Topic 3 Getting Ready

The successful delivery of the superchips has established a strong foundation for the implementation of the Flying Plan. Currently, UGOT is preparing to embark on a space mission to explore the mystery of the Moon and Mars.

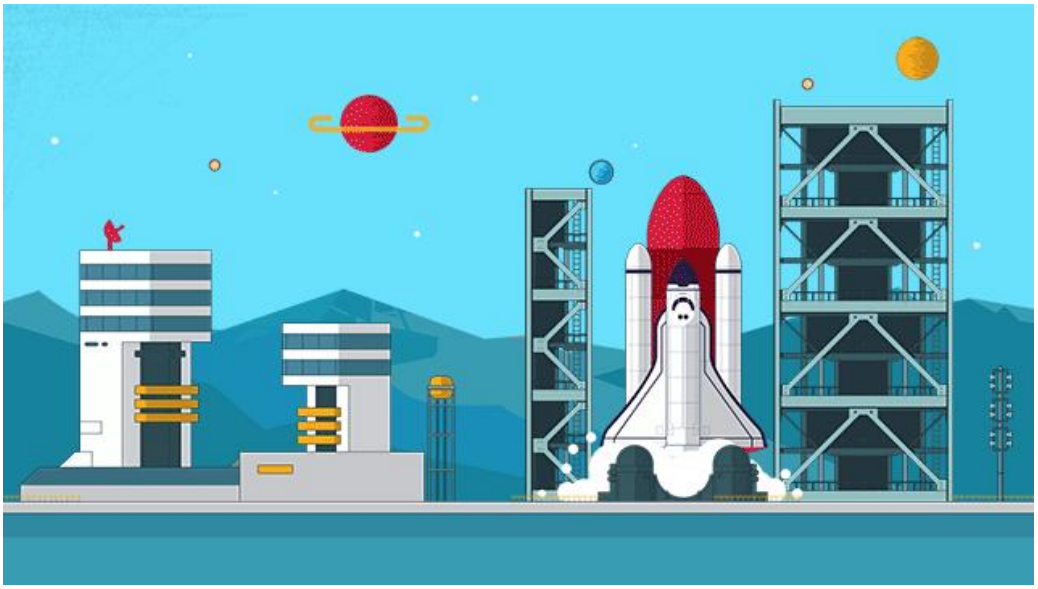

Figure 3.1 Rocket launch

I. Super UGOT

To ensure the success of UGOT's exploration missions to the Moon and Mars, final tests must be conducted before launching the rocket.

These tests include display, motion control, and function tests, as well as simulation tests on UGOT's four important functional systems: main controller light, speech, motor, and servo. The purpose of these tests is to ensure a smooth launch of the Flying Plan.

In order to perform simulation function tests of UGOT's main controller lights, speech, motors, and servos, the following problems need to be solved:

1. How do we select the functions that require testing?
2. How can we simulate the test and stop the test based on the selection?

II. Happy Learning

i. Background Knowledge

UGOT function tests

Figure 3.2 UGOT Transforming Car

1. Screen display: The main controller display supports displaying text and the background colour.
2. Main controller light bar effect: controls the main controller light bars to light up in the specified colour or turn off.
3. TTS speech: broadcasts the input text content via speeches.

4. Motor: controls one or more motors to rotate at a specified speed.
5. Servo: controls one or more servos to rotate to a specified angle.

ii. Coding Knowledge

1. input() function

input() is a built-in function in Python to receive user input. The input will be used as the return value of the function.

Demo program:

```
1  input("Please select the function you want to test:")
```

```
Please select the function you want to test: 1
'1'
```

Content input by the user via the keyboard

The return value of the function, which is a string

Figure 3.3 input() function test

Note that the data obtained by the input() function is always of type string. If you want to convert data to other types (such as integers, floating point numbers, etc.), use the appropriate function.

2. Variables

In Python, a variable can be thought as a box that holds data. The data is taken out for comparison each time it is used, and new data is stored as appropriate at the end.

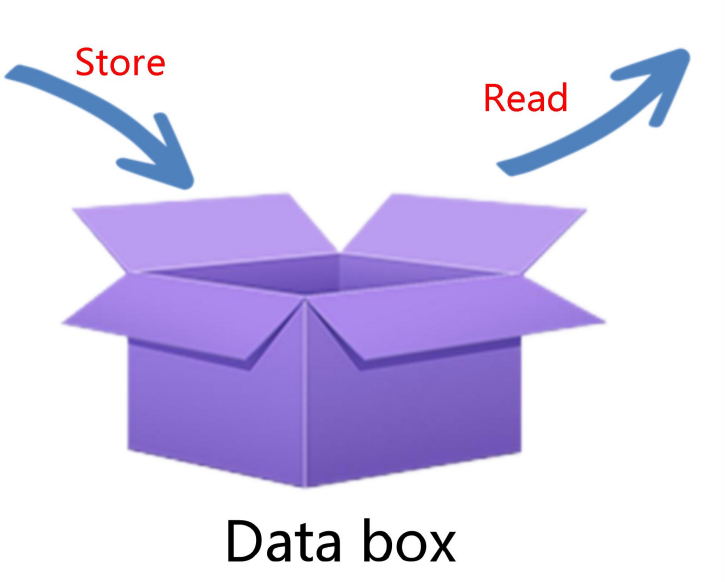

Figure 3.4 Introduction of variables

If we want to use a variable saved by the computer, we can use the variable name to get it. The following rules apply to the naming of variables:

(1) **Variable names must begin with a letter or an underscore, not a number.** For example, 'student1' is valid, but '1student' is not.

(2) **Variable names can only contain letters, numbers, characters, and underscores**. Special characters (@, ¥, #, etc.) are not allowed.

(3) **Variable names are case-sensitive.** For example, 'Student' and 'student' are two different variables.

(4) **You cannot use Python keywords as variable names.** For example, avoid using words such as 'for', 'if', and 'else'.

(5) **Variable names should describe the use or the content of the variable as much as possible.** For example, to create a variable to store the user's age, you can use age as the variable name.

(6) **When the variable name consists of two or more words, an underscore can be used to separate them.** Example: my_age.

Demo program:

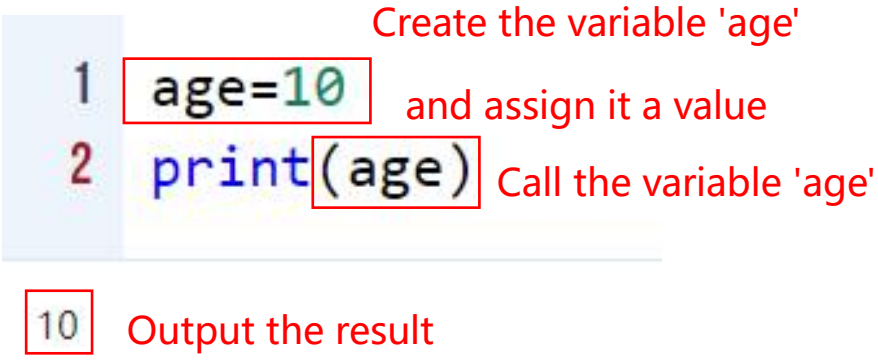

Figure 3.5 Demo variable program

3. Main controller light bar colour display

We can use the 'show_light_rgb(lights, red, green, blue)' method to illuminate a certain colour of the main controller lights. The details are as follows:

Method: show_light_rgb(lights, red, green, blue)

Function: controls the lighting of a certain colour on a light bar of the main controller light

Parameters:

lights: light bar list [0-3]; [0] represents the upper light bar, [1] represents the left light bar, [2] represents the right light bar, [3] represents the lower light bar, and [0, 1] represents the upper and left light bars.

4. Main controller light off

We can use the 'turn_off_lights()' method to turn off all the main controller lights. The details are as follows:

Method: `turn_off_lights()`

Function: turn off all main controller lights

5. Speech broadcast

We can play TTS speeches using the 'play_audio_tts(data, voice_type, wait)' method. The details are as follows:

Method: play_audio_tts(data, voice_type, wait)

Function: plays TTS speeches

Parameters:

data: string type; you can write the content to be played

voice_type: the timbre of the speeches to be played, with 0 representing a female voice and 1 representing a male voice

wait: boolean type; it judges whether there is blocking; the values include True and False;

the default value is False, indicating a non-blocking state.

6. Motor rotation

We can use the 'transform_motor_control(lf, rf, lb, rb)' method to control the rotation of the UGOT Transforming Car's motors. The details are as follows:

Method: transform_motor_control(lf, rf, lb, rb)

Function: controls the rotation of the UGOT Transforming Car's motors

Parameters:

lf,rf,lb,rb: lf represents the speed of the left front wheel, rf indicates the speed of the right front wheel, lb represents the speed of the left rear wheel, and rb represents the speed of the right rear wheel. The speed range is -360 to 360 rpm.

7. Servo rotation

We can use the 'transform_arm_control(joint, position, time)' method to control the rotation of the UGOT Transforming Car's servos. The details are as follows:

Method: transform_arm_control(joint, position, time)

Function: controls the rotation of the UGOT Transfoming Car's servos

Parameters:

joint: wheel arm; int type; number 1 represents the left front arm, number 2 represents the left rear arm, number 3 represents the right rear arm, and number 4 represents the right front arm.

III. Creative Factory

With the knowledge that we have gained, let's test the functions of UGOT together!

1. Task 1

Task overview: Program to implement the selection of functions to be tested.

```python
#Initialisation
from ugot import ugot
u=ugot.UGOT()
u.initialize("192.168.11.77")
u.transform_restory()
u.screen_display_background(0)    #Set the background of the main control display to black
u.transform_set_chassis_height(3) #The UGOT Transforming Car is 3 cm high

while True: #Loop
  x=input("Select a function you want to detect: 1/Light, 2/Speech, 3/Motor, 4/Servo, Others/Exit:")
  #Use the input() function to obtain what the user inputs through the keyboard
  #The X variable stores the unknown content input by the user

  if x=="1":   #Compare the input content with the target
    u.screen_print_text_newline("Detection of main controller light",1) #Show the text in white on the main control display
  elif x=="2":
    u.screen_print_text_newline("Speech detection",1)
  elif x=="3":
    u.screen_print_text_newline("Motor detection",1)
  elif x=="4":
    u.screen_print_text_newline("Servo detection",1)
  else:
    u.screen_print_text_newline("Detection completed!",1)
    u.screen_clear()  #Clear the main control display
    break
```

Figure 3.6 Demo program of Task 1

2. Task 2

Task overview: Program to implement various simulation tests of the corresponding functions according to the selection.

```
#Initialisation
from ugot import ugot
u=ugot.UGOT()
u.initialize("192.168.11.77")
u.transform_restory()
u.screen_display_background(0)    #Set the background of the main control display to black
u.transform_set_chassis_height(3) #The UGOT Transforming Car is 3 cm high

while True:
  x=input("Select a function you want to detect: 1/Light, 2/Speech, 3/Motor, 4/Servo, Others/Exit:")
  if x=="1":
    u.show_light_rgb([0,1,2,3], 255, 0, 0)      #Four light bars light up in red
    u.screen_print_text_newline("The main controller lights function normally!",1)
  elif x=="2":
    u.play_audio_tts("Hello!",1,True)            #Say 'Hello' with a male voice
    u.screen_print_text_newline("Speech function is normal!",1)
  elif x=="3":
    u.transform_motor_control(80, 80, 80, 80)    #Control the four motors of the Transforming Car to rotate
    u.screen_print_text_newline("Motors function normally!",1)
  elif x=="4":
    u.transform_arm_control(1, 30, 10)    #Control the four servos of the Transforming Car to rotate
    u.screen_print_text_newline("Servos function normally!",1)
  else:
    u.screen_print_text_newline("Detection completed!",1)
    u.screen_clear()     #Clear the main control display
    u.turn_off_lights()  #Turn off all lights
    u.transform_stop()   #The Transforming Car stops
    break
```

Figure 3.7 Demo program for Task 2

IV. Knowledge Summary

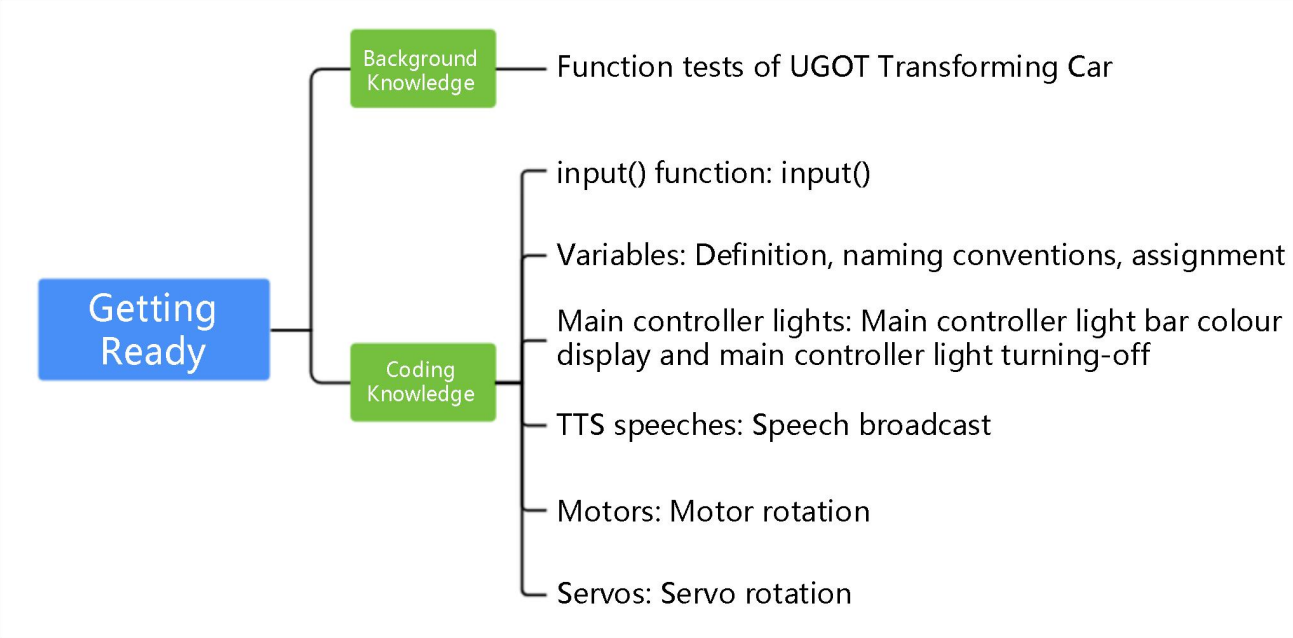

Figure 3.8 Knowledge summary

V. Extension

i. Practice & Innovation

Complete the following extension task:

Try to refine the program to implement the speech broadcast function instead of displaying the prompt content.

ii. Additional Knowledge

Variables and constants in Python

Variables

Variables are entities that store data and their values can be changed. In Python, variables don't need to be declared. They are created when a

value is first assigned to them. Python is a dynamic language, which means that the type of a variable can be changed during program execution; for example, an integer can be assigned to a variable and then later a string can be assigned to the same variable.

Constants

Constants are special variables whose values should not be changed after they are defined. Python does not have built-in support for constants, so you can modify a constant's value at any time.

However, Python programmers usually use all-caps variable names for constants. This convention serves as a reminder to other programmers that the value of the variable should not be changed.