen displays a right arrow; otherwise, the screen displays a cross.




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    if robot.percept_figer_location(0):
        robot.act_svg("../../source_tmp/1641463059.jpg")
    elif robot.percept_figer_location(1):
        robot.act_svg("../../source_tmp/1641463069.jpg")
    else:
        robot.act_svg("../../source_tmp/1641463085.jpg")
```

6. "While - do" Programming Block

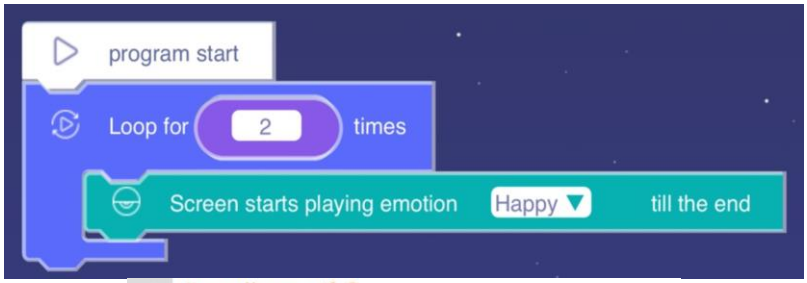
Name: While - do
Function: This programming block is a loop structure (while-type loop) with its own conditional judgment, which can realize the repeated execution of some or all programs.

When the program executes to this programming block, the conditional judgment is performed first. When the condition is complete, it enters the loop and executes the loop body; otherwise, it does not enter the loop.
Parameters: This programming block has 1 parameter; the hexagon box is the judgment condition. The conditional expression can be varied, but the result is only True (true) and False (false). (Note: Hexagons represent -- expressions.)
Example: When the program starts, the loop starts. When the distance sensor detects that the distance from the obstacle is greater than 20 cm, the screen loops to play the happy animation expression.


```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    while ((robot.distance_sensor(17)) > (200)):
        robot.act_face(0, 1)
```

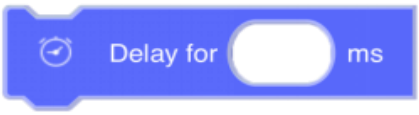
7. "Loop for ( ) times" Programming Block

Name: Loop for ( ) times
Function: This programming block is a loop structure (for-type loop) with its own conditional judgment, which can realize the repeated execution of some or all programs. When the program executes to this programming block, the conditional judgment is performed first. When the condition is not established, it enters the loop and executes the loop body; otherwise, it does not enter the loop.
Parameters: This programming block has 1 parameter; the oval box is the judgment condition, which is a specific numerical value, and the results are True and False. (Note: The oval represents -- a numerical value.)
Example: When the program starts, the screen loops the happy animated expression 2 times.

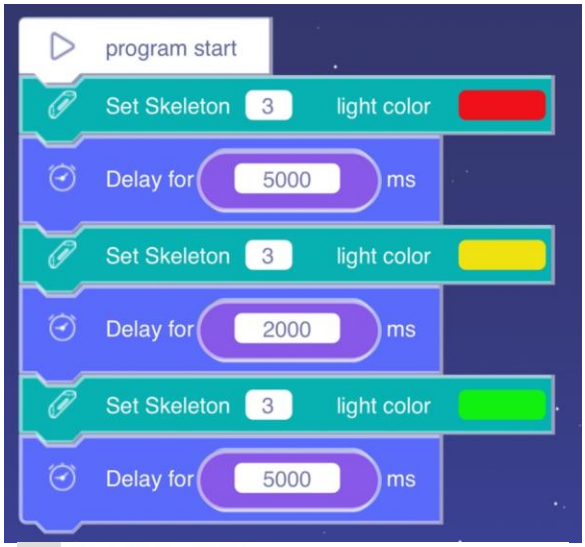



```
1 # coding=utf-8
2 import random
3 import sys
4 sys.path.append('/userdata/')
5 from clicbot import *
6 robot = Clicbot()
7 #program start
8 for i in range((2)):
9     robot.act_face(0, 1)
```

8. "Delay for ( ) ms" Programming Block

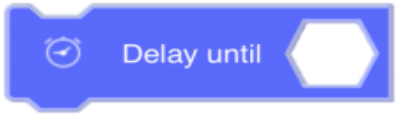


Name: Delay for ( ) ms
Function: When the program executes to this block, it needs to stay there for the corresponding time, and when the waiting time is reached, it continues to execute the following program.
Parameters: This programming block has 1 parameter, which is the time value in ms, that is, 1000ms = 1s. (Note: The oval represents -- the value.)
Example: After the program starts, the No. 3 skeleton displays a red light for 5 seconds, then a yellow light for 2 seconds, and finally a green light for 5 seconds.

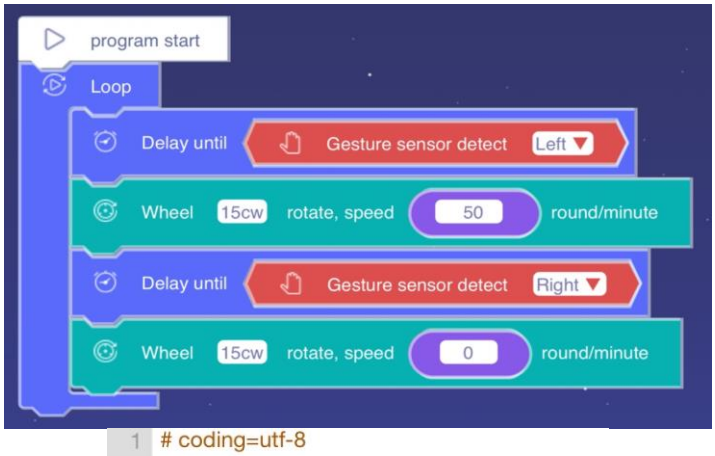



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
robot.act_skele_color("3","#f01019")
sleep((5000))
robot.act_skele_color("3","#f0e210")
sleep((2000))
robot.act_skele_color("3","#10f011")
sleep((5000))
```

9. "Delay until" Programming Block

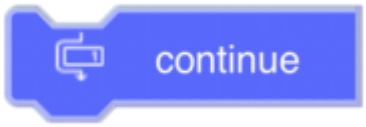


Name: Delay until
Function: When the program executes to this programming block, it is necessary to repeat the judgment until the condition is complete and then continue the execution of the following program.
Parameters: This programming block has 1 parameter, that is, the hexagon box is the judgment condition. The conditional expression can be varied, but the result is only True and False. (Note: Hexagons represent -- expressions.)
Example: The program starts, waiting for the screen to detect a gesture to the left, wheel 15 turns clockwise at 50r/min, and then continues to wait for the wheel to stop turning when the screen detects a gesture to the right.

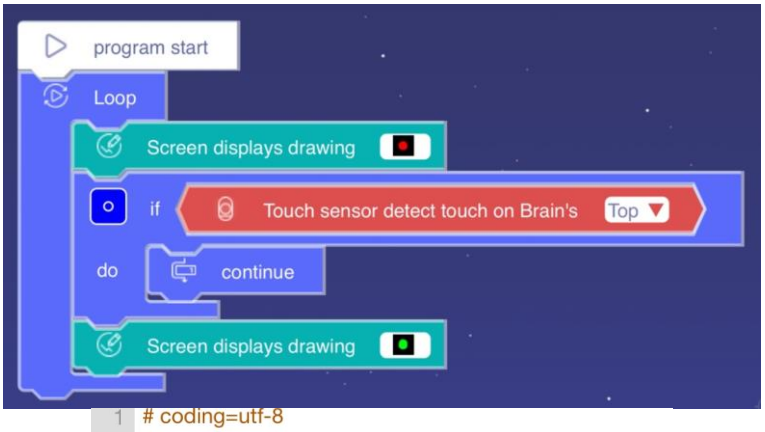



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    while True:
        if((robot.percept_figer_location(0))):
            break
    robot.act_wheel_rotate("15cw", (50))
    while True:
        if((robot.percept_figer_location(1))):
            break
    robot.act_wheel_rotate("15cw", (0))
```

10. "Continue" Programming Block



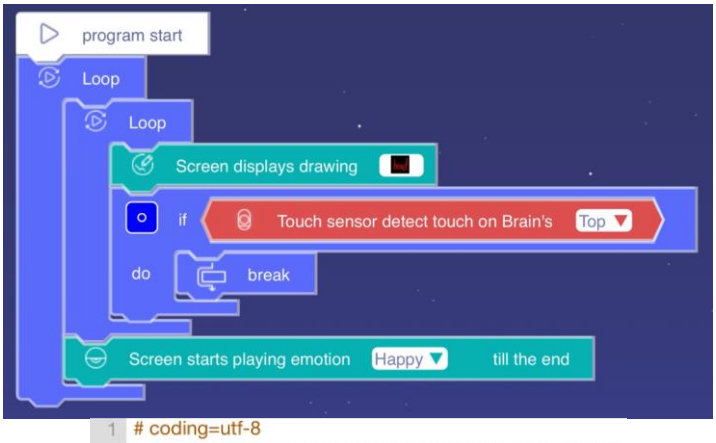
Name: Continue
Function: Commonly used in while and for loops, when the program executes to this programming block, it can skip the remaining statements of the current loop and continue to the next loop.
Parameters: none
Example: At the beginning of the program, the screen alternately displays "red circle" and "green circle", and when stroking the top of the brain, the screen only displays "red circle".




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_svg("../../source_tmp/1641524424.jpg")
    if robot.percept_check_capacitance(0):
        continue
    robot.act_svg("../../source_tmp/1641524485.jpg")
```

11. "Break" Programming Block

Name: Break
Function: It can terminate the loop statement, which is often used in while and for loops. When the program executes to this programming block, it terminates the outer loop closest to the break statement, and then continues to execute other programs.
Parameters: none
Example: When the program starts, the screen loops to display the word "head". When stroking the top of the brain, it jumps out of the loop and displays the happy expression once, and then the program repeats.

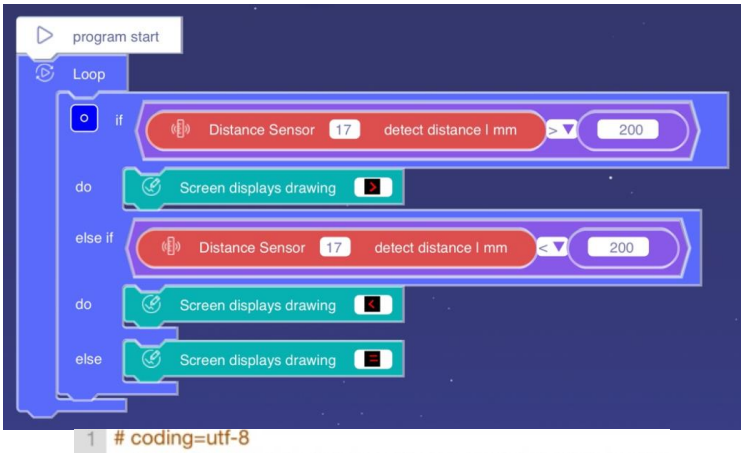



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    while True:
        robot.act_svg("../../source_tmp/1641524485.jpg")
        if robot.percept_check_capacitance(0):
            break
    robot.act_face(0, 1)
```

The "Operators" Library

1. "Relational Operators" Programming Block

Name: Relational Operator
Function: Also called the comparison operator, it is used to determine the magnitude of two values. The result is of Boolean type, i.e. True when the operator corresponds to a relationship, otherwise False.
Parameters: Two numerical parameters in this programming block, the operator contains ”>“、”<“、”=“、 ”≠“、”≥“、”≤“。（Note: The ellipse represents -- the numerical value.)
Example: When the distance sensor detects a distance greater than 20 cm, the screen displays ">"; when the distance sensor detects a distance less than 20 cm, the screen displays "<"; otherwise, the screen displays "=".



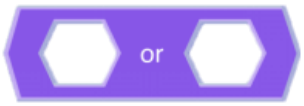

```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    if (robot.distance_sensor(17)) > (200):
        robot.act_svg("../../source_tmp/1641527883.jpg")
    elif (robot.distance_sensor(17)) < (200):
        robot.act_svg("../../source_tmp/1641527853.jpg")
    else:
        robot.act_svg("../../source_tmp/1641527909.jpg")
```

2. "Arithmetic Operators" Programming Block

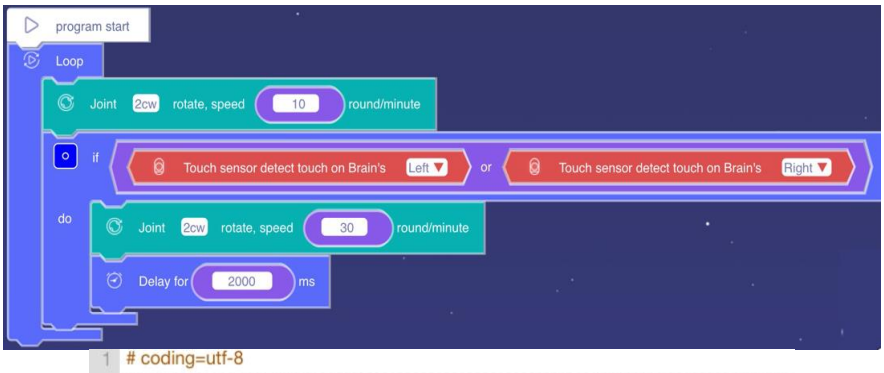
Name: Arithmetic Operator
Function: A symbol used to perform basic arithmetic operations.
Parameters: Two numerical parameters in this programming block, the operator contains ”+“、”-“、”*“、”/“。（Note: The ellipse represents --the numerical value.)
Example: The program starts, the program loops 10 times, wheel No. 15 follows the clockwise direction, the speed starts from 0 r/min, accelerates and rotates in the form of adding 27 to each speed value, and each rotation lasts for 2 seconds.

```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
for i in range((10)):
    robot.act_wheel_rotate("15cw", (robot.percept_wheel_rotate(15) + 27))
    sleep((2000))
```

3. "Logical operator _or_ '" Programming Block



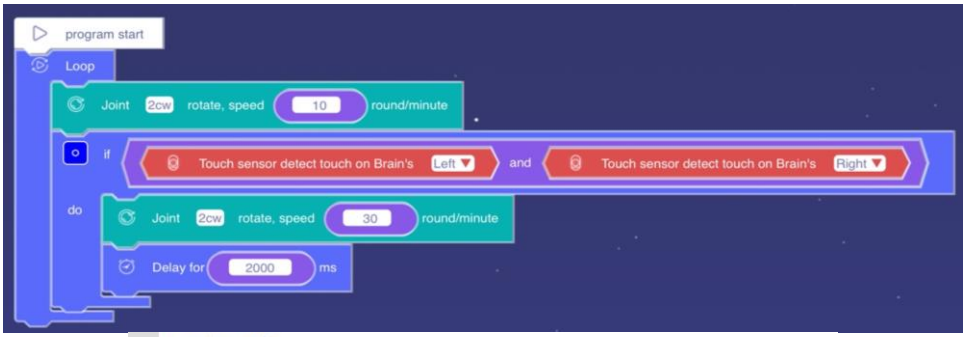
Name: Logical operator "or"
Function: The logical operator "or" is equivalent to "or" in daily communication. The hexagons on both sides of "or" are two judgment conditions. If either of the two judgment conditions is true, the result of the operator is True; if neither judgment condition is true, the result of the operator is False.
Parameters: Two conditional expressions in this programming block. (Note: Hexagons represent -- expressions.)
Example: At the beginning of the program, the No. 2 joint rotates clockwise at a speed of 10 r/min. When stroking the left or right side of the brain, the joint rotates at a speed of 30 r/min for 2 seconds.




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_joint_rotate("2cw", (10))
    if (robot.percept_check_capacitance(1)) or (robot.percept_check_capacitance(2)):
        robot.act_joint_rotate("2cw", (30))
        sleep((2000))
```

4. "Logical Operators" 'AND' "Programming Block

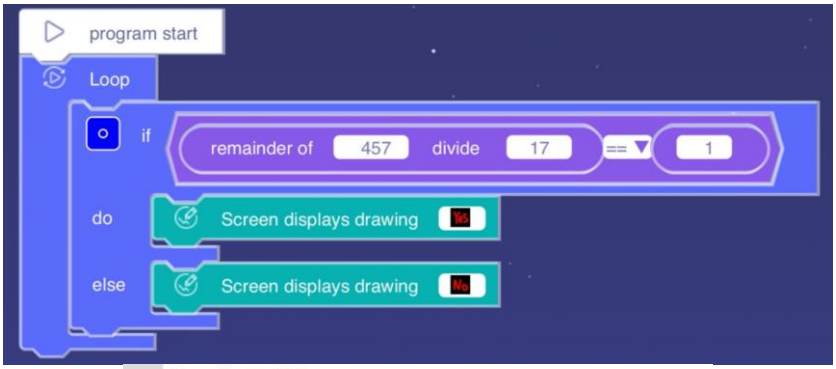
Name: Logical operator "and"
Function: The logical operator "AND" is equivalent to "AND" in daily communication. The hexagons on both sides of "AND" are two judgment conditions. If the two judgment conditions are established at the same time, the result of the operator is True; if either of the two judgment conditions is not established, the result of the operator is False.
Parameters: Two conditional expressions in this programming block. (Note: Hexagons represent -- expressions.)
Example: At the beginning of the program, the No. 2 joint rotates clockwise at a speed of 10 r/min. When the left and right sides of the brain are stroked at the same time, the joint rotates at a speed of 30 r/min for 2 seconds.




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_joint_rotate("2cw", (10))
    if (robot.percept_check_capacitance(1)) and (robot.percept_check_capacitance(2)):
        robot.act_joint_rotate("2cw", (30))
        sleep((2000))
```

5. "Remainder of" Programming Block

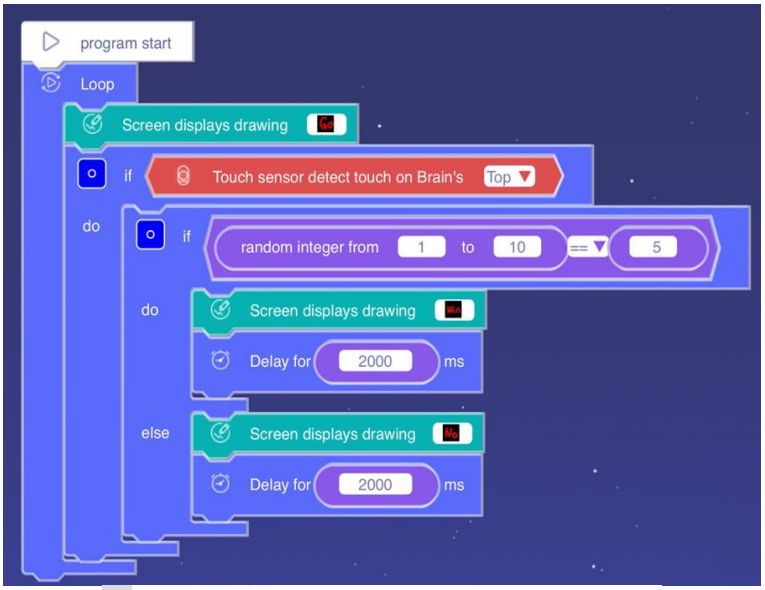
Name: Remainder of
Function: One of the arithmetic operators used to take the remainder of two numbers.
Parameters: This programming block has two values, the options have built-in numbers panel, from 0-9.
Example: Loop to determine whether the remainder of 457 divided by 17 is equal to 1, if it is equal to 1, the screen displays "Yes", otherwise, displays "No".




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    if (457%17) == (1):
        robot.act_svg("../../source_tmp/1641537202.jpg")
    else:
        robot.act_svg("../../source_tmp/1641537189.jpg")
```

6. "Random Integer from" Programming Block

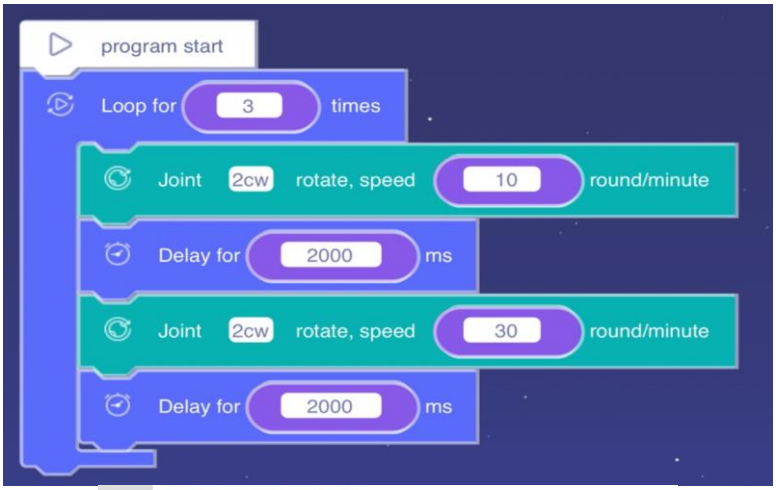
Name: random integer
Function: Used to generate a random number within the specified range, the range of the interval does not include the following numbers. For example: the random number between (1, 3) has 1 and 2, excluding 3.
Parameters: This programming block has two values, the options have built-in numbers panel, from 0-9.
Example: (lottery game) The program starts, the screen displays "Go", when stroking the top of the brain, the system takes a random number between 1 and 9, if the random number is 5, the screen displays "Win" for 2 seconds; otherwise, the screen displays "No" for 2 seconds.




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    robot.act_svg("../../source_tmp/1641537837.jpg")
    if robot.percept_check_capacitance(0):
        if (random.randrange(1,10)) == (5):
            robot.act_svg("../../source_tmp/1641537736.jpg")
            sleep((2000))
        else:
            robot.act_svg("../../source_tmp/1641537189.jpg")
            sleep((2000))
```

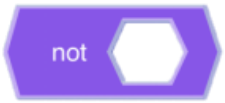
7. "Numerical" Programming Block

Name: Value
Function: Customizable value.
Parameters: This programming block has a built-in digital panel, from 0-9.
Example: At the beginning of the program, the No. 2 joint rotates clockwise at 10 r/min for 2 seconds, and then rotates at 30 r/min for 2 seconds; repeat 3 times.

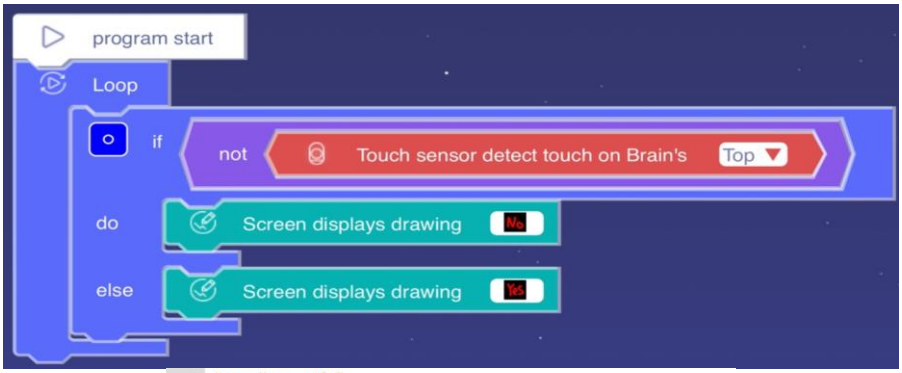



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
for i in range((3)):
    robot.act_joint_rotate("2cw", (10))
    sleep((2000))
    robot.act_joint_rotate("2cw", (30))
    sleep((2000))
```

8. "Logical operator NOT " Programming Block



Name: Logical operator "NOT"
Function: The logical operator "NOT" refers to the inverse of the original value. The hexagon on the right side of "not" is the judgment condition. If the judgment condition is true, the value of the operator is False. If the judgment condition is not true, the value of the operator is True.
Parameters: This programming block has a conditional expression. (Note: Hexagons represent -- expressions.)
Example: The program starts and judges that if the brain head is touched, the screen displays "Yes", and if the brain head is not touched, the screen displays "No".

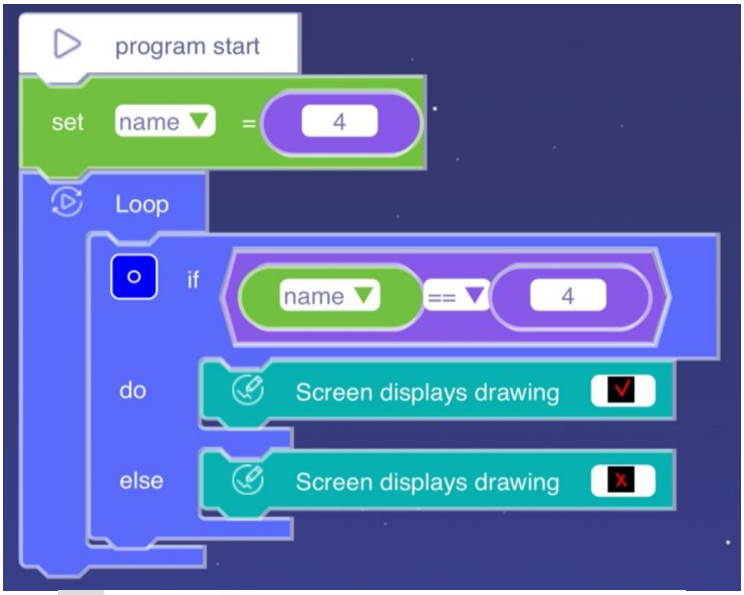



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
while True:
    if not((robot.percept_check_capacitance(0))):
        robot.act_svg("../../source_tmp/1641537189.jpg")
    else:
        robot.act_svg("../../source_tmp/1641537202.jpg")
```

The "Variable" Library

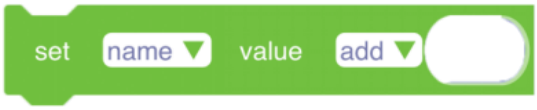
1. "Variable Assignment" Programming Block

Name: variable assignment
Function: Variables are quantities that can be changed and can be created independently. When using it for the first time, you need to assign a value to the variable.
Parameters: This programming block has 2 parameters, the first parameter is "name represents the variable name", and the name of the created variable can be selected in the drop-down list. The second parameter is a specific value. (Note: The ellipse represents -- a value.)
Example: Loop judgment, if the variable name is equal to 4, the screen displays a checkmark, otherwise a cross is displayed.

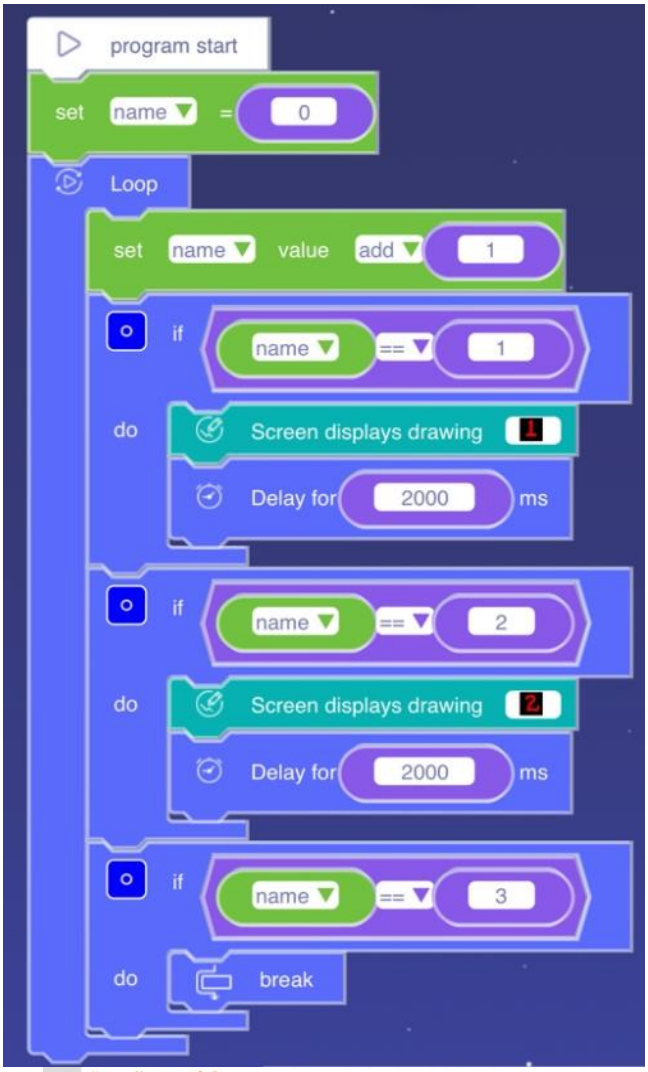



```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
name = 4
while True:
    if name == (4):
        robot.act_svg("../../source_tmp/1641545394.jpg")
    else:
        robot.act_svg("../../source_tmp/1641545417.jpg")
```

2. "Variable value addition and subtraction" Programming Block



Name: Variable value addition and subtraction operation
Function: This programming block can realize the addition and subtraction of variable values.
Parameters: This programming block has 3 parameters, the first parameter is "name represents the variable name", the name of the created variable can be selected in the drop-down list, the second parameter is the operator list (including addition and subtraction), and the third parameter is a specific value. (Note: The ellipse represents -- a value.)
Example: Loop judgment, when the variable value is 1, the screen displays the number 1, and continues to judge after 2 seconds; when the variable value is 2, the screen displays the number 2, and continues to judge after 2 seconds; when the variable value is 3, Jump out of the loop and the program stops.



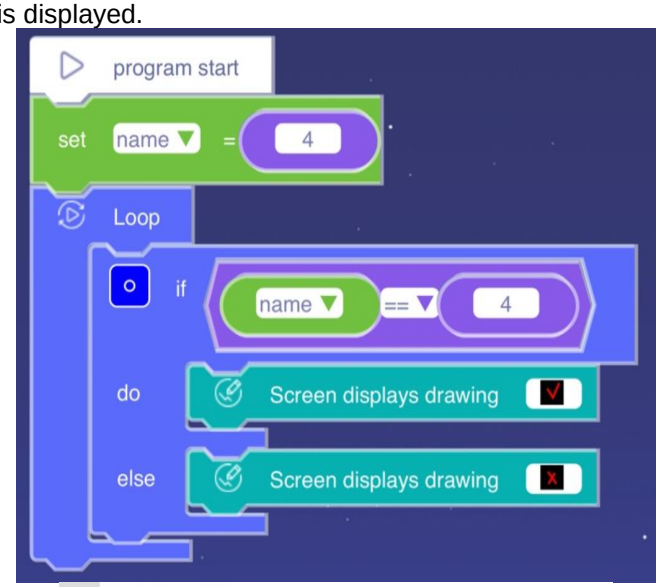

```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
name = 0
while True:
    name += (1)
    if name == (1):
        robot.act_svg("../../source_tmp/1641545664.jpg")
        sleep((2000))
    if name == (2):
        robot.act_svg("../../source_tmp/1641545394.jpg")
        sleep((2000))
    if name == (3):
        break
```

3. "Variable Name" Programming Block

Name: variable name
Function: Variables are quantities that can be changed and can be created independently. When you first create it, you need to name the variable.
Pay attention to naming conventions

- It can be composed of letters, numbers, and underlines (_), where numbers cannot begin;
- Cannot be a Python keyword, but can contain keywords;
- Cannot contain spaces.

Parameters: This programming block has 1 parameter, "name represents the variable name", and the created variable name can be selected in the drop-down list.
Example: Loop judgment, if the variable name is equal to 4, the screen displays a checkmark, otherwise a cross is displayed.




```
# coding=utf-8
import random
import sys
sys.path.append('/userdata/')
from clicbot import *
robot = Clicbot()
#program start
name = 4
while True:
    if name == (4):
        robot.act_svg("../../source_tmp/1641545394.jpg")
    else:
        robot.act_svg("../../source_tmp/1641545417.jpg")
```

The "Motion" Library

Premade demo-motions, rotation motions and steering wheel motions of the robot configuration will appear in this section.