Reinforcement & Applications

Andey Robins & Dr Borowczak

November 29 & December 1, 2022

Outline

Outline

- Schedule
- Monty Hall Simulations
- Giant Rat Breeding
- Fermi Paradox
- Micro:Bits
- The Next Steps

Class Schedule

Week	Tuesday	Thursday	Assignments
Nov 28	Reinforcement	Reinforcement	Lab
Dec 5	AMA & Pizza	Review & QnA	Final Quest Retakes
Dec 13	Final Quest	Comprehensive	10:15 AM

Final Quest will replace your lowest graded assignment.

Reinforcement

Today's Plan

Like the Thursday before break, we'll be exploring applications and uses of python!
Impractical Python Projects

Projects

1. Monty Hall Problem
2. Breed Bulldog sized Rats
3. Model the Fermi Paradox

Monty Hall Problem

A showcase of the Monty Hall problem

1. Contestants are shown three doors
2. The contestant chooses a door
3. One of the unselected doors is opened to reveal a goat
4. The contestant can switch doors or keep their selection
5. If they end up opening the door with the car, they win!

Optimal Strategy

1. Choose a door
2. After a door is revealed, switch your door
3. Win 2/3 of the time.

The Controversy

A popular "Ask column" provided the correct solution (i.e. switch) as an answer in 1990, but was met with major push back from the general public.
We'll experimentally prove the answer by simulating many many instances of the problem.

Simulation

```
for i in range(1000):
  doors = setup_doors()
```

```python
def setup_doors():
  car_idx = random.randint(0, 2)
  doors = ["goat", "goat", "goat"]
  doors[car_idx] = "car"
  return doors
```

```python
def choose_door():
  return random.randint(0, 2)
```

Why did we put this in a function?

```python
def reveal_door(doors, choice):
  if choice == 0:
    if doors[1] == "car":
      return 2
    else:
      return 1
  elif choice == 1:
    if doors[0] == "car":
      return 2
    else:
      return 0
  else: # choice == 2
    if doors[0] == "car":
      return 1
    else:
      return 0
```

```python
def switch_wins(doors, revealed, choice):
  doors_idxs = [0, 1, 2]
  if revealed > choice:
    doors_idxs.pop(revealed)
    doors_idxs.pop(choice)
  else:
    doors_idxs.pop(choice)
    doors_idxs.pop(revealed)
  return doors[doors_idxs[0]] == "car"
```

```python
def stay_wins(doors, revealed, choice):
  return doors[choice] == "car"
```

Simulation Results

```
$ python3 monte.py
Welcome to the Monty Hall Solution Simulator
The game will be run 1000 times for staying and switching.
Beginning simulation...
Finished simulation. Results to follow.
The win rate of staying is 0.334
The win rate of switching is 0.666
```

Monty Hall Math

A visual demonstration of the answer to the Monty hall problem

Genetically Breeding Giant Rats

I feel like a mad scientist today, and I've decided that I'd like to create a race of super-mutant, giant rats. To see if it's even feasible, I'd first like to mode the outcomes in python to determine if we can actually do this (because that's the only challenge we need to overcome in this scenario).

Strategy

The Rat Breeding Strategy

Information about the brown rat

Parameters for our program

Simulation

```
def main():
  # create initial population
  generations = 0
  parents = populate(NUM_RATS, INITIAL_MIN_WT,
                     INITIAL_MAX_WT, INITIAL_MODE_WT)
  pop_fitness = fitness(parents, GOAL)

  # create new generations
  while pop_fitness < 1 and generations < GENERATION_LIMIT
    selected_rats = select(parents, NUM_RATS)
    children = breed(selected_rats, LITTER_SIZE)
    children = mutate(children, MUTATE_ODDS,
                      MUTATE_MIN, MUTATE_MAX)
    parents = selected_rats + children
    pop_fitness = fitness(parents, GOAL)
    generations += 1
```

```
def populate(num_rats, min_wt, max_wt, mode_wt):
  rats = []
  while num_rats > 0:
    rats.append(random.triangular(min_wt, max_wt, mode_wt)
    num_rats -= 1
  return rats
```

```python
def fitness(population, goal):
  avg = statistics.mean(population)
  return avg / goal
```

```python
def select(population, num_to_retain):
  sorted_population = sorted(population)
  return sorted_population[-num_to_retain:]
```

```python
def breed(rats, litter_size):
  small_pop = rats[int(len(rats)/2):]
  random.shuffle(small_pop)
  large_pop = rats[:int(len(rats)/2)]
  random.shuffle(large_pop)

  children = []
  for rat1, rat2 in zip(small_pop, large_pop):
    for child in range(litter_size):
      if rat1 < rat2:
        child = random.randint(int(rat1), int(rat2))
      else:
        child = random.randint(int(rat2), int(rat1))
      children.append(child)

  return children
```

```python
def mutate(children, mutate_odds, mutate_min, mutate_max):
  for idx, rat in enumerate(children):
    if mutate_odds >= random.random():
      children[idx] = round(rat *
                  random.uniform(mutate_min, mutate_max))
  return children
```

Simulation Results

```
...
Generation 355 fitness = 0.8798
Generation 356 fitness = 0.8829
Generation 357 fitness = 0.8878
Generation 358 fitness = 0.9170
Generation 359 fitness = 0.9730
Generation 360 fitness = 1.0119

number of generations = 361
number of years = 36
```

Fermi Paradox

Fermi Paradox

Since the Universe is $\sim$13 billion years old, and even a modestly space faring civilization could explore the entire milky way galaxy in $\sim$10 million years. The radio bubbles of these groups would likely be larger than their explored space, so: *where are all the aliens?*

A spiral galaxy similar to the Milky Way

Drake Equation

A modeling of the drake equation

Constant Values

Fermi Constant Estimates

Calculating the Drake Equation

```
R_STAR = 3
F_P = 1
N_E = 0.2
F_L = 0.13
F_I = 1
F_C = 0.2
L = 10 ** 9

def drake_estimation():
  return R_STAR * F_P * N_E * F_L * F_I * F_C * L

print(int(drake_estimation()))
```

```
15,600,000
```

Using these estimates, they place an approximate estimation for the number of detectable civilizations at ~15.6 million.

Note: detectable in this context means that they could be detected in the MWG not they could be detected from Earth

Radio Bubbles

Assuming that an alien civilization would only be leaking incidental radio waves (i.e. not broadcasting their existence into the universe), their presence would be defined by a "radio bubble."
This is a "bubble" approximately 200 LY across centered around the civilization's home planet.
We will model the galaxy as a collection of radio cubes, a very loose estimate of a radio bubble.

Simulation

```
def main():
  cube_count = radio_cubes_in_galaxy(RADIO_BUBBLE_RADIUS)
  civilization_count = drake_estimation()
  locations = []
```

```python
def drake_estimation():
  return int(R_STAR * F_P * N_E * F_E * F_I * F_C * L)
```

```python
def radio_cubes_in_galaxy(radio_bubble_radius):
  galaxy_volume = 3.14 * 50_000 * 50_000 * 1_000
  bubble_volume = 3.14 * (4/3) * radio_bubble_radius
  return galaxy_volume / bubble_volume
```

```
for i in tqdm(range(civilization_count)):
  locations.append(random.randint(1, cube_count))

counts = {}
for val in tqdm(locations):
  if val in counts:
    counts[val] += 1
  else:
    counts[val] = 1

detected_civs = 0
for val in tqdm(counts.values()):
  if val > 1:
    detected_civs += val
```

Simulation Results

```
Populating galaxy
100%|==========================| 15600000/15600000
Counting civs per cube
100%|==========================| 15600000/15600000
Counting overlapping civs
```

Questions

Micro:Bits

A Micro:Bit

What's In a Micro:Bit

- Bluetooth
- Accelerometer
- Pin Out
- Buttons
- Light sensor

Writing a student quiz application

1. Students select an answer
2. Teacher gets feedback on proportion of each answer
3. Teacher can push out a correct answer
4. Teacher can lock micro:bits
5. Students get individual feedback on if they're correct

Student Code

Track the following: - the state of the system (selecting answer, waiting for correct, locked, right, wrong) - the selected answer - expected feedback

State machine has the following states:

- Choosing an answer (state 0)
- Waiting for the teacher (state 1)
- Answer right or wrong (state 1.5)
- Locked (state 2)

```python
def on_button_pressed_a():
    global selected_answer
    selected_answer += -1
    wrapNumber()
input.on_button_pressed(Button.A, on_button_pressed_a)

def on_button_pressed_b():
    global selected_answer
    selected_answer += 1
    wrapNumber()
input.on_button_pressed(Button.B, on_button_pressed_b)
```

```python
def wrapNumber():
    global selected_answer
    while selected_answer > 5:
        selected_answer += -5
    while selected_answer < 0:
        selected_answer += 5
```

```python
def on_button_pressed_ab():
    global state, correct
    radio.send_number(selected_answer)
    state = 1
    correct = 0
input.on_button_pressed(Button.AB, on_button_pressed_ab)
```

```python
def on_forever():
    if state == 0:
        showAnswerLetter(selected_answer)
    if state == 1:
        if correct == 0:
            # show question mark
            basic.show_leds("""...""")
        elif correct == 1:
            basic.show_icon(IconNames.YES)
        else:
            basic.show_icon(IconNames.NO)
    if state == 2:
        # show blank screen
        basic.show_leds("""...""")
basic.forever(on_forever)
```

```python
def on_received_string(receivedString):
    global selected_answer, correct, state
    selected_answer = 0
    correct = 0
    if receivedString == "next question":
        state = 0
    if receivedString == "lock":
        state = 2
radio.on_received_string(on_received_string)
```

```
def on_received_value(name, value):
    global state, correct
    state = 1
    if value == selected_answer:
        correct = 1
    else:
        correct = 2
radio.on_received_value(on_received_value)
```

Everything Else

```
correct = 0
state = 0
selected_answer = 0
radio.set_group(1)
```

That's the entire program! You'll notice a couple things:

- There isn't a main function
- We "register" functions
- Lots of global variables!

Teacher Code

Track the following:

- Total student answers for each option
- Correct answer

```python
def on_received_number(receivedNumber):
    answers[receivedNumber] = answers[receivedNumber] + 1
radio.on_received_number(on_received_number)
```

```python
def showGraph():
    global bar_pixels, bar_height
    getMax()
    for index2 in range(6):
        answers[index2] = answers[index2] * 5
    for index3 in range(6):
        bar_pixels = answers[index3] / answer_max
        bar_height = 0
        while bar_height <= bar_pixels:
            led.plot(index3, 5 - bar_height)
            bar_height += 1
    for index4 in range(6):
        answers[index4] = answers[index4] / 5
```

```python
def on_gesture_shake():
    radio.send_string("lock")
input.on_gesture(Gesture.SHAKE, on_gesture_shake)

def on_forever():
    showGraph()
basic.forever(on_forever)
```

Other code looks very similar or identical to what we've presented for the student code, so I won't repeat it here. The entire program is available through Codio in the file `teacher.py`

Micro:Bots

Change of plans

996e8000-ac48-11ea-9e71-3469a6bb9d4f

The Next Steps

This has been a great start to your computer science journey, but how do you move forward from here?

- Studying/Practice
- Courses
- Learning new Languages

How to Practice

We've mentioned before, but just writing code is often the best way to practice! Fortunately, there are a lot of great places you can go to practice programming problems at various levels!

kattis.com

codingame.com

projecteuler.net

adventofcode.com

Courses

UWyo COSC Courses

How to Learn a New Language

1. Learn the basic, atomic pieces
2. Solve some basic problems
3. Learn the tools
4. Build a medium project and “release” it
5. Read about and implement best practices

```cpp
#include <iostream>
int main() {
  int max = 0;
  bool show_text = false;
  int my_nums = { 0, 2, 3, 50, 1};

  if (show_text)
    std::cout << "The array: ";

  // ....
```

```
 // ....
 for (auto num: my_nums) {
  if (num > max)
   max = num;
  if (show_text)
   std::cout << num << " ";
 }
 if (show_text) {
  std::cout << std::endl << "Max number: "
       << max << std::endl;
 }
}
```

Tools

Instead of the interpreter `python3` we use the compiler `gcc`.
Instead of pip, we use "header files."
Instead of `import` we say `include`.

Releasing Something

2030 Bloodsugar project

Best Practices

Questions