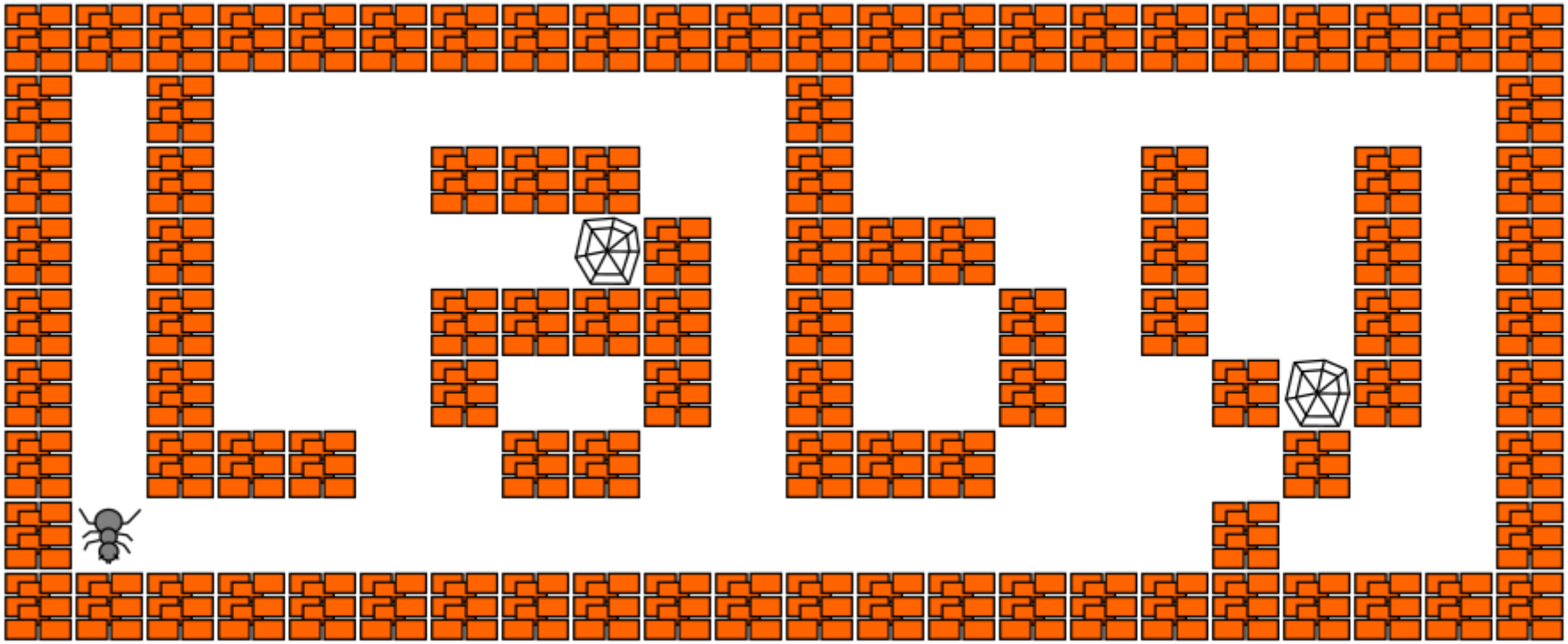
Laby using Python



Hamilton Python Users Group
Ian Stewart
18 May 2020

Laby using Python

Laby is a GUI application to learn how to program with ants and spider webs. You have to move an ant out of a labyrinth, while avoiding spider webs and moving rocks in your path.

Using Laby, you can learn: OCaml, C, C++, Java, Prolog, Ruby, Pascal, JavaScript, Python, Lua, Vala and Scheme.

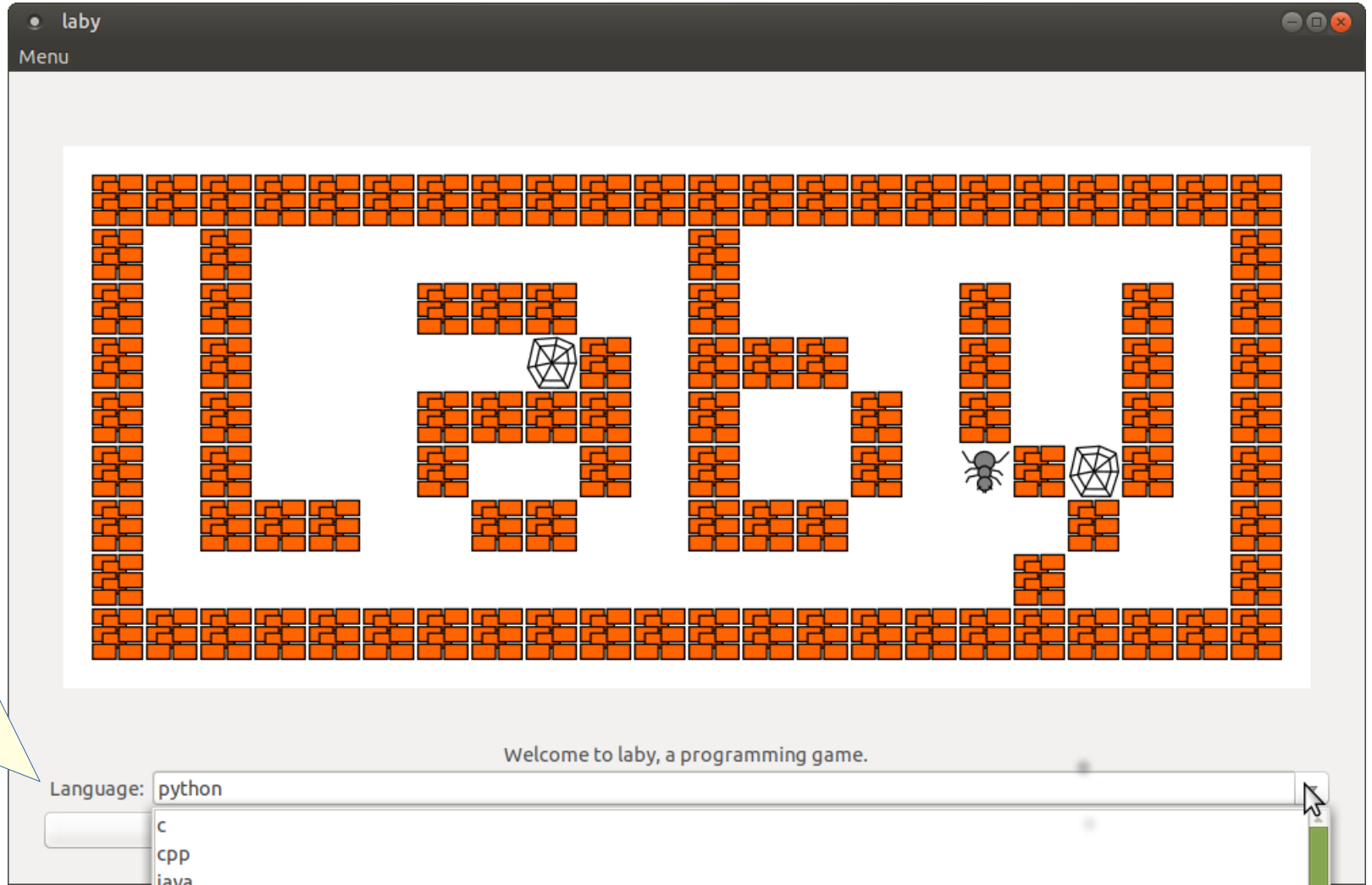
Installation on Linux / Debian platform:

```
$ sudo apt install laby
```

Laby using Python

- Initial Development: 2007
- Current Version: V0.6.4-2. July 2017
- Github site: https://github.com/sgimenez/laby
- Website: https://sgimenez.github.io/laby/
- Key Author: Stéphane Gimenez – and mostly French folks
- Linux install: /usr/share/laby/
- Linux Executable: /usr/games/laby
- Labyrinths in: /usr/share/laby/levels
- Written in: OCaml
- Python: /usr/share/laby/mods/python/
- Robot module: /usr/share/laby/mods/python/lib/robot.py
- Executing scripts: /tmp/ant-nnnnn-n/ program.py robot.py

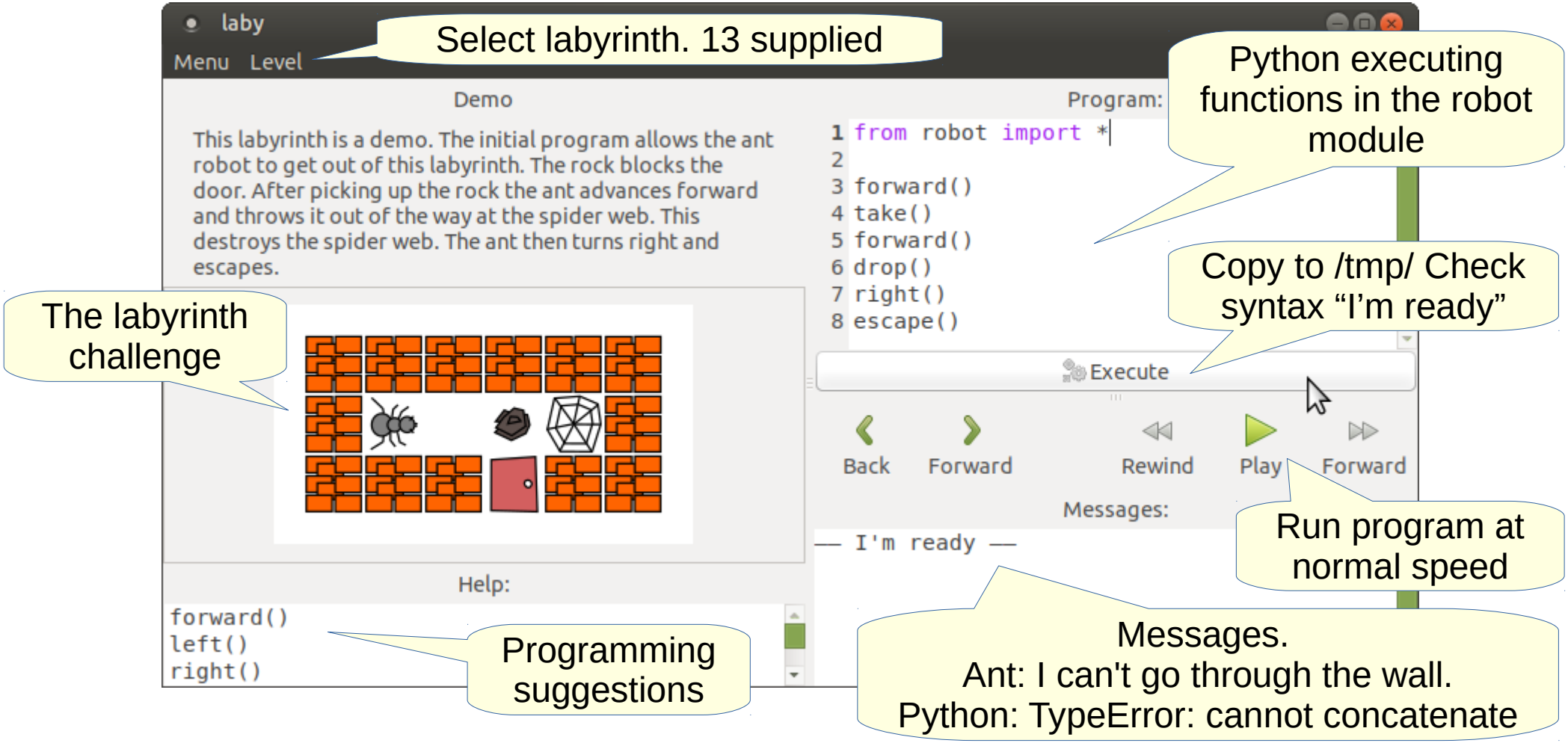
Laby using Python - Launching



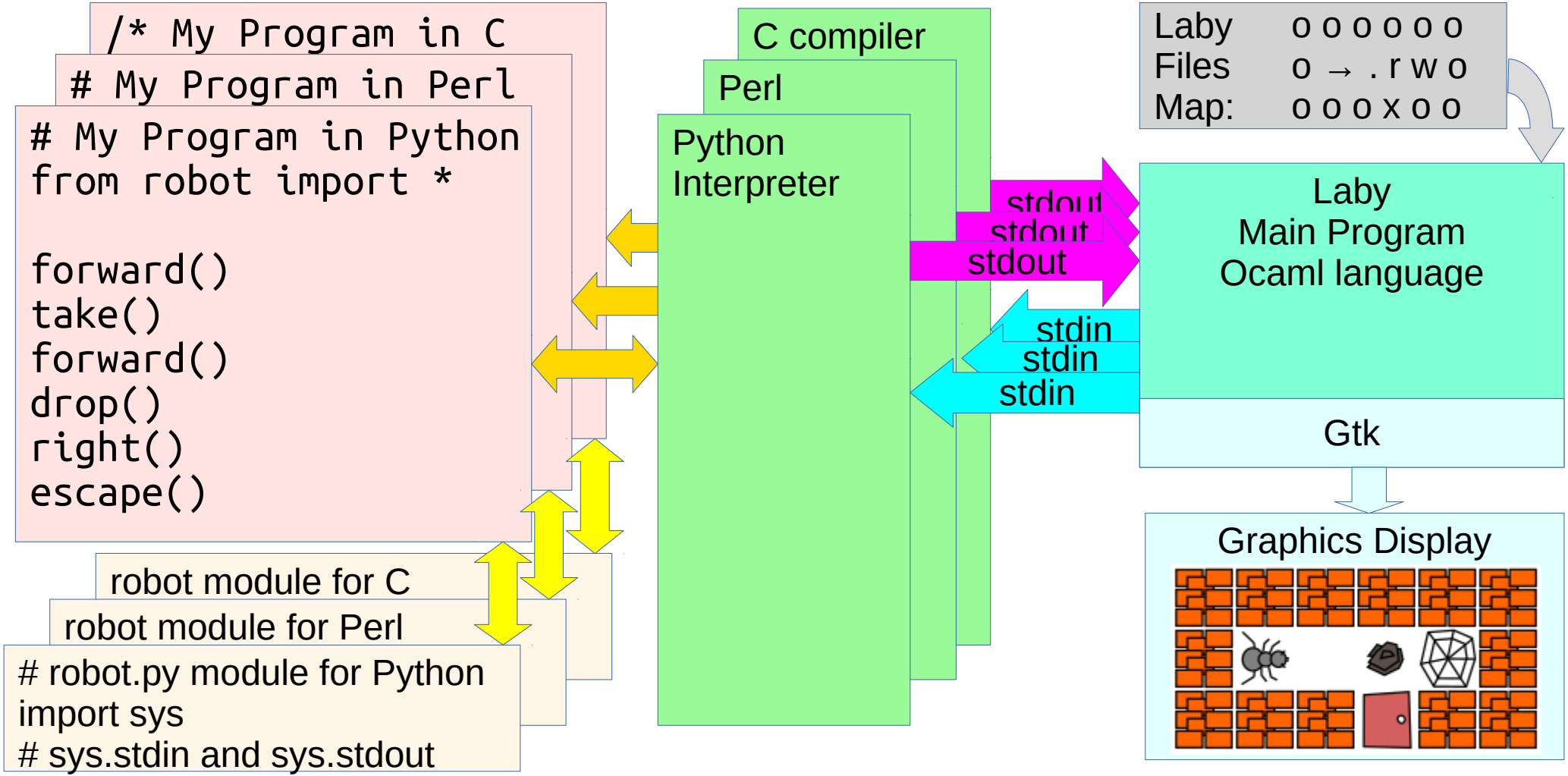
Home Screen.

On Drop down menu.
Select Python as the Language

Laby using Python: Main Window



Laby using Python – Conceptual Diagram

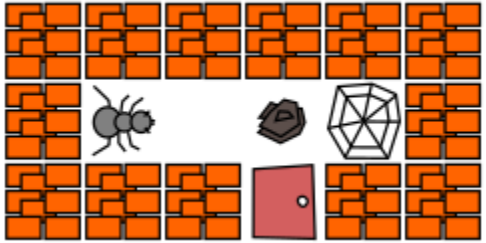


Laby. Steps in execution of program

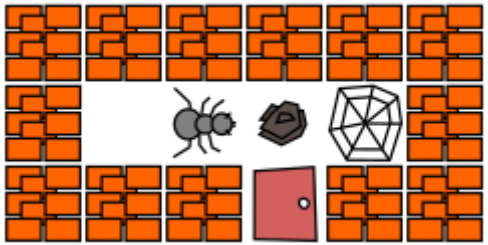
```
# My Program in Python
from robot import *

forward()
take()
forward()
drop()
right()
escape()
```

From laby file create initial graphic

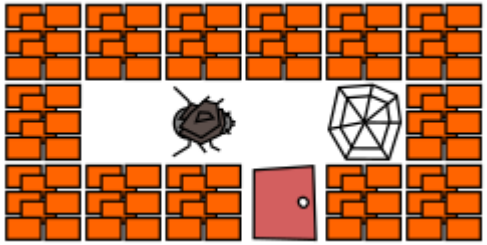
Laby	o o o o o o
Files	o → . r w o
Map:	o o o x o o



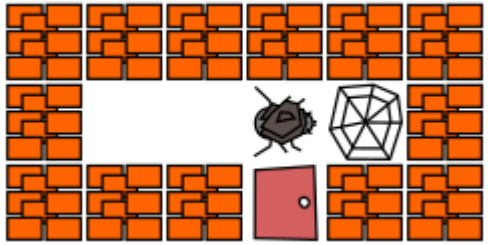
1. forward()



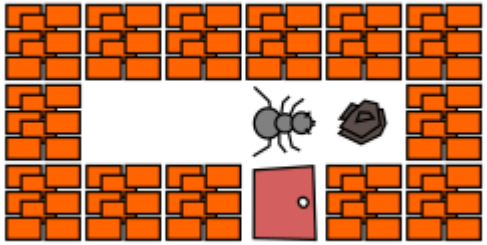
2. take()



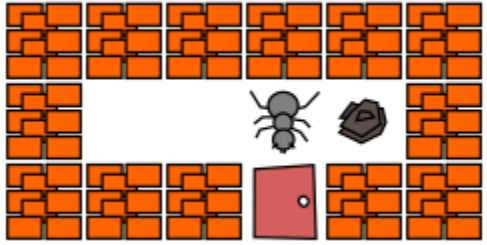
3. forward()



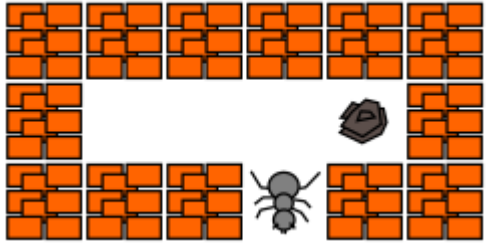
4. drop()



5. right()

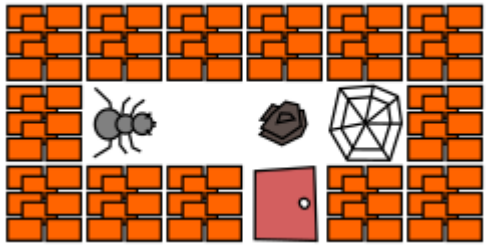


6. escape()



Laby. .laby files and the map:

Laby	o o o o o o
Files	o → . r w o
Map:	o o o x o o



```
ian@X200:/usr/share/laby$ ls
conf  levels  mods  scripts  sound  syntax  te
ian@X200:/usr/share/laby$ cd levels
ian@X200:/usr/share/laby/levels$ ls
0.laby   1c.laby  2c.laby  4a.laby
1a.laby  2a.laby  3a.laby  4b.laby
1b.laby  2b.laby  3b.laby  counting-the-rocks.
ian@X200:/usr/share/laby/levels$ cat 0.laby |
map:
o o o o o o
o → . r w o
o o o x o o

title:
text                Demo
```

To edit and create your own labyrinths its easier if only User priv is required:
$ sudo chmod 777 /usr/share/laby/levels

Laby. Rules

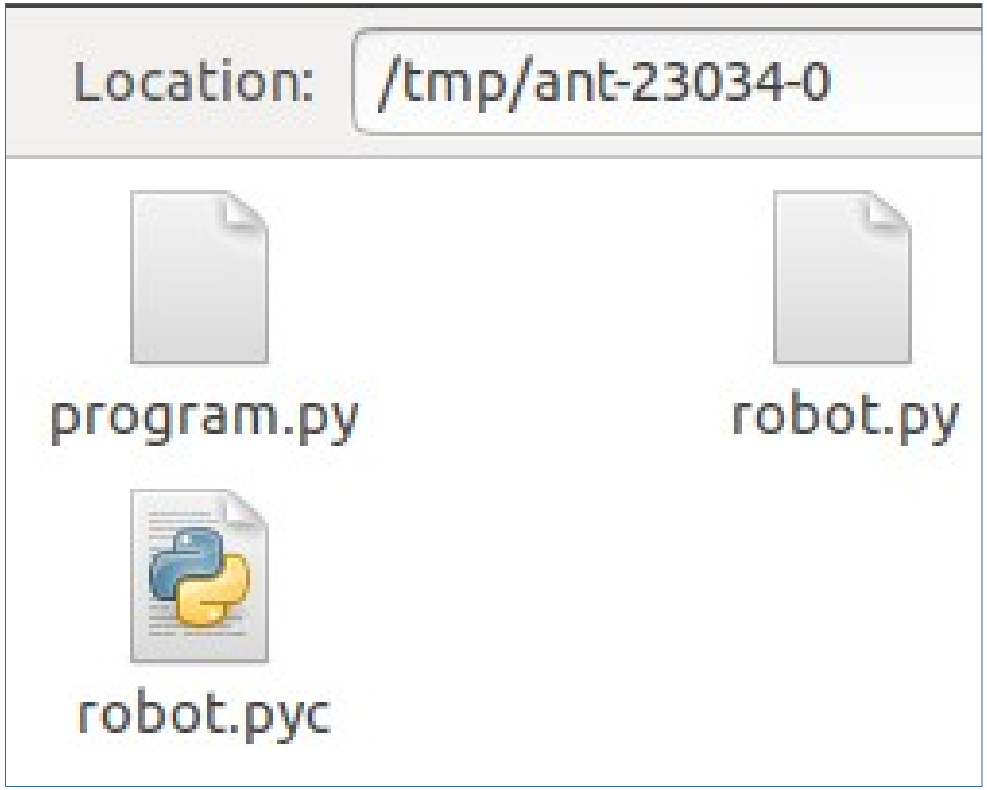
- Ant can only move forwards one square at a time.
- Ant can only determine what is in the one square in front of it.
- Ant can only turn at 90 degree increments left or right.
- Ant can carry one rock at a time.
- Ant can not fit out the exit door carrying a rock.
- If the Ant goes into a spiders web the ant is captured. No escape.
- If an Ant drops a rock in a spiders web the web is destroyed.
- A rock in front of an ant blocks it from going forward.

Rules with Random Rocks and Webs:

- Rocks and Webs may be random in where they appear.
- Ant can walk over broken rocks and small spider webs.

Laby. “Execute”

On clicking “Execute”:



- Code in your program window is written to your temporary ant folder as the file “program.py”
- Also “robot.py” is copied from /usr/share/laby/mods/python/lib/ to your temporary ant folder.
- Plus the robot.py code is also compiled into byte code and stored as the file robot.pyc.

Laby using Python ~ Location of robot.py

- robot.py is in: /usr/share/laby/mods/python/lib/
- A unique ant folder is created in /tmp/ant-nnnnn-n.
  E.g. /tmp/ant-5260-0/

```
$ ls /tmp/ant-5260-0/
program.py  robot.py  robot.pyc
```

- The code in the Program window gets written to program.py in /tmp/ant/-nnnnn-n on clicking execute.
- robot.py module is “copied” to /tmp/ant-nnnnn-n/. Thus robot.py in the python path of program.py.

Laby using Python ~ Location of robot.py

- robot.py is in: /usr/share/laby/mods/python/lib/
- A unique ant folder is created in /tmp/ant-nnnnn-n.
  E.g. /tmp/ant-5260-0/

```
$ ls /tmp/ant-5260-0/
program.py  robot.py  robot.pyc
```

- The code in the Program window gets written to program.py in /tmp/ant/-nnnnn-n on clicking execute.
- robot.py module is “copied” to /tmp/ant-nnnnn-n/. Thus robot.py in the python path of program.py.

Laby. Program and Robot module placed in /tmp/ant-n

Location: /tmp/ant-10655-0

program.py robot.py robot.pyc

program.py

```
from robot import *

right()
forward()
take()
left()
forward()
drop()
right()
forward()
```

robot.py

```
import sys;

def output(s):
  sys.stdout.write(s + "\n");
  sys.stdout.flush();

def input():
  l = sys.stdin.readline();
  if l == "quit\n": exit(0);
  return l;

def left():
  output("left"); input();

def right():
  output("right"); input();
```

Laby. robot.py

```python
import sys;

def output(s):
  sys.stdout.write(s + "\n");
  sys.stdout.flush();

def input():
  l = sys.stdin.readline();
  if l == "quit\n": exit(0);
  return l;

def left():
  output("left");
  input();

def right():
  output("right"); input();

def forward():
  output("forward"); input();

def take():
  output("take"); input();

def drop():
  output("drop"); input();

def escape():
  output("escape"); input();

def say(s):
  output("say " + s); input();

Void = 0;
Wall = 1;
Rock = 2;
Web = 3;
Exit = 4;
Unknown = 5;

def look():
  output("look");
  ans = input();
  if (ans == "void\n"): return Void;
  if (ans == "wall\n"): return Wall;
  if (ans == "rock\n"): return Rock;
  if (ans == "web\n"): return Web;
  if (ans == "exit\n"): return Exit;
  return Unknown;

output("start");
input()
```

76 x PEP8 recommendations. Plus using input() which is a built-in function.

Laby

Changed input() to response() - used for look().

Otherwise response() may be just to prevent race conditions?

```python
import sys


def output(s):
    sys.stdout.write(s + "\n")
    sys.stdout.flush()


def response():
    line = sys.stdin.readline()
    if line == "quit\n":
        exit()
    return line


def left():
    output("left")
    response()


def right():
    output("right")
    response()


def forward():
    output("forward")
    response()


def take():
    output("take")
    response()


def drop():
    output("drop")
    response()


def escape():
    output("escape")
    response()


def say(s):
    output("say " + s)
    response()


Void = 0
Wall = 1
Rock = 2
Web = 3
Exit = 4
Unknown = 5


def look():
    output("look")
    ans = response()
    if (ans == "void\n"):
        return Void
    if (ans == "wall\n"):
        return Wall
    if (ans == "rock\n"):
        return Rock
    if (ans == "web\n"):
        return Web
    if (ans == "exit\n"):
        return Exit
    return Unknown


output("start")
response()
```

robot.py
PEP8
compliant

Laby. Summary of 8 x robot.py functions:

Move: forward()

Turn: left(), right()

Rocks: take(), drop()

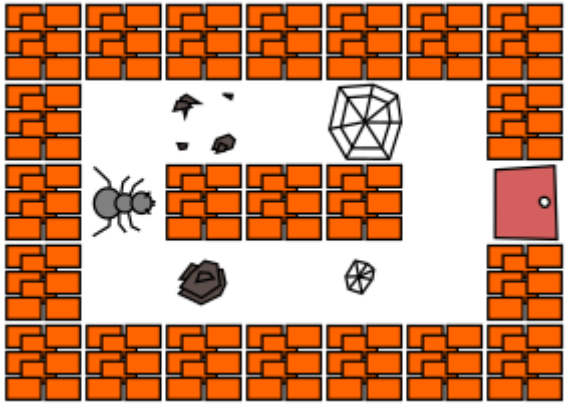
Exit Door: escape()

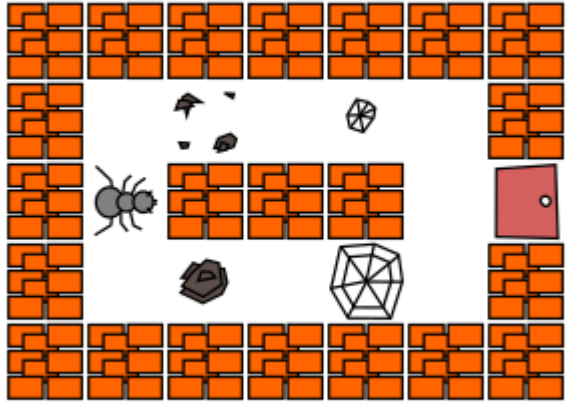
Messages: say(string)

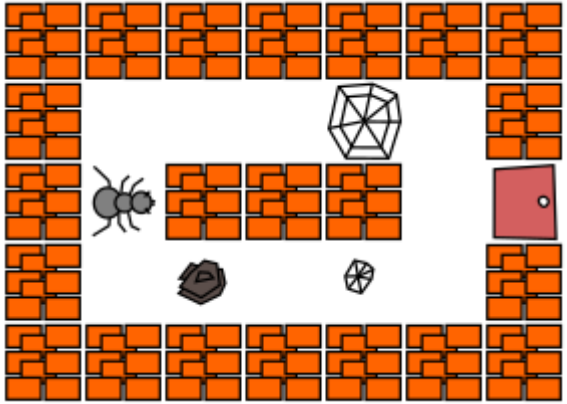
What’s in front?: look()
Possible return strings: void, wall, exit, rock, web, unknown

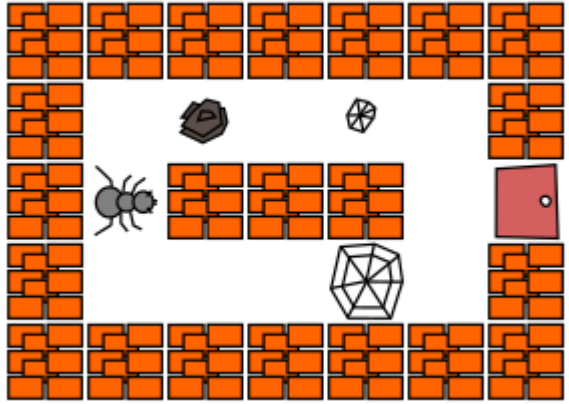
Laby. Random Rock and Web using “R” and “W”

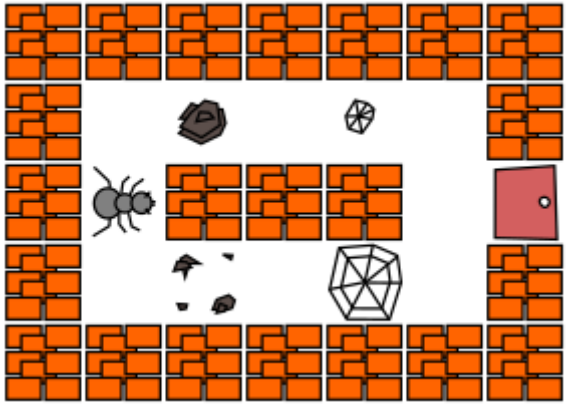
```
map:
o o o o o o o
o . R . W . o
o → o o o . x
o . R . W . o
o o o o o o o
```











Laby. Ant direction?:

Internally the program knows the direction of the ant at all times. Four icons are used to display the ants four possible directions it can face, north, east, south and west.

There is not a robot module function to retrieve the current direction that the ant is facing.

A programming approach is to manually assign the direction the ant is facing at the start. E.g. “North”. All turns that are made are tracked relative to this initial direction.

Laby. Example Python conditional code:

```
for i in range(5):
    forward()

if look() == Void:
    forward()

while look() == Void:
    forward()

integer = look():
Ocaml returns: void wall rock web exit
Python code returns: integer 0 to 5
Variables assigned to integers: Void, Wall, Rock,
Web, Exit, Unknown
```

Laby. Example code: Rock removal

```
# Get a rock out of the way to go past

if look() == Rock:
    take()
    left()
    left()
    drop()

    right(), right(), forward()
```

Comma separated function calls?
Works OK. Is it PEP8?

Laby. Example code: Go straight function

```
def go_straight():
    while look() == Void:
        forward()



go_straight()
```

Laby. Example code: General Python

```
from robot import *
import os
import time

say(sys.version)
say(os.uname()[0])
say(os.getcwd())
time.sleep(1)
```

say() can only say strings.

Avoid race condition

```
2.7.17 (default, Apr 15 2020, 17:
Linux
/tmp/ant-4975-0
```

Python version 2

Program:

```
1 from robot import *
2 import os
3 import time
4
5 say(sys.version)
6
7 say(os.uname()[0])
8
9 say(os.getcwd())
10
11 time.sleep(1)
```

Remove semicolon

sys imported by robot

Execute

Back Forward Rewind Play Forward

Messages:

```
— I'm ready —
2.7.17 (default, Apr 15 2020, 17:20:14)
Linux
/tmp/ant-4975-0
```

Laby. Example code: General Python

```
from robot import *
import os
import time

say(sys.version)
say(os.uname()[0])
say(os.getcwd())
say(os.path.expanduser("~"))
say(os.sep)
say(time.ctime())
time.sleep(1)
sys.exit("Exiting...")
```

```
── I'm ready ──
2.7.17 (default, Apr 15 20
Linux
/tmp/ant-4975-1
/home/ian
/
Tue May 12 10:04:10 2020

Exiting...
```

Laby: Ant GPS and Ant Logging system

```
home_path = os.path.expanduser("~")
dir_name = "laby_data"
file_name = "gps_data"
full_path = home_path + os.sep + dir_name
full_path_file = full_path + os.sep + file_name

# Create a laby_data dir off ~/ folder
if not os.path.exists(full_path):
    os.makedirs(full_path)

    # If ~/ not given, then folder created in cwd:
    # /tmp/ant-4975-0/laby_data
```

Laby: Ant GPS and Ant Logging system

```
M1 = "Laby Global Positioning File. "
gps = [0, 0, 0] # Initial X, Y, coordinates & Direction

with open(full_path_file, "w") as fout:
    fout.write(M1 + time.ctime() + "\n")
    s = ""
    for item in gps:
        s = s + str(item) + ", "
    fout.write(s[:-2] + "\n")
```

```
$ cat /home/ian/laby_data/gps_data
Laby Global Positioning File. Tue May 12 08:51:16 2020
0, 0, 0
```

Laby: Ant GPS and Ant Logging system

```
def gps_update(gps, action="forward"):
    # Update the gps list file [x,y,d]
    # Define constants for index into the gps list
    X = 0
    Y = 1
    D = 2
   # Direction Values: North, East, South and West
    N = 0
    E = 1
    S = 2
    W = 3
   # Escape is one step forward
    if action == "escape":
        action = "forward"
```

Laby: Ant GPS and Ant Logging system

```
if action == "forward":
    if gps[D] == N:  # North
        gps[Y] += 1
    elif gps[D] == E:  # East
        gps[X] += 1
    elif gps[D] == S:  # South
        gps[Y] -= 1
    elif gps[D] == W:  # West
        gps[X] -= 1

elif action == "left":
    gps[D] -= 1
    if gps[D] == -1:
        gps[D] = 3
```

Laby: Ant GPS and Ant Logging system

```
elif action == "right":
    gps[D] += 1
    if gps[D] == 4:
        gps[D] = 0

# Log gps list to disk as comma seperated values
with open(full_path_file, "a") as fout:
    s = ""
    for item in gps:
        s = s + str(item) + ", "
    fout.write(s[:-2] + "\n")

return gps
```

Laby: Ant GPS and Ant Logging system

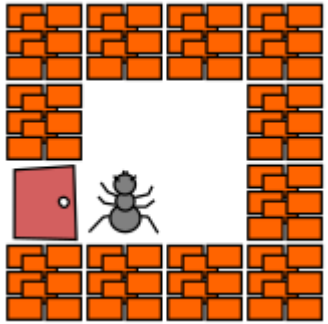
```
# Tracking option 1. - String

say("gps = [" + str(gps[0]) + ", "
              + str(gps[1]) +", "
              + str(gps[2]) + "]")
gps = [0, 1, 0]

# Tracking option 2. List comprehension. Slow output.
[say(str(i)) for i in gps]

0
1
0
```

Laby: Ant GPS and Ant Logging system

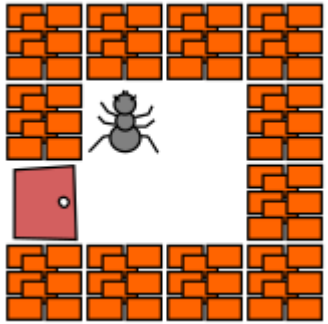
```
# Main program - Testing gps function

gps = [0, 0, 0]
gps = gps_update(gps,"forward")
gps = gps_update(gps,"right")
gps = gps_update(gps,"forward")
gps = gps_update(gps,"right")
gps = gps_update(gps,"forward")
gps = gps_update(gps,"right")
gps = gps_update(gps,"forward")
gps = gps_update(gps,"escape")
```

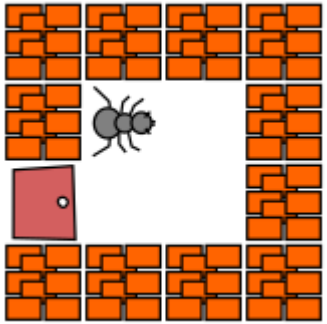
Laby: Ant GPS and Ant Logging system

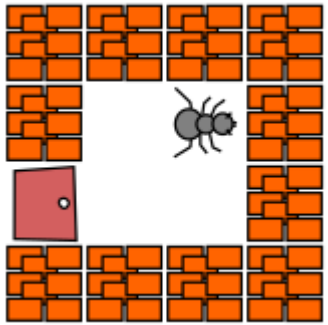
```
# Main program
# with robot
# functions
gps = [0, 0, 0]
```



```
forward()
gps = gps_update(gps,"forward")

right()
gps = gps_update(gps,"right")

forward()
gps = gps_update(gps,"forward")
```

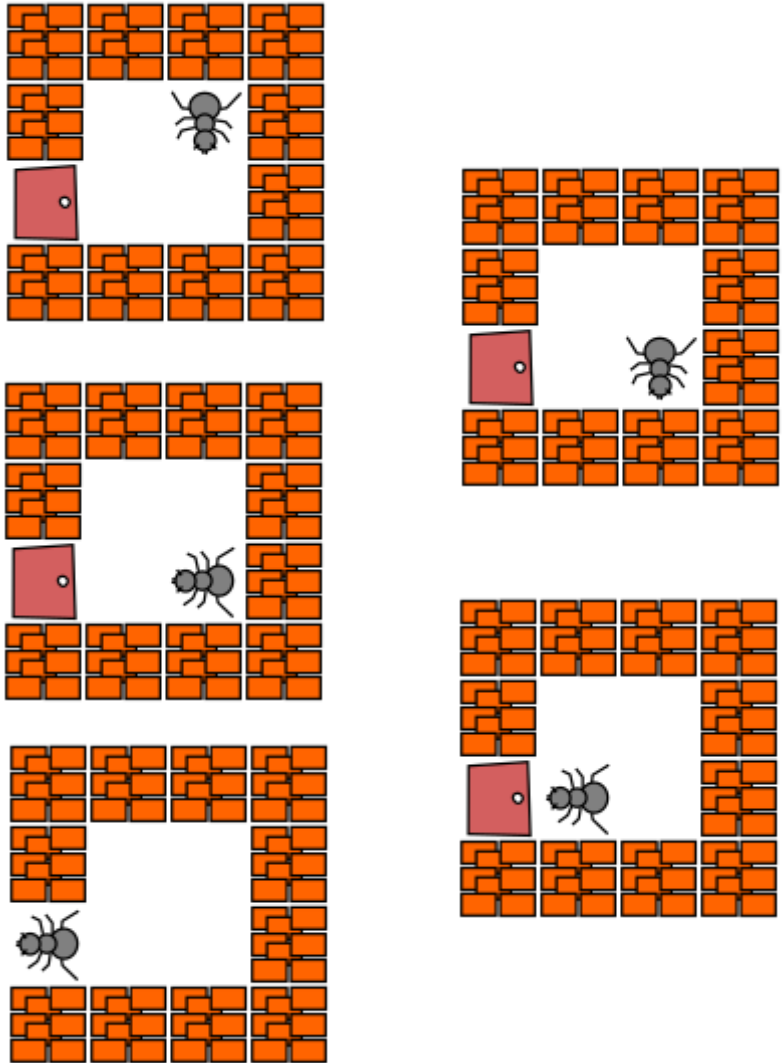






Laby: Ant GPS and Ant Logging system

```
right()
gps = gps_update(gps,"right")

forward()
gps = gps_update(gps,"forward")

right()
gps = gps_update(gps,"right")

forward()
gps = gps_update(gps,"forward")

escape()
gps = gps_update(gps,"escape")
```



Laby: Ant GPS and Ant Logging system

```
$ cat /home/ian/laby_data/gps_data
Laby Global Positioning File. Tue May 12 11:31:00 2020
0, 0, 0
0, 1, 0
0, 1, 1
1, 1, 1
1, 1, 2
1, 0, 2
1, 0, 3
0, 0, 3
-1, 0, 3
```

Ant last seen heading off into the sunset.
i.e. 3 = West

Laby. Backup of program.py 1/2

- There is no backup of your program code so adding the following will perform a backup to a local folder.

```
import time
import shutil

# Backup python program.py to local folder
ts = time.strftime("%Y-%m-%d_%H-%M-%S",time.localtime())

home_path = os.path.expanduser("~")
dir_name = "laby_data"
file_name = "program_" + ts + ".py"
full_path = home_path + os.sep + dir_name
dest_path_file = full_path + os.sep + file_name
```

Laby. Backup of program.py

```
if not os.path.exists(full_path):
    os.makedirs(full_path)

if os.path.isfile("program.py"):
    say("Performing backup...")
    shutil.copy2("program.py", dest_path_file)
    say("Backup is: " + dest_path_file)
```

```
~$ cat laby_data/program_2020-05-12_18-36-14.py
from robot import *
import os
import time
import shutil
```

Laby. Mapping the Ant Universe

If we can track the coordinates and direction of the Ant, then we should be able to collect what the Ant can see and build a map of the Ant’s universe.

In the beginning we don’t know how big is the Ant’s universe and which way the ant is facing.

Assumptions:

- Universe is less than 10 x 10
- Ant starts off facing “north”.

Laby. Mapping the Ant Universe: Create Grid

```
def create_grid():
    # Start with an empty grid - i.e. *
    string = "*"
    grid_max = 21
    offset = 10

    grid = [[string for i in range(grid_max)]
                    for j in range(grid_max)]

    # Insert the starting point in the middle at 0,0
    grid[0 + offset][0 + offset] = "s"
    return grid
```

Grid updates follow X,Y convention:
grid[X][Y] = "s"

Laby. Write out the grid the simple way.

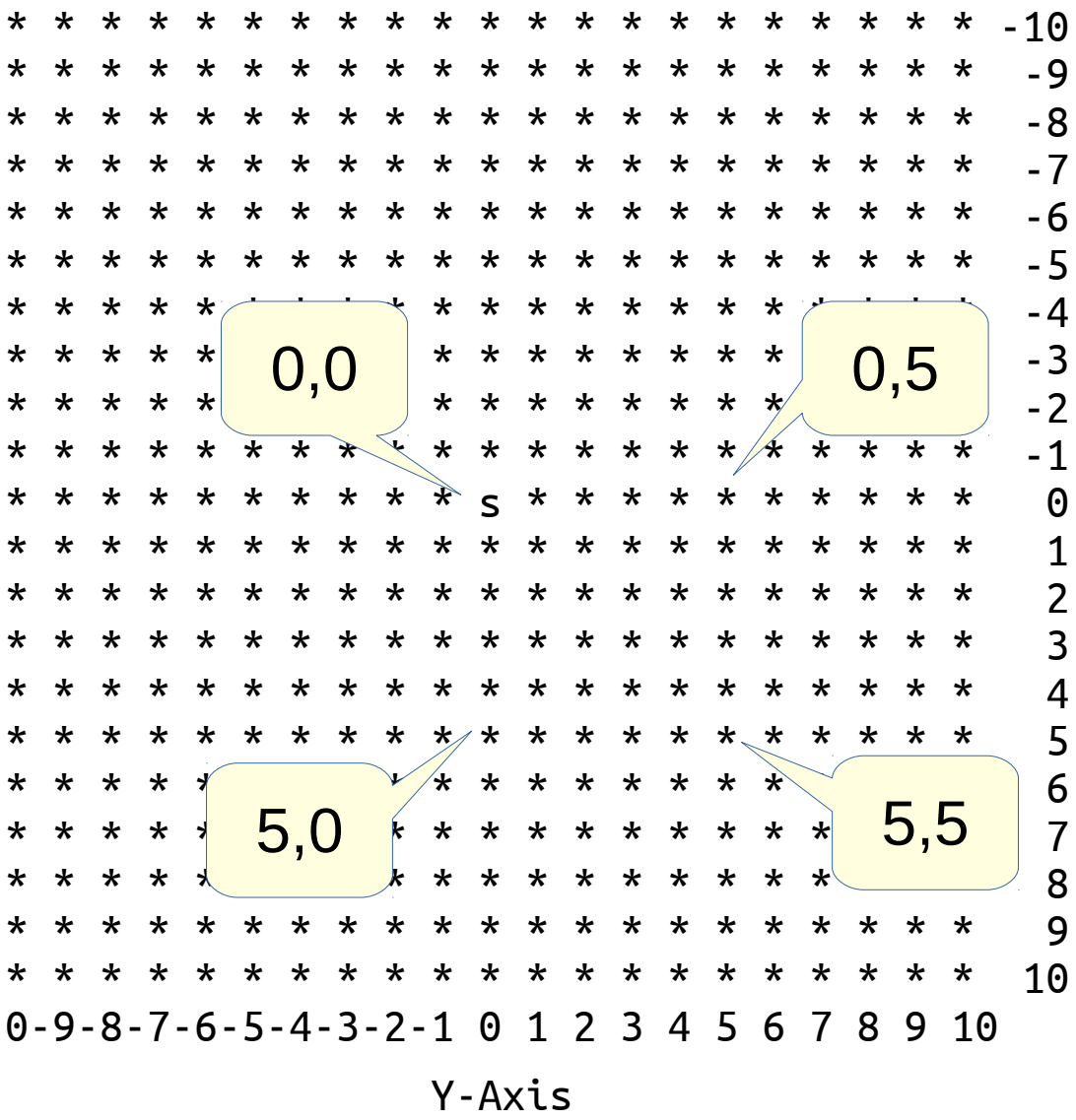
```
def write_grid(grid):
  with open("/tmp/grid", "w") as fout:
    count = -10
    for item in grid:
      s = ""
      for sub_item in item:
        s += sub_item + " "
      fout.write("{}{: >3}\n".format(s, count))
      count += 1

    row_0 = "0-9-8-7-6-5-4-3-2-1 0 1 2 3 4 5 6 7 8 9 10\n"
    fout.write(row_0)
    row_1 = "                    Y-Axis"
    fout.write(row_1)
```

Laby.
Default Grid.

Huh? 90 Degrees rotated from what is normal.

Should be turn one quadrant anticlockwise.

”s” for Start at 0,0.



Laby. Write out the grid rotated to “normal”.

```
def rotate_grid(grid):
    s = ""
    count = offset
    for k in range(grid_max):
        grid_row_list = []
        for i in reversed(range(grid_max)):
            j = (grid_max - 1) - k
            grid_row_list.append(grid[i][j])

        # Reverse the temp row list
        grid_row_list = list(reversed(grid_row_list))
        for index in range(grid_max):
            s += "{} ".format(grid_row_list[index])
```

Laby. Write out the grid rotated to “normal”.

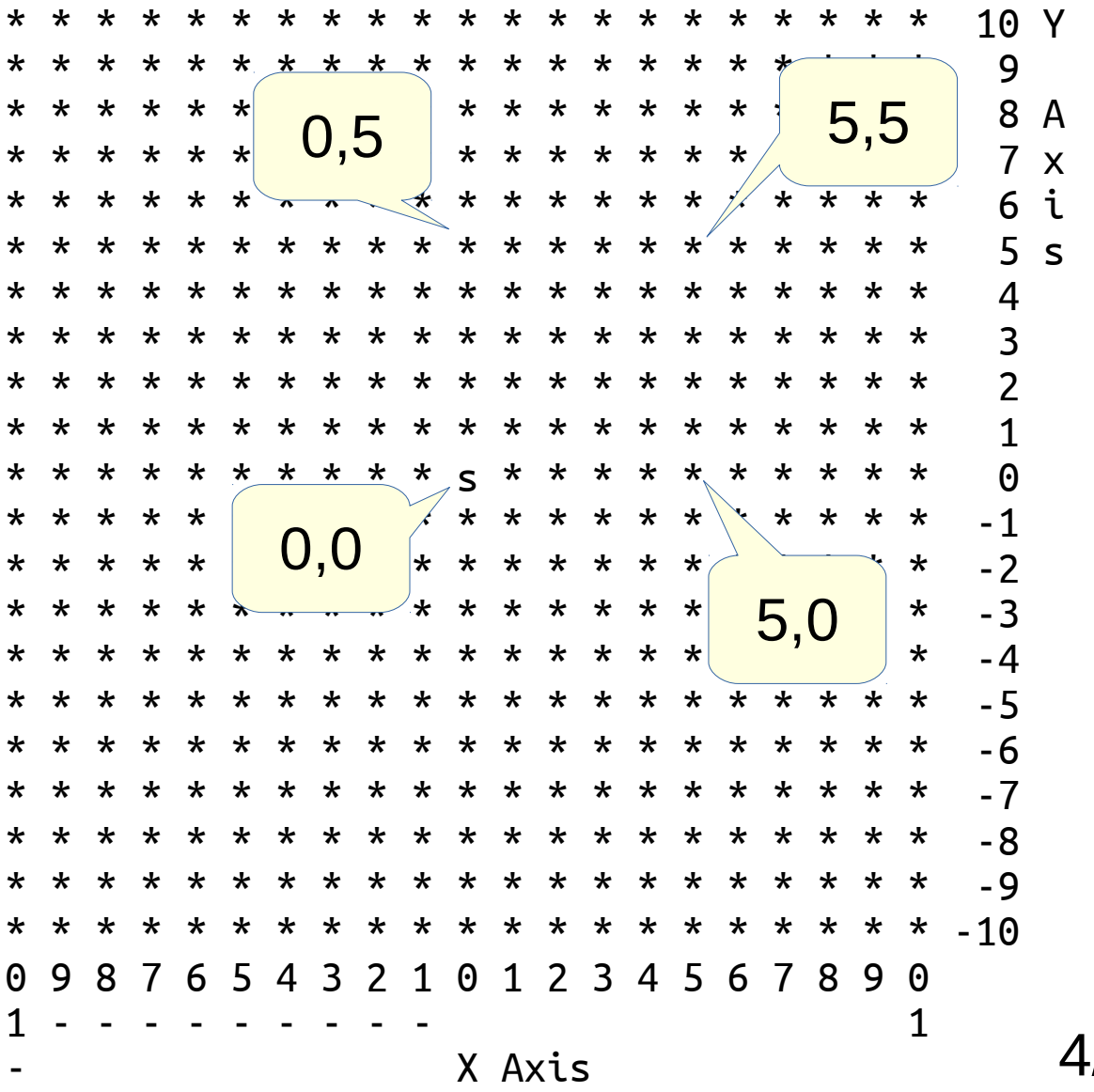
```
        # Build the Y Axis labelling
        if k < 6:
            s1 = y_axis_label[k]
        else:
            s1 = ""

        s += "{: >3} {}\n".format(count, s1)
        count -= 1

    # Build the X Axis labelling
    s += xaxis_label()
    return s
```

Laby. Write out the grid rotated to “normal”.

```
def write_grid(grid_string):
    with open("/tmp/grid", "w") as fout:
        fout.write(grid_string)


grid = create_grid()

grid_string = rotate_grid(grid)

# Write rotated grid to grid file.
write_grid(grid_string)
```

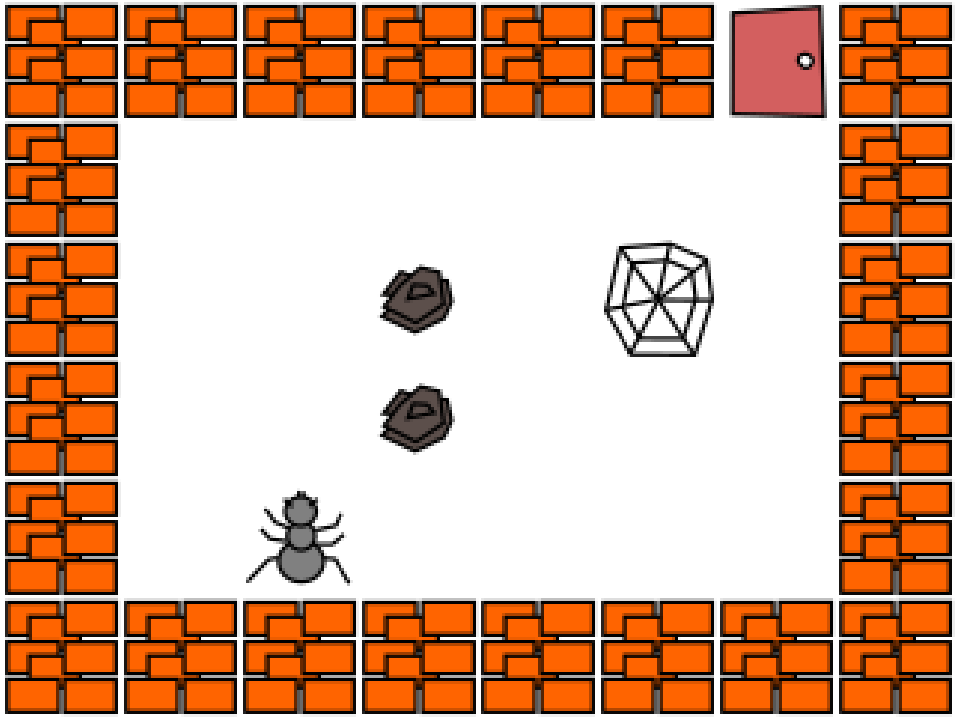
Laby. Write out the grid rotated to “normal”.

Grid is now rotated to “normal” position.



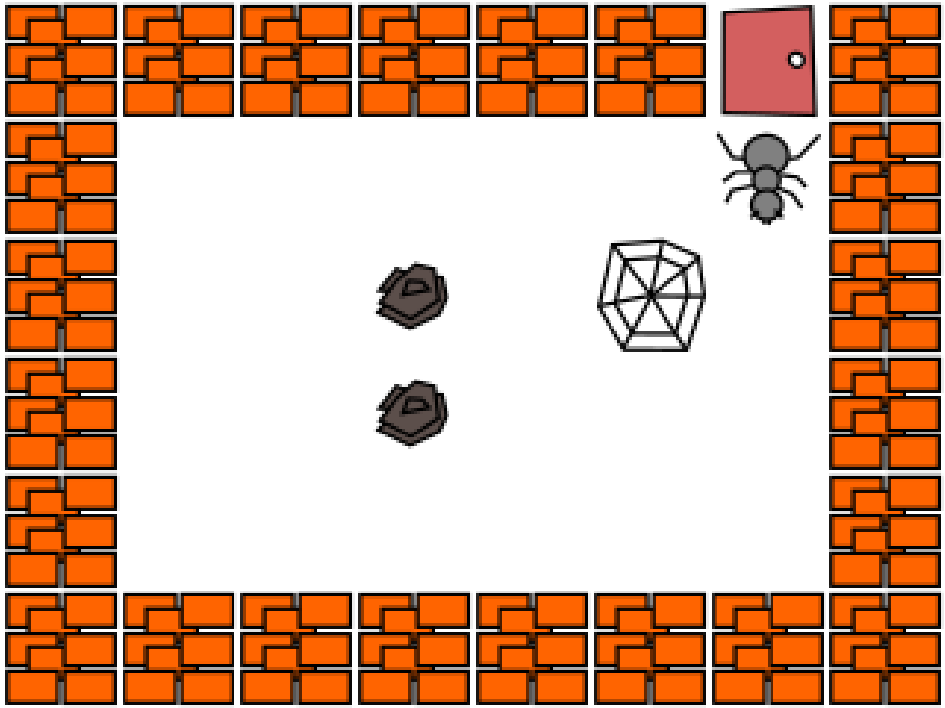
Laby. Ant Goes Exploring.

Starts here...



...explores...

Stops here...



Laby. Explorers Map

```
* * * * * * * * * * * * * * * * * * * * *   10 Y
* * * * * * * * * * * * * * * * * * * * *    9
* * * * * * * * * * * * * * * * * * * * *    8 A
* * * * * * * * * * * * * * * * * * * * *    7 x
* * * * * * * * * * * * * * * * * * * * *    6 i
* * * * * * * * * * * * * * * * * * * * *    5 s
* * * * * * * * * o o o o o x * * * * *    4
* * * * * * * * o . . . . . v o * * * *    3
* * * * * * * * o . . r . w . o * * * *    2
* * * * * * * * o . . r . . . o * * * *    1
* * * * * * * * o . s . . . . o * * * *    0
* * * *   o o o o o o * * * * *   -1
* * * *   * * * * * * * * * * * *   -2
* * * *   * * * * * * * * * * * *   -3
* * * * * * * * * * * * * * * * * * * * *   -4
* * * * * * * * * * * * * * * * * * * * *   -5
* * * * * * * * * * * * * * * * * * * * *   -6
* * * * * * * * * * * * * * * * * * * * *   -7
* * * * * * * * * * * * * * * * * * * * *   -8
* * * * * * * * * * * * * * * * * * * * *   -9
* * * * * * * * * * * * * * * * * * * * *   -10
0 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 0
1 - - - - - - - - -                     1
-                   X Axis
```

0,0

```
* * * * * * * * * *
* * o o o o o x * *
* o . . . . . v o *
* o . . r . w . o *
* o . . r . . . o *
* o . s . . . . o *
  o o o o o o * *
* * * * * * * * *
```

Enlarged

0,0

Laby. Explorers Map

```
* * * * * * * * * * * * * * * * * * * * *   10 Y
* * * * * * * * * * * * * * * * * * * * *    9
* * * * * * * * * * * * * * * * * * * * *    8 A
* * * * * * * * * * * * * * * * * * * * *    7 x
* * * * * * * * * * * * * * * * * * * * *    6 i
* * * * * * * * * * * * * * * * * * * * *    5 s
* * * * * * * * * o o o o o x * * * * *    4
* * * * * * * * o . . . . . v o * * * *    3
* * * * * * * * o . . r . w . o * * * *    2
* * * * * * * * o . . r . . . o * * * *    1
* * * * * * * * o . s . . . . o * * * *    0
* * * *   o o o o o o * * * * *   -1
* * * *   * * * * * * * * * * * *   -2
* * * *   * * * * * * * * * * * *   -3
* * * * * * * * * * * * * * * * * * * * *   -4
* * * * * * * * * * * * * * * * * * * * *   -5
* * * * * * * * * * * * * * * * * * * * *   -6
* * * * * * * * * * * * * * * * * * * * *   -7
* * * * * * * * * * * * * * * * * * * * *   -8
* * * * * * * * * * * * * * * * * * * * *   -9
* * * * * * * * * * * * * * * * * * * * *   -10
0 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 0
1 - - - - - - - - -                     1
-                   X Axis
```

0,0

```
* * * * * * * * * *
* * o o o o o x * *
* o . . . . . v o *
* o . . r . w . o *
* o . . r . . . o *
* o . s . . . . o *
  o o o o o o * *
* * * * * * * * *
```

Enlarged

0,0

Laby. Version 0.6.4-2. Only Python2 support

Supported languages – If they are installed

```
:~$ ls /usr/share/laby/mods/
c  cpp  java  js  lua  ocaml  pascal
perl  prolog  python  ruby  scheme  vala
```

Code run in Laby Program window when using python...

```
say(sys.version)
2.7.17 (default, Apr 15 2020, 17:20:14)
```

Laby. Add Python3 support

- Copy the python tree to a new python3 tree...

```
:~$ sudo cp -R /usr/share/laby/mods/python/. \
  /usr/share/laby/mods/python3
```

- Python3 tree. Laby launch page has python3 option.

```
:~$ tree /usr/share/laby/mods/python3
/usr/share/laby/mods/python3
├── help
├── lib
│   └── robot.py
├── rules
└── skel
```

Files still need to be edited to provide Python 3

- Maybe change so you don't need sudo to edit the files

```
$ sudo chmod 777 -R  /usr/share/laby/mods/python3
```

Laby. Version 3.64. Only Python2 support

Code: say(sys.version)

Message: 2.7.17 (default, Apr 15 2020, 17:20:14)

```
:/usr/share/laby/mods/python$ cat rules
info:
need python


run:
fetch   robot.py
dump program.py
spawn   python program.py
```

Laby. Edited “rules” file and restart. Now Python 3

```
:/usr/share/laby/mods/python3$ cat rules
info:
need python3

run:
fetch   robot.py
dump program.py
spawn   python3 program.py
```

Added “3”

Added “3”

Code: say(sys.version)
Message: 3.6.9 (default, Apr 18 2020, 01:56:04)

Laby. Edited “skel” file and remove semicolon

Line 1. With semicolon...

```
:~$ cat /usr/share/laby/mods/python3/skel
from robot import *;
```

With semicolon removed...

```
:~$ cat /usr/share/laby/mods/python3/skel
from robot import *
```

Removed “;”

This semicolon seemed to cause problems with adding other modules in the Laby Program window.

Laby. Re-Write robot.py for Python3:

Instead of sys.stdout.write(), sys.stdout.flush() and sys.stdin.readline(); use input() with prompt.

```
def output(s):
  sys.stdout.write(s + "\n");
  sys.stdout.flush();

def input():
  l = sys.stdin.readline();
  if l == "quit\n": exit(0);
  return l;

def left():
  output("left");
  input();
```

```
def left():
    response = input("left\n")
    return response


def right():
    response = input("right\n")
    return response


def forward():
    response = input("forward\n")
    return response
```

Laby. Re-Write robot.py for Python3:

look() function is simplified under Python 3 using input()

```
def look():
    """
    look needs to be performed twice.
    First time it may return "ok".
    Responses from look():
    void, wall, rock, web, exit, unknown
    """
    response = input("look\n")
    response = input("look\n")
    if response == "quit":
        exit(0)
    return response
```

Laby. Replace robot.py code

In /usr/share/laby/mods/python3/lib/ is the robot.py module. Using Python3 this module imports OK and appears to work reasonably well.

A replacement version of robot.py has been created. This is more PEP8 compliant and utilizes Python3 features. Refer to: https://github.com/irsbugs/laby_python3

Laby. Summary of additional functions

- Python 3 version check
- Positioning with X, Y coordinates and Directions
- Log ant movement to local CSV file storing X,Y,D.
- Backup the program.py file to a local folder.
- Drop a rock left / drop a rock right
- Explore around each position
- Create a grid map of the room

Rather than have these functions take up space in the Program window, move additional functions to the robot.py module in /usr/share/laby/mods/python3/lib

Laby. robot.py files:

Various flavours of robot.py are posted here:

https://github.com/irsbugs/laby_python3/tree/master/robot_files

Review the README.md at the above and select which file you want to become your robot.py.

Laby. Issues and Work-arounds:

- May fail to execute the last command. Add to the program code, after the last command:

```
import time
time.sleep(1)
```

- look() function may not return correct data on first attempt. Edit the robot.py file and in the look() function get it to execute the input() function twice:

```
response = input("look\n")
response = input("look\n")
```

Laby. Ant Official Intelligence:

The function ant_official_intelligence() is also included in robot.py. Using this function the ant will be able to travel through many maze designs and escape.

This function utilizes these additional functions: drop_rock(), and either check_right() or check_left()

Select your maze design, then in the Program window enter either:

```
ant_official_intelligence("right")
ant_official_intelligence("left")
```

Laby.

- Review Github repository files
  https://github.com/irsbugs/laby_python3

- Demo

Laby using Python ~ Questions / End

Other programming challenges:

- Have two adjoining rooms with all the rocks randomly located in one room. Ant must move all rocks to the other room before exiting.

- Have a big room with randomly placed rocks and webs. Ant has to effectively mow the lawns. Must move rocks out of the way to mow their square and must mow around spider webs.