Confirmation of qualification

To get started with the working tasks, demonstrate your knowledge level.

Prove that you are ready for a "brainstorm"!

Confirmation of qualification

Which commands fit the description?

Change the thickness of the turtle's pen by 5 pixels	?
Start/Stop filling with the current color	?
Pen up/Pen down	?
Change the color of the turtle's pen to red	?
Move the turtle to the point with coordinates (20, 130)	?

Confirmation of qualification

Which commands fit the description?

Change the thickness of the turtle's pen by 5 pixels	`pensize(5)`
Start/Stop filling with the current color	`begin_fill() end_fill()`
Pen up/Pen down	`penup() pendown()`
Change the color of the turtle's pen to red	`color("red")`
Move the turtle to the point with coordinates (20, 130)	`goto(20, 130)`

Confirmation of qualification

Which command changes the turtle's speed?

How to hide an executor (remove from the screen)?

Confirmation of qualification

Useful commands:

Command	*Value*
`speed(`<number in 0 - 10 range>`)`	Change the turtle's speed
`hideturtle()`	Hide the executor turtle from the screen (only the picture remains)

Speed 3

Speed 10

Confirmation of qualification

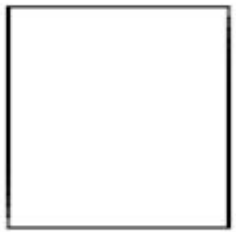
What is the best way to render a square you know?

And what about the five-pointed star?

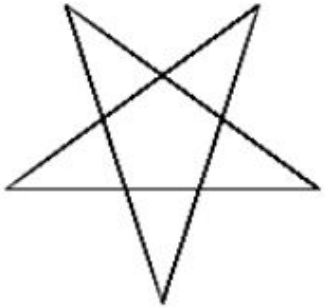
Confirmation of qualification

These figures can be rendered using a loop:

```
from turtle import *
for i in range(4):
    forward(100)
    left(90)
hideturtle()
exitonclick()
```



```
from turtle import *
for i in range(5):
   forward(150)
   left(144)
hideturtle()
exitonclick()
```



Confirmation of qualification

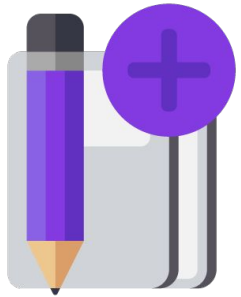
Qualifications confirmed!

Great, you are ready to brainstorm and complete your work task!

Confirmation of qualification

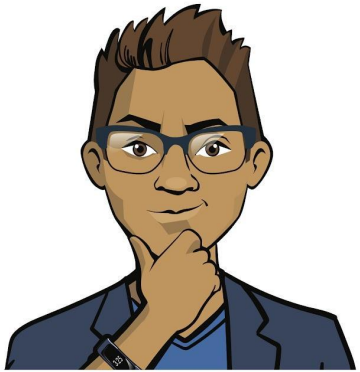
"Brainstorm":

Rendering of figures with the condition analysis



Working with conditions

Let's consider several different tasks for rendering figures depending on the data entered by user.



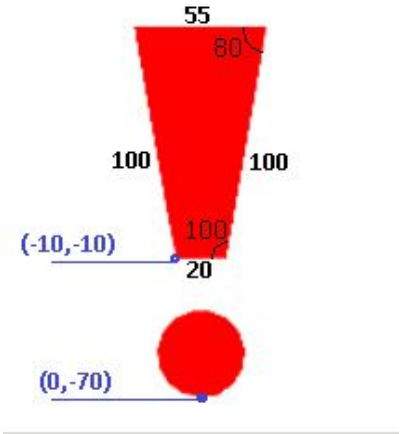
"Brain storm"

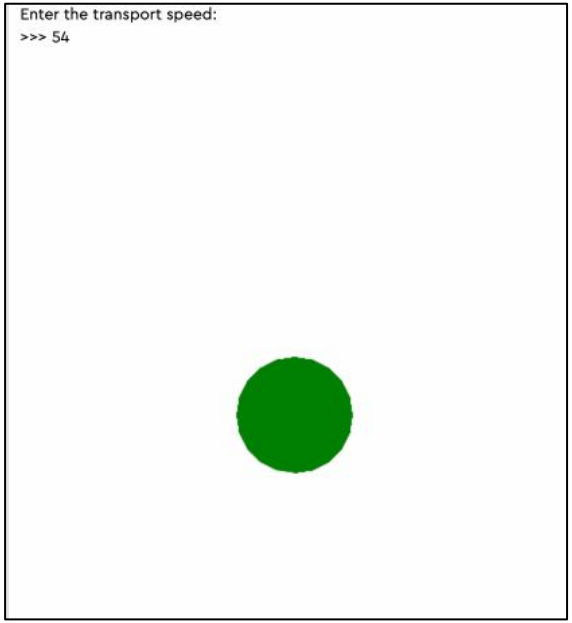
Let's return to the task under discussion

Task. The police decided to mount a scoreboard displaying a warning if the car has gone over the speed limit. The speed limit within the city is 60 km/h. Code a program that requests and analyzes the speed as it is shown in the picture.

Figures Parameters:

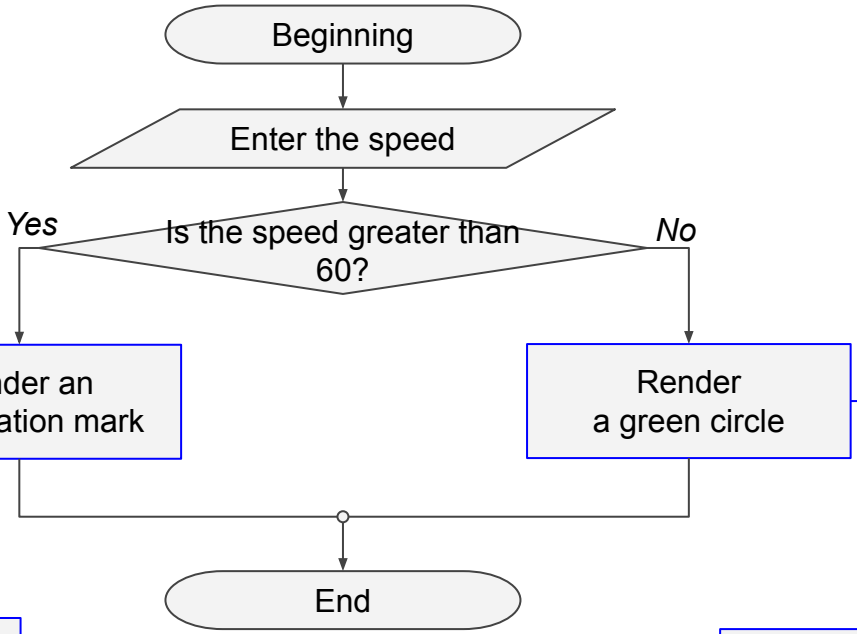
- A **circle** of green color with a radius of 60.
- **Exclamation mark** of red color with parameters given in the picture:





"Brain storm"

Solution search flowchart:



A separate linear **algorithm** (function) **that renders a sign**

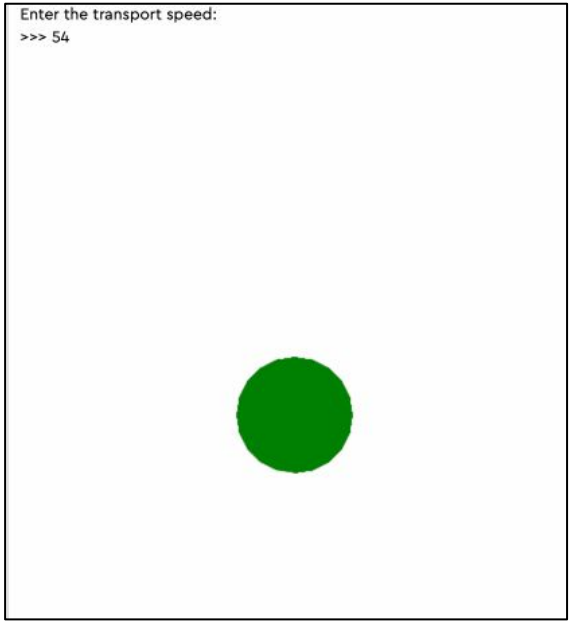
A separate linear **algorithm** (function) **that renders a circle**

"Brain storm"

Sample code:

```
def speed_ok():
    color("green")
    begin_fill()
    circle(50)
    end_fill()

def speed_over():
    color("red")
    penup()
    goto(0,-70)
    pendown()
    begin_fill()
    circle(18)
    end_fill()
    penup()
    #and so on
```

```
from turtle import *
speed = int(input("Transport speed:"))
if speed <= 60:
    speed_ok()
if speed > 60:
    speed_over()
hideturtle()
exitonclick()
```



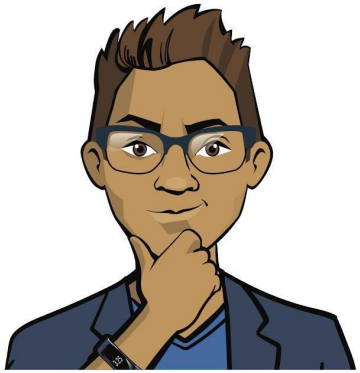
"Brain storm"

Before we continue:

In Russia, there is a "non-fined threshold for speeding". If the driver exceeds the speed by less than 20 km/h, a warning is issued instead of a fine.

The police intend **to add a yellow circle to the program** for warning within the non-fined threshold.

How can we change the program? Describe the new flowchart.



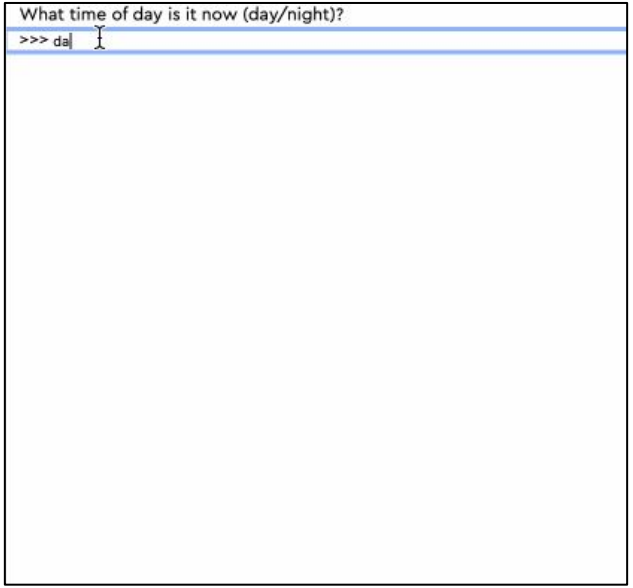
"Brain storm"

Program the night mode!

Task. Configure the program interface with day and night mode. The policeman will enter the time of day. If it is "day" at the moment, then the program should render the Sun. If it is "night" at the moment, then the program should render the Moon. Otherwise, leave the screen blank.

Figures Parameters:

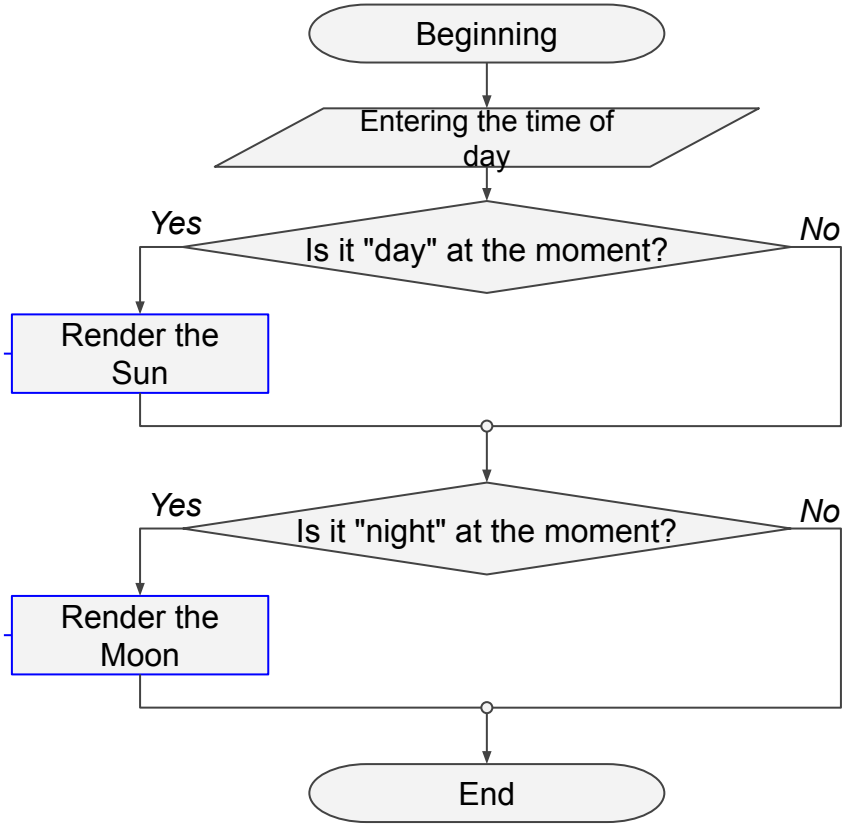
- **The Sun:** the color is yellow (yellow), the length of the segment at its base is 100, the number of rays is 18.
- **The Moon:** there is a beige (bisque) circle at its base with a radius of 50.



"Brain storm"

Solution search flowchart:

A loop **algorithm** (function) **that renders the Sun**

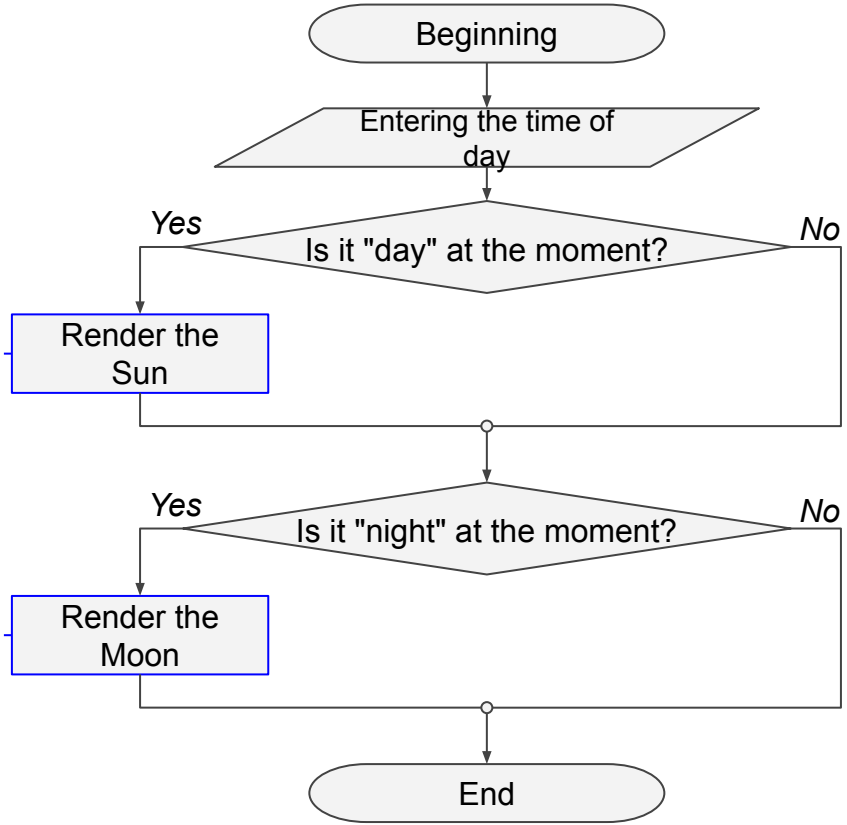


Linear **algorithm** (function) **that renders a circle**

"Brain storm"

Solution search flowchart:

A loop **algorithm** (function) **that renders the Sun**



Linear **algorithm** (function) **that renders a circle**

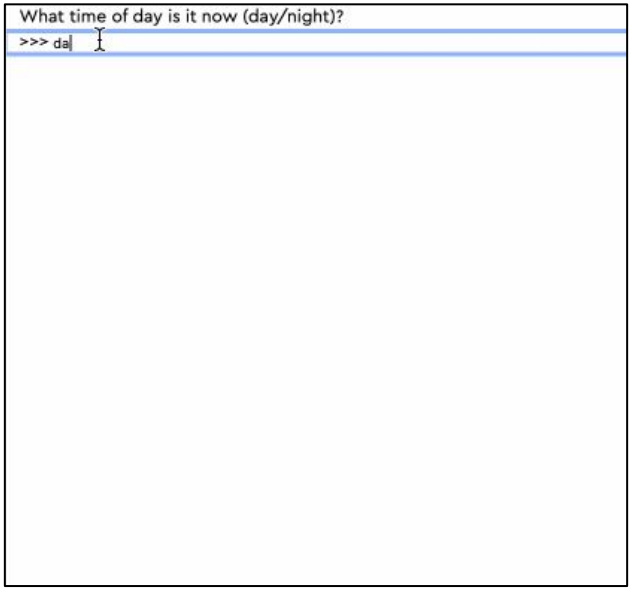
"Brain storm"

Is it possible to use another type of conditional statement? Which one?

Sample solution:

```
def day():
    color("yellow")
    begin_fill()
    for i in range(18):
        forward(100)
        left(100)
    end_fill()

def night():
    color("bisque")
    begin_fill()
    circle(50)
    end_fill()
```

```
from turtle import *
speed(0)
answer = input("What time of day is it now (day/night)?")
if answer == "day":
    day()
if answer == "night":
    night()
hideturtle()
exitonclick()
```

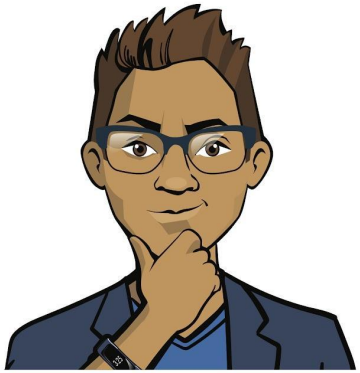


"Brain storm"

Before we continue:

The ProTeam design department reviewed the program and recommended improving the night mode.

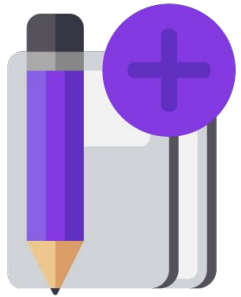
- **How can we change the program** so that, in night mode, the moon would be rendered on a dark blue background?
- **How can we make the Moon more realistic** and add several craters?



"Brain storm"

"Brainstorm":

Rendering of figures with the condition analysis

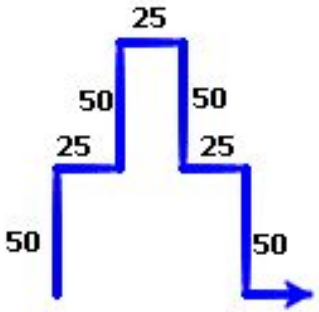


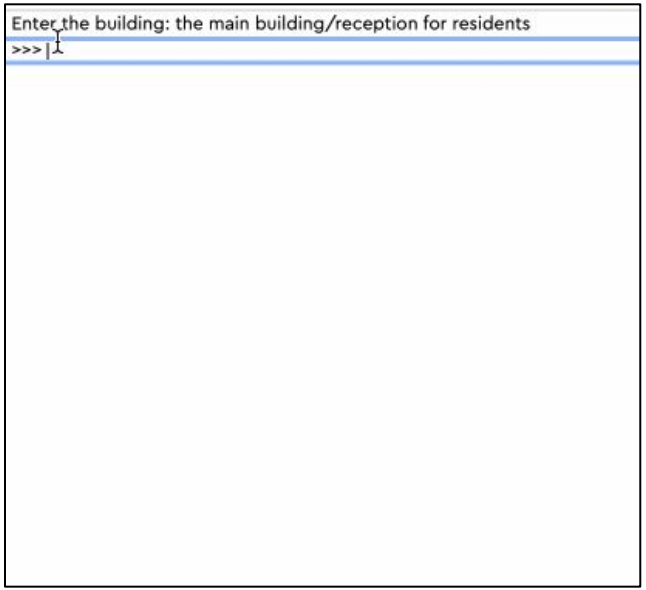
Replenish the database of the recognition system

Task. Add the city administration building to the database of the urban environment objects. It is composed of two buildings: the main one (blue) and the reception for residents (green). A fence of the appropriate color should be rendered in response to a request, whether it is "main building" or "reception for residents." In case of entering any other data, the screen should be left blank.

Fence parameters:

- There are 4 sectors in the fence. The blue color — blue, the green color — green.
- Parameters of a single sector:

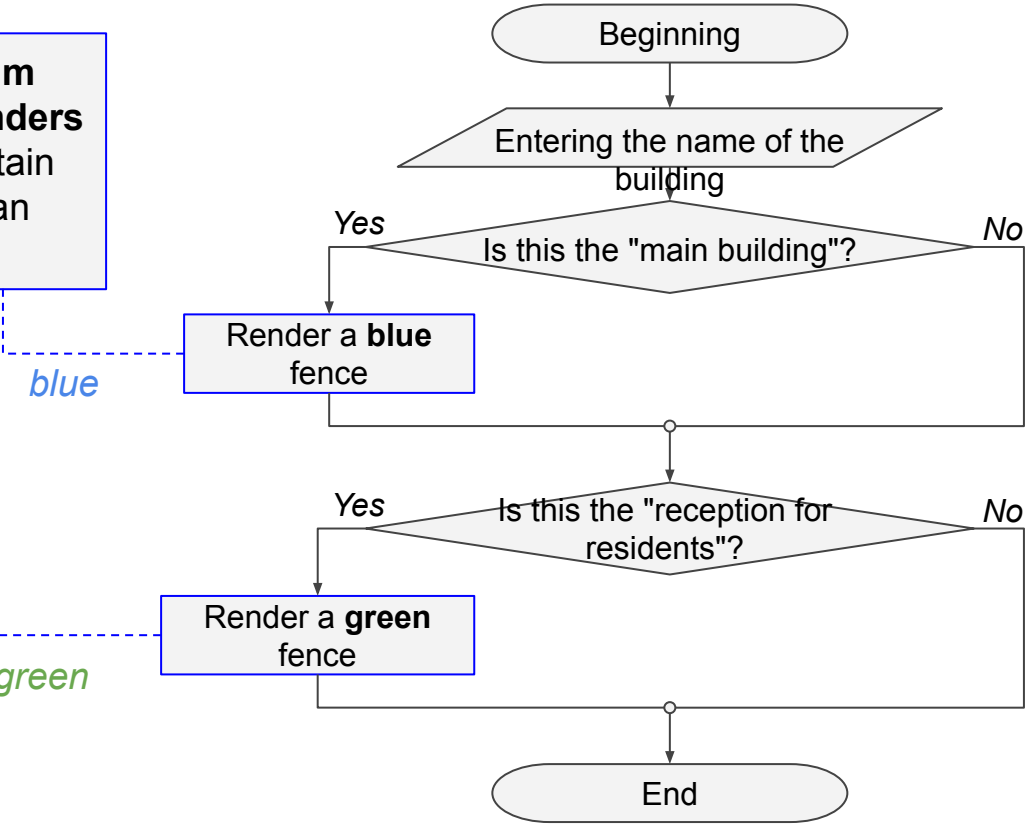




"Brain storm"

Solution search flowchart:

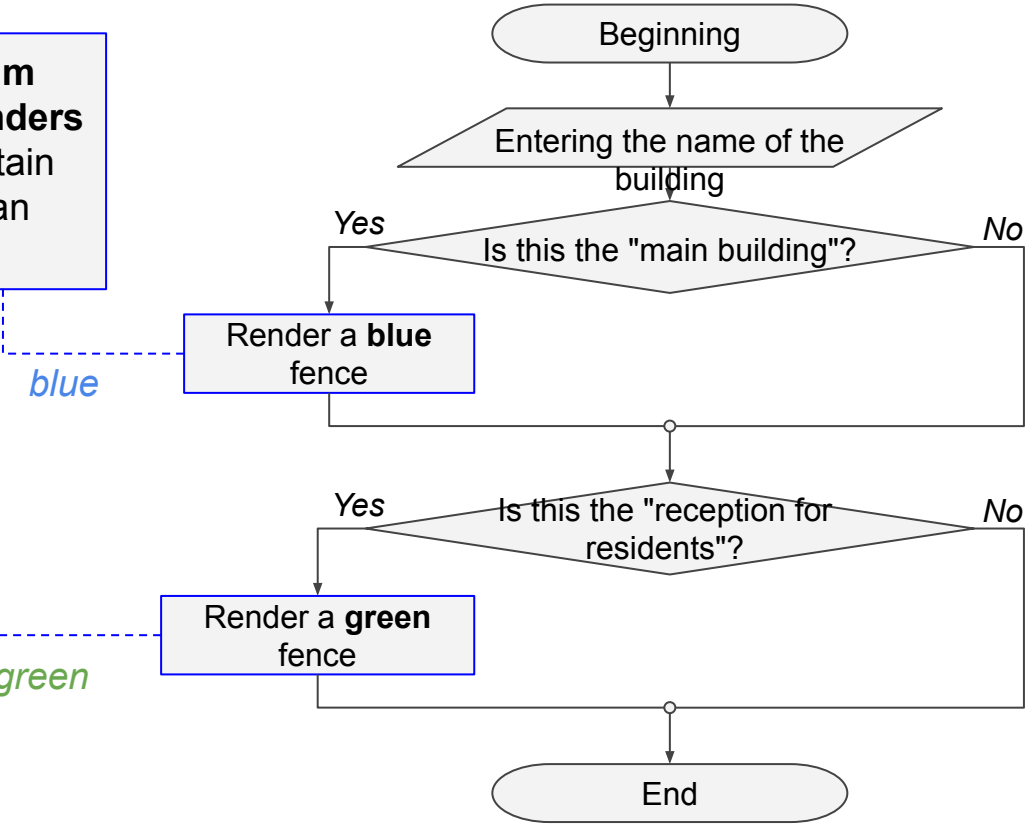
A loop **algorithm** (function) **that renders a fence** of a certain color (color is an argument)



"Brain storm"

Solution search flowchart:

A loop **algorithm** (function) **that renders a fence** of a certain color (color is an argument)



"Brain storm"

Is it possible to use another type of conditional statement? Which one?

Sample code:

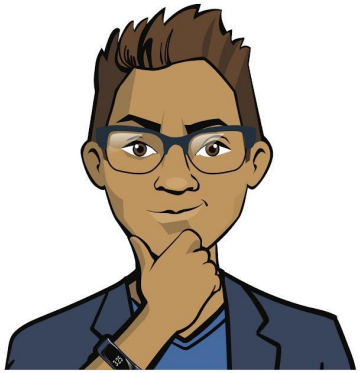
```
def fence(color_f):
    color(color_f)
    penup()
    goto(-215, 0)
    pendown()
    for i in range(4):
        left(90)
        forward(50)
        right(90)
        forward(25)
        left(90)
        #and so on
```

```
from turtle import *
speed(0)
pensize(3)
answer = input("Enter the building (main building/reception for
residents):")
if answer == "main building":
    fence("blue")
if answer == "reception for residents":
    fence("green")
hideturtle()
exitonclick()
```

"Brain storm"

Before we continue:

1. **Fix the program:** the city architect has observed the fence of the reception for residents to be shorter than the main building's. If the main building has 4 sectors, then the reception should have 3 ones.
2. The tester has complained of the program interface being inconvenient as it requires entering the full name of the building. **How can we simplify entering the building's name?**



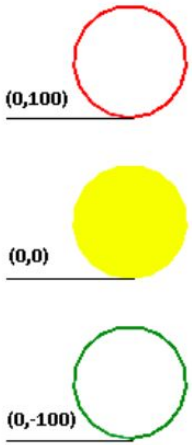
"Brain storm"

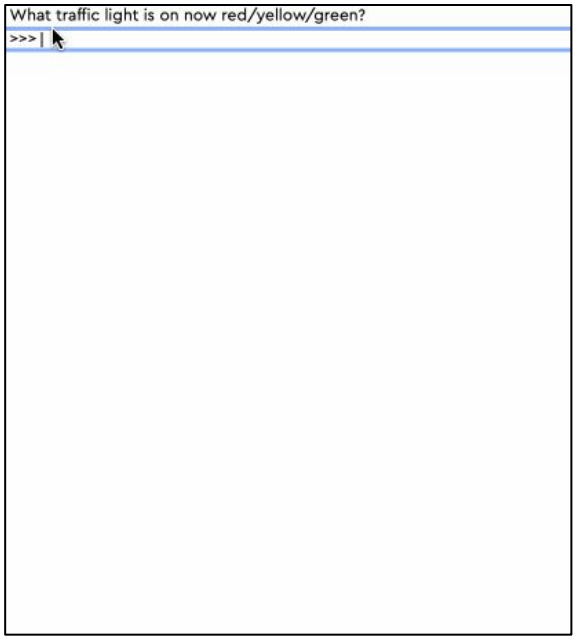
Program the digital traffic light

Task. To analyze the reasons for road accidents, the cameras should recognize traffic light signals. Replenish the cameras' database with the images of different signals: red, yellow, and green lights turned on. The program should paint over the required signal upon entering its color; for example, if "green" is entered, it means that the green light is turned on.

Traffic light parameters:

- **Colors:** red, yellow, green.
- **Circles:** radius is 35, location:





"Brain storm"

Solution search flowchart:

Algorithm "Turn the red light on"

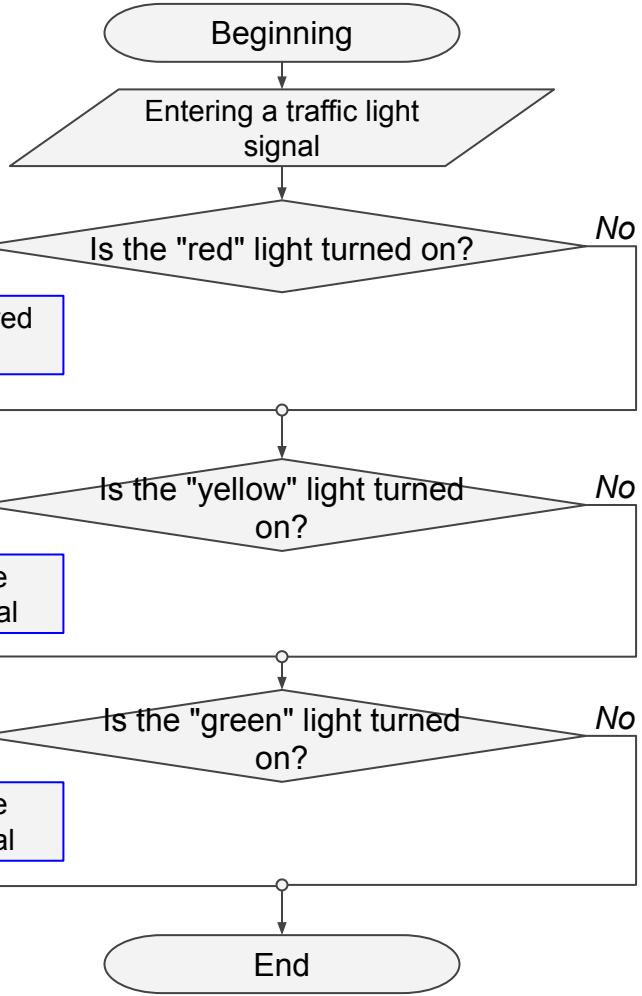
Algorithm "Turn off the red light"

Algorithm "Turn on the yellow light"

Algorithm "Turn off the yellow light"

Algorithm "Turn on the green light"

Algorithm "Turn off the green light"



"Brain storm"

Sample solution:

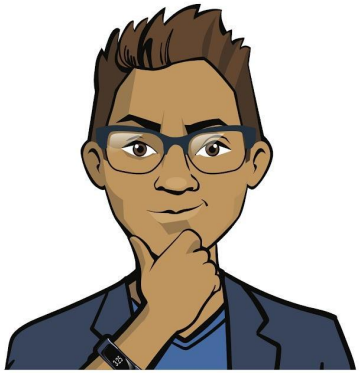
```
def red_light_on():
    color("red")
    penup()
    goto(0, 100)
    pendown()
    begin_fill()
    circle(35)
    end_fill()

def red_light_off():
    color("red")
    penup()
    goto(0, 100)
    pendown()
    circle(35)

#similar to other functions
```

```
from turtle import *
speed(0)
answer = input("Traffic light signal (red/yellow/green):")
if answer == "red":
    red_light_on()
    yellow_light_off()
    green_light_off()

#similar to other signals

hideturtle()
exitonclick()
```

"Brain storm"

Before we continue:

1. The customer asked for adding a rectangular black frame with a thickness of 5 pixels to the traffic light. **Describe the necessary changes in the program**.
2. In our city, the green light is turned on for 1 minute in average. **How can we supplement the program** so that the green signal would turn off after 60 seconds of being turned on?



"Brain storm"