

Reinforcement & Applications

Andey Robins & Dr Borowczak

April 20, 2023

Outline

- ▶ Monty Hall Simulations
- ▶ Giant Rat Breeding
- ▶ Fermi Paradox
- ▶ The Next Steps

Today's Plan

We'll be exploring applications and uses of python!

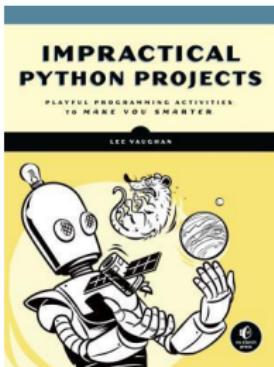


Figure 1: Impractical Python Projects

Projects

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

Monty Hall Problem

Monty Hall Problem

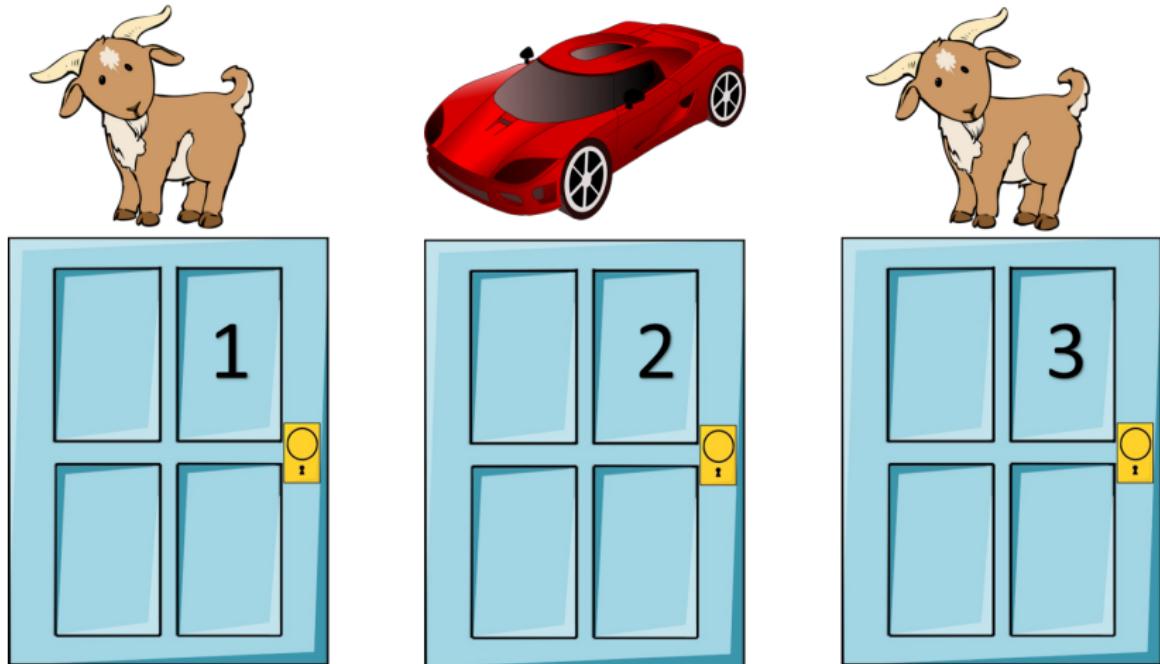


Figure 2: A showcase of the Monty Hall problem

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 $\frac{2}{3}$ 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()
    choice = choose_door()
    revealed_door = reveal_door(doors, choice)

    if switch_wins(doors, revealed_door, choice):
        switch_win_count += 1

    if stay_wins(doors, revealed_door, choice):
        stay_win_count += 1
```

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

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

Why did we put this in a function?

```
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
```

```
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"
```

```
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.
```

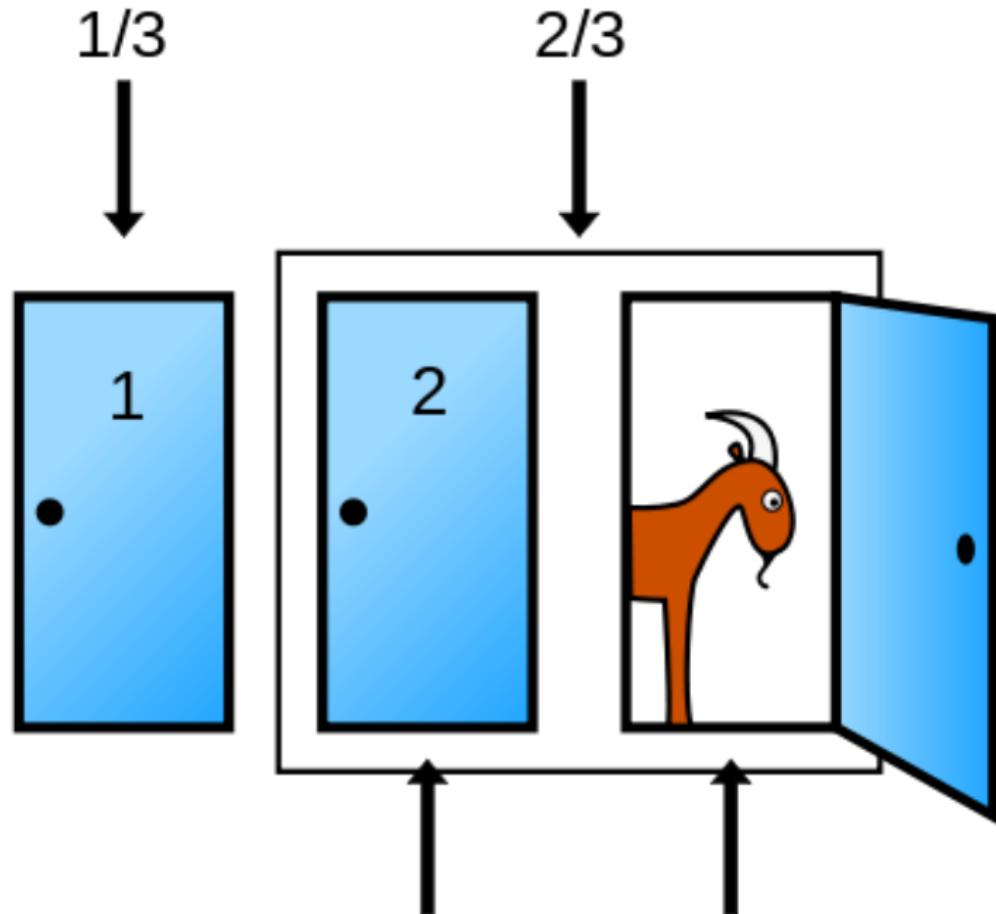
```
Beginning simulation...
```

```
Finished simulation. Results to follow.
```

```
The win rate of staying is 0.334
```

```
The win rate of switching is 0.666
```

Monty Hall Math

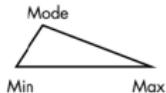
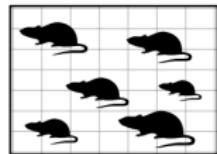


Giant Rats

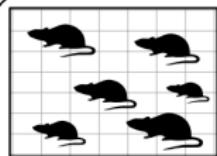
Genetically Breeding Giant Rats

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).

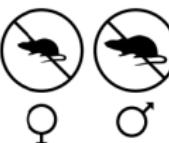
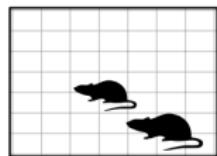
Strategy



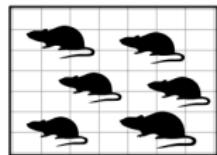
Populate: Establish initial population and range of weights.



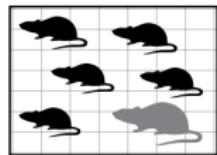
Grade: Evaluate fitness by comparing mean population weight to a target weight.



Select: Cull smallest males and females.



Breed: Repopulate with random weights based on weight range of the selected rats.



Mutate: Randomly alter weights on a few rats. Most outcomes reduce weight.

Table 7-1: Brown Rat Weight and Breeding Statistics

Parameter	Published values
Minimum weight	200 grams
Average weight (female)	250 grams
Average weight (male)	300–350 grams
Maximum weight	600 grams*
Number of pups per litter	8–12
Litters per year	4–13
Life span (wild, captivity)	1–3 years, 4–6 years

*Exceptional individuals may reach 1,000 grams in captivity.

Figure 5: Information about the brown rat

Table 7-2: Input Assumptions for the Super-Rats Genetic Algorithm

Variable and value	Comments
GOAL = 50000	Target weight in grams (female bullmastiff)
NUM_RATS = 20	Total number of <i>adult</i> rats your lab can support
INITIAL_MIN_WT = 200	Minimum weight of adult rat, in grams, in initial population
INITIAL_MAX_WT = 600	Maximum weight of adult rat, in grams, in initial population
INITIAL_MODE_WT = 300	Most common adult rat weight, in grams, in initial population
MUTATE_ODDS = 0.01	Probability of a mutation occurring in a rat
MUTATE_MIN = 0.5	Scalar on rat weight of least beneficial mutation
MUTATE_MAX = 1.2	Scalar on rat weight of most beneficial mutation
LITTER_SIZE = 8	Number of pups per pair of mating rats
LITTERS_PER_YEAR = 10	Number of litters per year per pair of mating rats
GENERATION_LIMIT = 500	Generational cutoff to stop breeding program

Figure 6: 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
```

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

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

```
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
```

```
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?*



Figure 7: A spiral galaxy similar to the Milky Way

Drake Equation

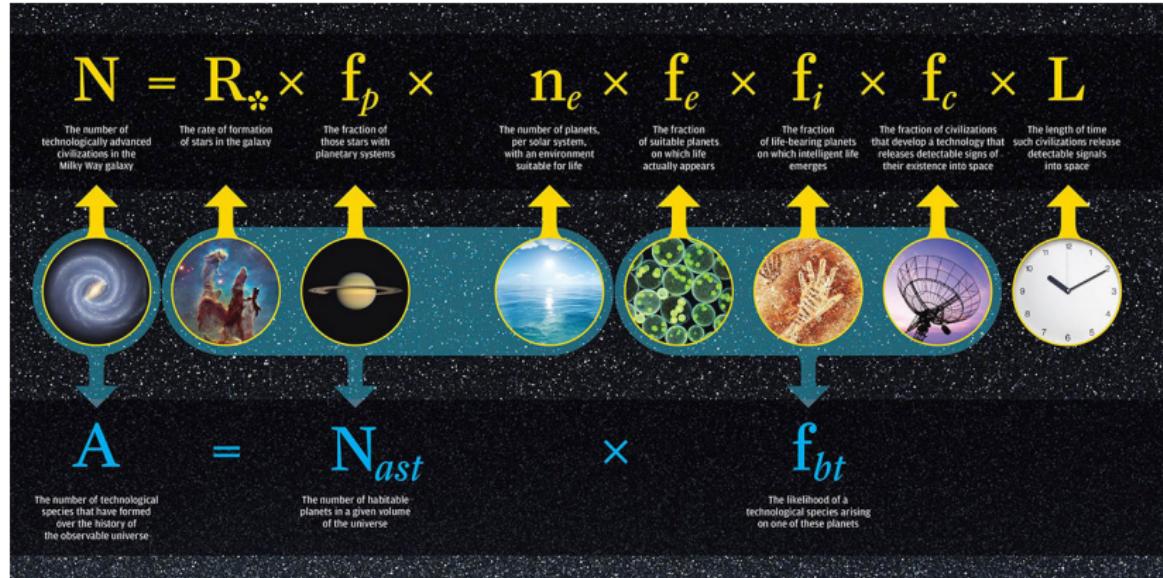


Figure 8: A modeling of the drake equation

Constant Values

Table 10-1: Some Drake Equation Inputs and Results

Parameter	Drake 1961**	Drake 2017	Your choices
R^*	1	3	
f_p	0.35	1	
n_e	3	0.2	
f_l	1	0.13	
f_i	1	1	
f_c	0.15	0.2	
L	50×10^6	1×10^9	
N	7.9×10^6	15.6×10^6	

**midpoint of ranges shown

Figure 9: 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 = []

# simulation to follow
```

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

```
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

100% |=====| 15593457/15593457

The number of civs who could have
detected each other = 13084

This is a rate of 0.0839%

In other words, the chance of a civilization being in a position to
detect another is 1 in 1750.

Questions

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!



WHY KATTIS CONTACT

A COMMUNITY FOR GREAT PROGRAMMERS

We offer a wide variety of coding challenges, a great community, a global ranking list and awesome jobs at worldwide companies. Sounds like your cup of tea?

For companies

For problem solvers

For universities

A screenshot of the Kattis website. At the top, there's a navigation bar with links for PROBLEMS, CONTESTS, DUELIST, RANKLISTS, JOBS, HELP, and ADMIN. A search bar and a user profile for 'Pekka Niemelä' are also present. Below the navigation, there's a section titled 'Welcome to the Kattis Problem Archive' featuring a cartoon cat at a podium. To the right, there's a 'Ranklist' table showing user scores. The table has columns for RANK, USER, and SCORE. The top few entries are: 920 (Pekka Niemelä, 25.0), 921 (24.9), 924 (24.7), 926 (24.6), 932 (24.5), 936 (Pekka Niemelä, 24.4), and 941 (24.3).

RANK	USER	SCORE [!]
920	Pekka Niemelä (>2 others)	25.0
921	(>2 others)	24.9
924	(>1 other)	24.7
926	(>5 others)	24.6
932	(>3 others)	24.5
936	Pekka Niemelä (>4 others)	24.4
941	(>1 other)	24.3

Figure 10: kattis.com

STEP UP YOUR CODING GAME

The new way to improve your programming skills while having fun and getting noticed

[GET STARTED](#)

We use cookies to analyze our site traffic and improve the quality of our services.
You can edit your cookie options at any time. [Find out more.](#)

[ACCEPT](#)[REFUSE](#)[CONFIGURE](#)

Figure 11: codingame.com

About Project Euler

What is Project Euler?

Project Euler is a series of challenging mathematical/computer programming problems that will require more than just mathematical insights to solve. Although mathematics will help you arrive at elegant and efficient methods, the use of a computer and programming skills will be required to solve most problems.

The motivation for starting Project Euler, and its continuation, is to provide a platform for the inquiring mind to delve into unfamiliar areas and learn new concepts in a fun and recreational context.

Who are the problems aimed at?

The intended audience include students for whom the basic curriculum is not feeding their hunger to learn, adults whose background was not primarily mathematics but had an interest in things mathematical, and professionals who want to keep their problem solving and mathematics on the cutting edge.



Figure 12: projecteuler.net

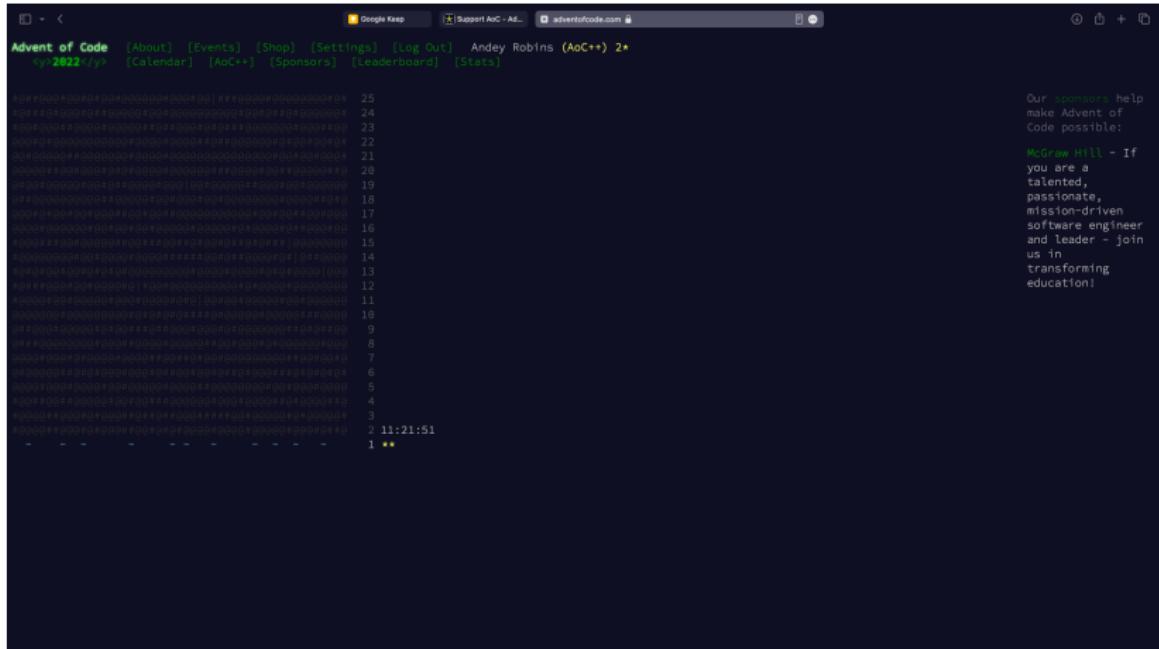


Figure 13: adventofcode.com

Courses

FRESHMAN YEAR								
COSC 1010	Intro to Comp Science	3	C		COSC 1030	Computer Science I	4	C
MATH 2200	Calculus I	Q	4	C	Prerequisite:	<i>C in COSC 1010</i>		
<i>Prerequisite: C in Math 1405 or 1450, MPE 5, Math ACT 27, Math SAT 640</i>					MATH 2205	Calculus II	Q	C
Science Elective- see list		PN	4	D	Prerequisite:	<i>C in Math 2200</i>		
USP: First Year Seminar		FYS	3	C	Science Elective- see list		PN	D
Total		14			USP: Communications I		C1	C
					Total		15	
SOPHOMORE YEAR								
COSC 2150	Computer Organization	3	C		COSC 3011	Intro to Software Design	3	C
Prerequisite:	<i>C in COSC 1030</i>				Prerequisite:	<i>C in COSC 2030</i>		
COSC 2030	Computer Science II	4	C		COSC 3020	Alg & Data Structures	4	C
Prerequisite:	<i>C in COSC 1030</i>				Prerequisite:	<i>C in COSC 2030 and COSC 2300</i>		
COSC 2300	Discrete Structures	3	C		MATH 2250	Elementary Linear Algebra	3	C
Prerequisite:	<i>C in COSC 1030 and MATH 2200</i>				Prerequisite:	<i>C in Math 2200 or Math 2350</i>		
USP: Human Culture		H	3	D	USP: Human Culture		H	D
USP: Communications II		C2	3	C	General Elective		3	D
Prerequisite:	<i>C in C1</i>				Total		16	
Total		16					Total	
JUNIOR YEAR								
COSC 3015	Functional Programming	3	C		COSC 3050	Ethics for Computer Prof	1	C
Prerequisite:	<i>C in COSC 2030 and COSC 2300</i>				Prerequisite:	<i>Junior Standing, COSC Major</i>		
COSC	Systems Course (4760 or 4820)	3	C		COSC	Theory Course (4100 or 4200)	3	C
Prerequisite:	<i>Varies</i>				Prerequisite:	<i>Varies</i>		
COSC 3765	Computer Security	3	C		COSC	COSC Elective I	3	C
Prerequisite:	<i>COSC 2030</i>				Prerequisite:	<i>Varies</i>		
Math/Science Elective - see list		4	C		General Elective		3	D
USP: US & Wyo Const.		V	3	D	General Elective		3	D
Total		16			Total		13	

Figure 14: 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

```
#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

The screenshot shows a GitHub repository page for the user 'andey-robins' named 'bloodsugar'. The repository is public. The main navigation bar includes links for Pull requests, Issues, Codespaces, Marketplace, and Explore. Below the bar, there are buttons for Pin, Watch (0), Fork (0), and Star (0). The repository name 'andey-robins / bloodsugar' is displayed with a Public badge. The 'Code' tab is selected, showing a list of files and their commit history. The commit history starts with a merge pull request from 'andeyt/overflow' on Oct 25, 2018, containing 22 commits. The list of files includes 'COSC_2030_Project_1.pdf', 'LICENSE', 'LinkedList.cpp', 'LinkedList.h', 'Node.cpp', 'Node.h', 'README.md', 'bloodsugar.cpp', and 'bloodsugar.out'. Each file entry shows its name, a brief description, and the date it was last modified (4 years ago). To the right of the code list, there are sections for 'About' (no description, website, or topics provided), 'Readme' (MPL-2.0 license), 'Watching' (0 watching), 'Forks' (0 forks), 'Releases' (no releases published, Create a new release), and 'Packages'.

andey-robins / bloodsugar Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file <> Code

andey-robins Merge pull request #7 from andeyt/overflow 2c9cf88 on Oct 25, 2018 22 commits

COSC_2030_Project_1.pdf Add Project Requirements File 4 years ago

LICENSE Initial commit 4 years ago

LinkedList.cpp Finish overflow trapping 4 years ago

LinkedList.h Finish overflow trapping 4 years ago

Node.cpp Add comments/documentation 4 years ago

Node.h Add comments/documentation 4 years ago

README.md Update README.md 4 years ago

bloodsugar.cpp Output weekly overflow 4 years ago

bloodsugar.out Finish overflow trapping 4 years ago

About

No description, website, or topics provided.

Readme

MPL-2.0 license

0 stars

0 watching

0 forks

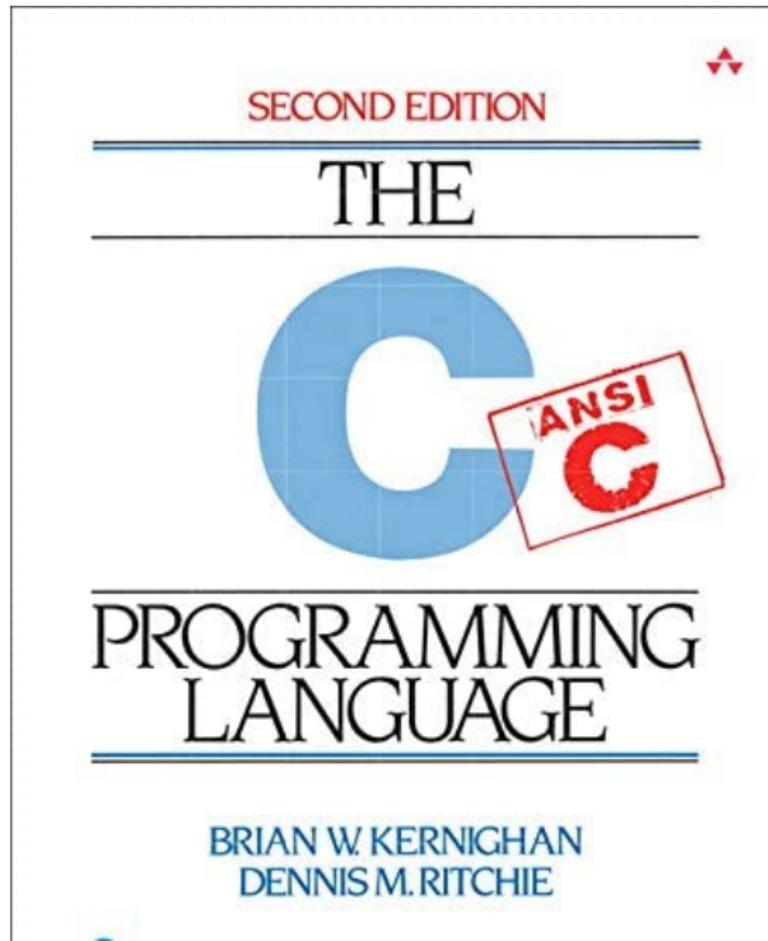
Releases

No releases published

Create a new release

Packages

Figure 15: 2030 Bloodsugar project



Questions