Computing with Python

CMPT 145

Copyright Notice

Learning Objectives

After studying this chapter, a student should be able to:

- Make use of lists to solve computational problems.
- Make use of dictionaries to keep data organized.
- Import Python modules and use functions defined in them.
- Open a datafile, and place the contents of the datafile into a list.

Sorting a list

- Assume an unorganized list of numeric values
- Produce a new list with the values in ascending order

Sorting a list

```
1  unsorted = [3, 2, 5, 7, 6, 8, 0, 1, 2, 8, 2]
2  sorted = list()
3
4  while len(unsorted) > 0:
5      out = min(unsorted)
6      unsorted.remove(out)
7      sorted.append(out)
8
9  print(sorted)
```

Sorting a list

- Does the script work?
- Is it a good program?
- How can you answer these questions?

Counting Primes: Prime Numbers

- A prime number is a positive integer evenly divisible only by 1 and itself
  - e.g. 4, 6, 8, 9 are not prime
  - e.g. 2, 3, 5, 7 are prime
  - By *definition*, 0, 1 are not considered prime.
- Question: How many of the numbers $1, \ldots, n$ are prime?

Counting Primes: Script 1

```
1  n = 100000  # end of range of numbers to check for primes
2
3  count = 0
4  for i in range(2, n + 1):
5      no_factors_found = True  # assume prime until disproven
6      f = 2
7      # check if i is prime by checking remainders
8      while no_factors_found and f < i:
9          if i % f == 0:
10             no_factors_found = False
11         f += 1
12
13     if no_factors_found:
14         count += 1
15
16 print("# Prime numbers between 2 and " + str(n) + ":", count)
```

Counting Primes: Script 1

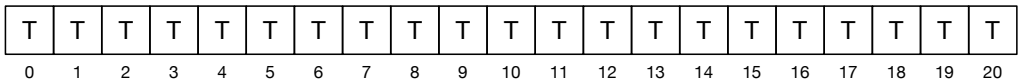
- Does the script work?
- Is it a good program?

Counting Primes: Script 2

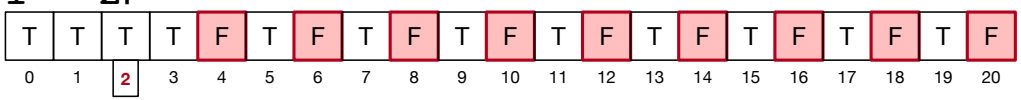
```
1  n = 20  # end of range of numbers to check for primes
2
3  still_is_prime = (n+1)*[True] # assume prime until disproven
4
5  for i in range(2, n):
6      if still_is_prime[i]:
7          # mark multiples of i as not prime
8          j = 2*i
9          while j <= n:
10             still_is_prime[j] = False
11             j += i
12
13 # now, every possible prime is a definite prime
14 count = sum([1 for v in still_is_prime[2:] if v])
15
16 print("# Prime numbers between 2 and " + str(n) + ":", count)
```

That for-loop

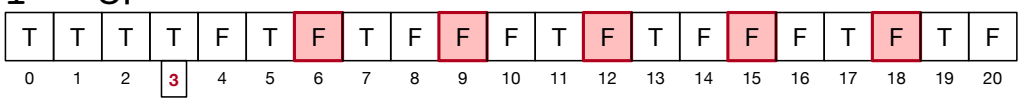
- Initially:



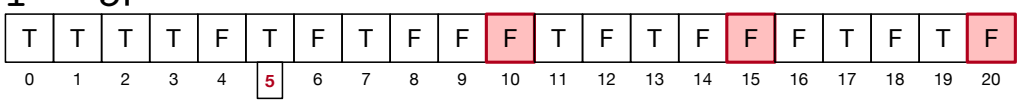
- `i = 2`:



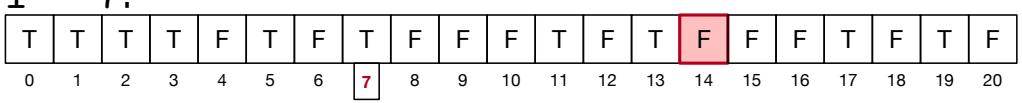
- `i = 3`:



- `i = 5`:



- `i = 7`:



Counting Primes: Script 2

- Does the script work?
- Is it a good program?
- Is it a better program than Script 1?

The Gambler's Ruin Problem

- Game outcomes:
  - 50% chance to win, 50% chance to lose
  - Winning earns 1 dollar
  - Losing removes 1 dollar
- Gambler's state:
  - Stake: initial amount of money
  - Goal: target amount of money
- Gambler plays games until:
  - Success: Stake increased to goal
  - Failure: Stake reduced to zero

The Gambler's Ruin Problem

- What is the probability of the gambler reaching their goal?
- What is the average number of bets placed before the gambler stops playing?

Example 1

Write a Python program to simulate the Gambler's Ruin:

- Complete a given number of $trials$.
- Each trial starts with a given $stake$ and a given $goal$.
- Each game has win_prob chance of winning.

After the simulation is complete, display the:

- Win percentage
- Average number of bets made per trial

Hints

Break up the problem into two parts:

- A function `gamble()` to simulate one trial.
- A simple loop to call `gamble()` multiple times

The Coupon Collector Problem

- A collector is trying to collect n unique coupons
  - Obtains one random coupon at a time.
  - All coupons are equally likely.
  - Repeats are allowed!
- Question: How many coupons will the collector own by the time they collect one of each kind of coupon?

Example 2

Write a Python program to:

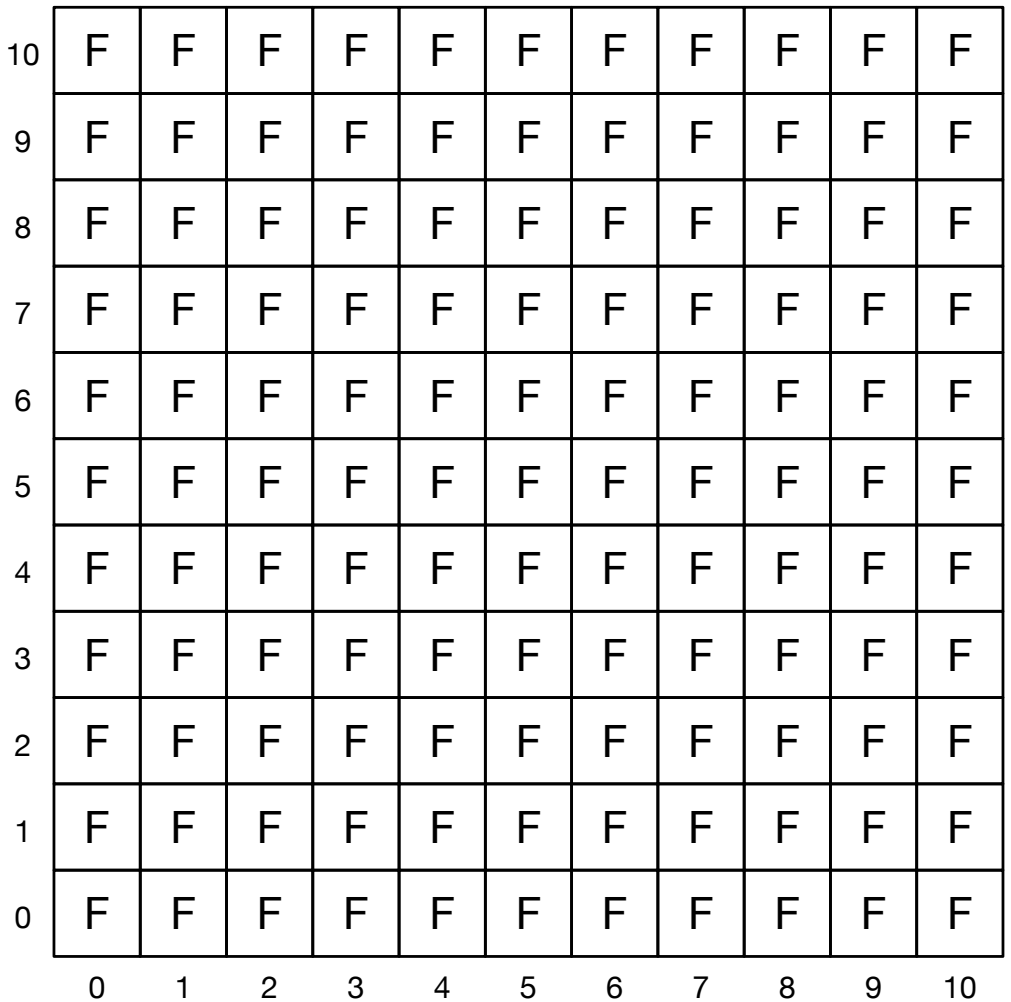
- Simulate the collection of coupons until all n coupons are obtained.
- Display the total number of coupons acquired.

Hint: Use a list of boolean values to keep track of which coupons have been collected already.

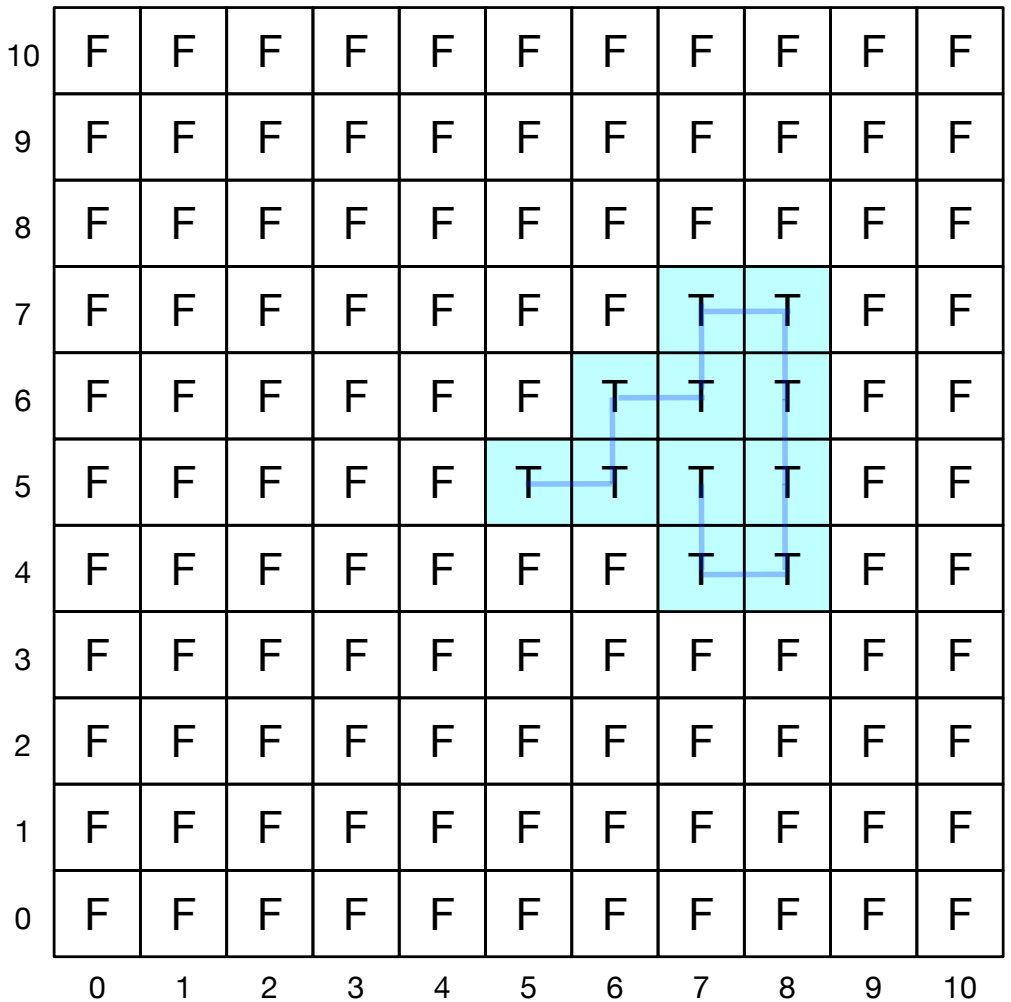
Self-Avoiding Random Walks

- A person in a room walks in random directions
  - Starts from middle of the room
  - Can step forward, backward, left, or right
  - Can't revisit locations — goes another direction
  - Walks until they reach a wall or dead end (no more legal steps)
- Question: What is the probability of hitting a dead end?

Initially


10 F F F F F F F F F F F
9 F F F F F F F F F F F
8 F F F F F F F F F F F
7 F F F F F F F F F F F
6 F F F F F F F F F F F
5 F F F F F F F F F F F
4 F F F F F F F F F F F
3 F F F F F F F F F F F
2 F F F F F F F F F F F
1 F F F F F F F F F F F
0 F F F F F F F F F F F
0 1 2 3 4 5 6 7 8 9 10

After a few random steps



Example 3

Write a Python program to simulate self-avoiding random walks:

- Complete given number of $trials$
- Each trial starts walker in the centre of an $n \times n$ room

After the simulation is complete, display the:

- Percentage of trials that end in dead ends

Hints

Break up the problem into two parts:

- A function `SARW()` to simulate one trial
- A simple loop to call `SARW()` multiple times