Building Brains 3

Professor Doug Blank
cs.brynmawr.edu/~dblank
dblank@cs.brynmawr.edu

Know your Robot: Senses

Reading Sensors

- Light sensors
  - **getLight**(POSITION)
  - **getBright**(POSITION)
  - POSITION is either “left”, “center”, “right”, 0, 1, 2
- Infrared (IR) sensors
  - **getIR**(POSITION) - “left”, “right”, 0, 1
  - **getObstacle**(POSITION) - “left”, “center”, “right”, 0, 1, 2
- POSITION can also be “all”

Reading Sensors

- Light sensors
  - Detect the amount of light
- Infrared (IR) sensors
  - Transmits and detects Infrared signal to infer that there is an obstacle
  - The IR signal must bounce off the obstacle

Building Brains 3

- Follow a maze
- Avoid obstacles
- Go to the light
- Run away from the light

Structure of a Robot Brain

- Read sensors
- Decide what to do
- Make Movement
- Repeat

Read Sensors

leftLight = getLight(“left”)
rightLight = getLight(“right”)

rightIR = getIR(“right”)
leftIR = getIR(“left”)

Making Decisions

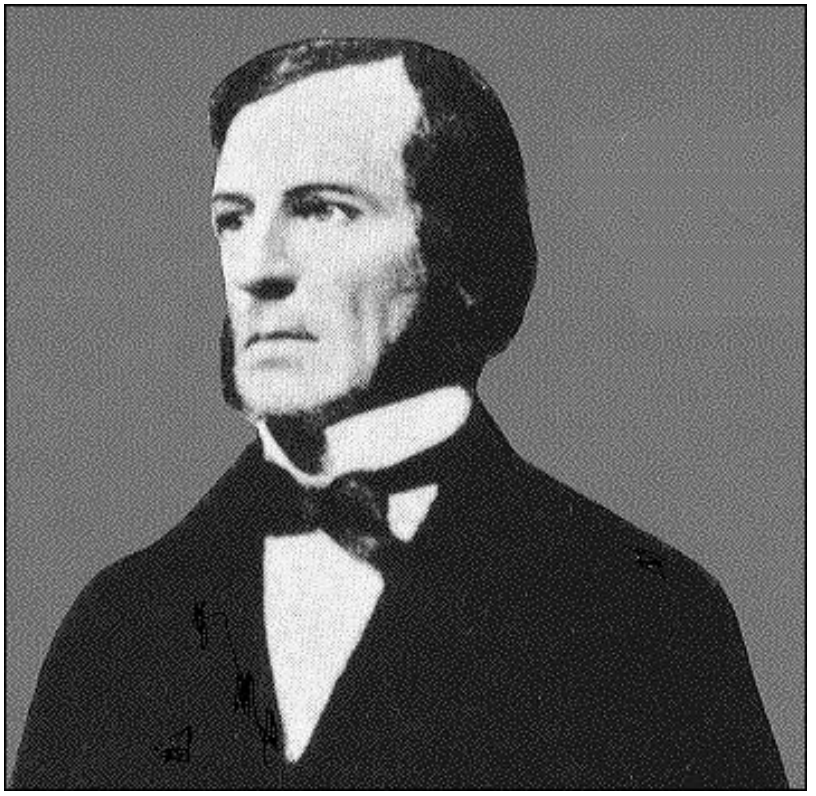
```
if (BOOLEAN-EXPRESSION):
    COMMAND
    COMMAND
    ...
```

Making Decisions

```
if (leftLight < 500):
    turnRight(1, 0.5)
```

Boolean Expressions

- Any expression that evaluates to either True or False
- Named after George Bool 1815 – 1864
- Boolean Logic, a topic for Discrete Math



Boolean Logic

Thus, if **x = horned** and **y = sheep**, then the successive acts of election represented by **x and y**, if performed on unity, give the whole of the class horned sheep. Boole showed that elective symbols of this kind obey the same primary laws of combination as algebraic symbols, whence it followed that they could be added, subtracted, multiplied and even divided, almost exactly in the same manner as numbers. Thus, **(1 – x)** would represent the operation of selecting all things in the world except horned things, that is, all not horned things, and **(1 – x) and (1 – y)** would give us all things neither horned nor sheep. By the use of such symbols propositions could be reduced to the form of equations, and the syllogistic conclusion from two premises was obtained by eliminating the middle term according to ordinary algebraic rules.

http://en.wikipedia.org/wiki/George_Boole

Boolean Expressions

(VALUE1 OPERATOR VALUE2)

```
>>> 1 < 2
True
```

Boolean Expressions

(VALUE1 OPERATOR VALUE2)

```
>>> 1 < 2
True

>>> leftLight < 500
True
>>> leftLight > 500
False
>>> leftLight == 500
False
```

IF command

```
If leftLight < 500:
    turnLeft(1, .5)
```

Making Decisions

```
if (BOOLEAN-EXPRESSION):
    COMMAND
    ...
else:
    COMMAND
    ...
```

Making Decisions

```
if (leftLight < 500):
    turnLeft(1, 1.2)
else:
    turnRight(1, 1.2)
```

Making Decisions

```
if (BOOLEAN-EXPRESSION):
    COMMAND
    ...
elif (BOOLEAN-EXPRESSION):
    COMMAND
    ...
```

Making Decisions

```
if (leftLight < 800):
    turnRight(1, .5)
elif (leftLight < 1000):
    turnRight(1, .7)
```

Making Decisions

```
if (BOOLEAN-EXPRESSION):
    COMMAND
    ...
elif (BOOLEAN-EXPRESSION):
    COMMAND
    ...
```

Boolean Expressions

- Can combine Boolean Expressions using:
  - and
  - or
- Can negate Boolean Expressions using:
  - not

Boolean Expressions

```
>>> leftLight = 2560
>>> rightLight = 30
>>> leftLight < 500 and rightLight > 500

>>> leftLight < 5000 and rightLight > 5000

>>> leftLight < 5000 and rightLight < 5000

>>> leftLight < 5000 or rightLight > 5000
```

Boolean Expressions

```
>>> leftLight = 2560
>>> rightLight = 30
>>> leftLight < 500 and rightLight > 500
False
>>> leftLight < 5000 and rightLight > 5000
False
>>> leftLight < 5000 and rightLight < 5000
True
>>> leftLight < 5000 or rightLight > 5000
True
```

Boolean Logic: And

- (True and True) is True
- (True and False) is False
- (False and True) is False
- (False and False) is False

Boolean Logic: Or

- (True or True) is True
- (True or False) is True
- (False or True) is True
- (False or False) is False

Or

- You could think of And and Or as Functions:

```
def Or(value1, value2):
    if value1:
        return True
    elif value2:
        return True
    else:
        return False
```

Boolean Functions

- Functions can return Boolean values

```
def obstacleInFront():
    if getIR(“left”) or getIR(“right”):
        return True
    else:
        return False

if obstacleInFront():
    turnAround()
```

Boolean Functions

- Functions can return Boolean values

```
def obstacleInFront():
    return (getIR(“left”) or getIR(“right”))

if obstacleInFront():
    turnAround()
```

Repeat

We've seen that Python's “for” command allows you to repeat an indented region N times

But what is you wanted to do something forever?

Repeat

We've seen that Python's “for” command allows you to repeat an indented region N times

But what is you wanted to do something forever?

Infinite Loop

Infinite Loop

```
while True:
    COMMAND
    ...
```

Infinite Loop

```
while True:
    speak(“Hello”)
```

Structure of a Robot Brain

- Read sensors
- Decide what to do
- Make Movement
- Repeat

Structure of a Robot Brain

```
while True:
    left = getLight(“left”)
    right = getLight(“right”)
    if left < right:
        turnLeft(1, .4)
    else:
        turnRight(1, .4)
```