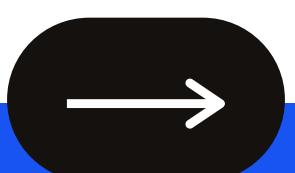


Python

Crack the Code !

An Introduction to python and Tech through
Personal Finance Tools





Sessions

Session 1

Situating Python and Basics

Session 2

Control Flow

Session 3

Loops, Lists

Session 4

Packages and Portfolios

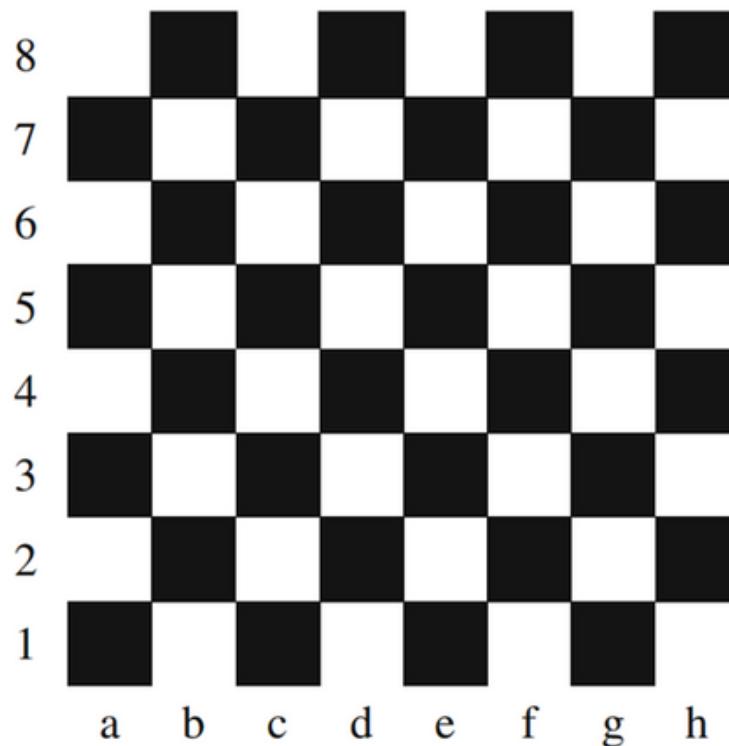
Exercise - Advanced

It is commonly said that one human year is equivalent to 7 dog years. However this simple conversion fails to recognize that dogs reach adulthood in approximately two years. As a result, some people believe that it is better to count each of the first two human years as 10.5 dog years, and then count each additional human year as 4 dog years.

Write a program that implements the conversion from human years to dog years described in the previous paragraph. Ensure that your program works correctly for conversions of less than two human years and for conversions of two or more human years. Your program should display an appropriate error message if the user enters a negative number.

Exercise - Advanced

Positions on a chess board are identified by a letter and a number. The letter identifies the column, while the number identifies the row, as shown below:

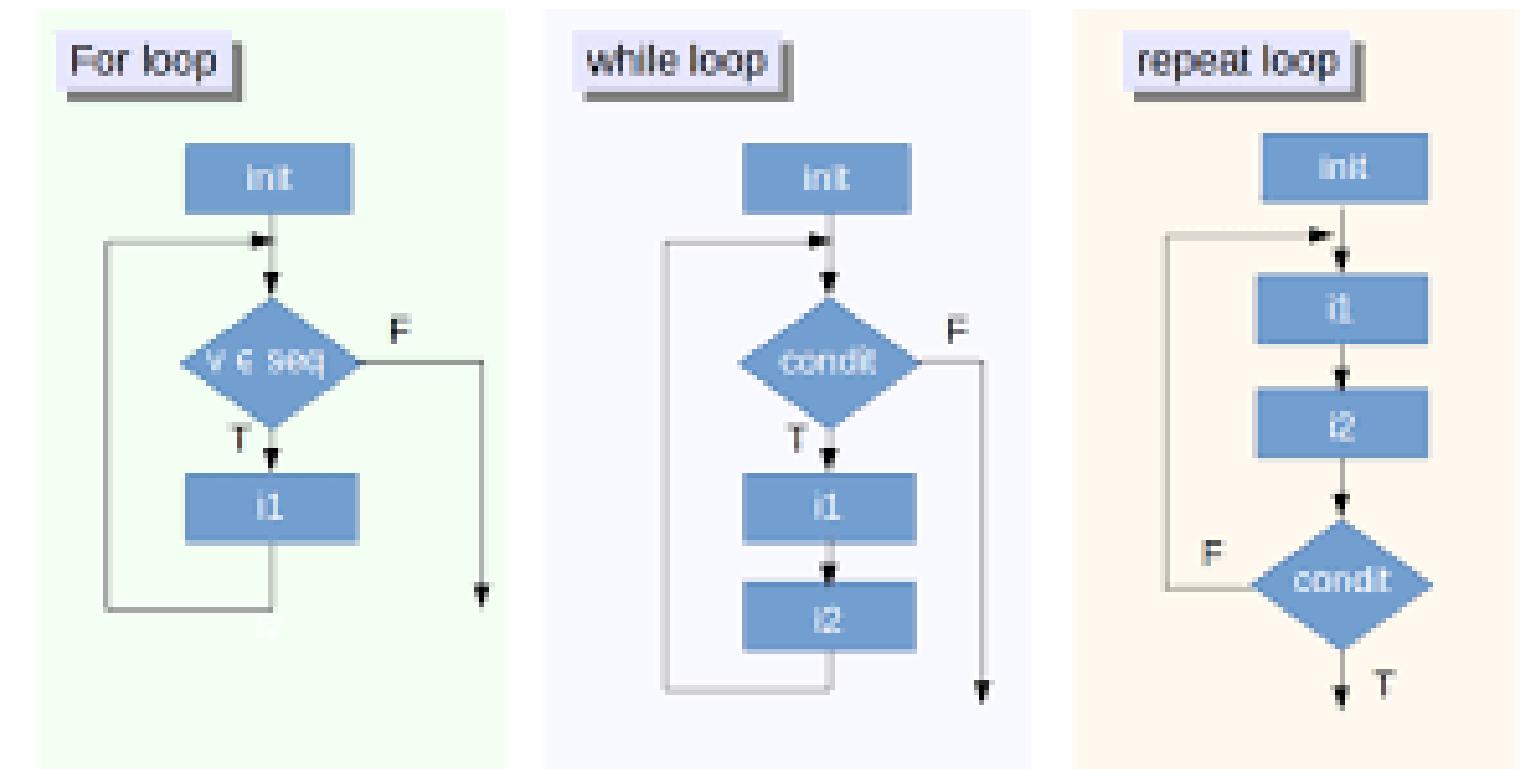


Write a program that reads a position from the user. Use an if statement to determine if the column begins with a black square or a white square. Then use modular arithmetic to report the color of the square in that row. For example, if the user enters a1 then your program should report that the square is black. If the user enters d5 then your program should report that the square is white. Your program may assume that a valid position will always be entered. It does not need to perform any error checking.

Basics of Code

- Loops

Every single one
Just repeating things - Do things
multiple times



Let's try!
Let's do examples!



Exercise - Basic

Write a program that displays a temperature conversion table for degrees Celsius and degrees Fahrenheit. The table should include rows for all temperatures between 0 and 100 degrees Celsius that are multiples of 10 degrees Celsius. Include appropriate headings on your columns. The formula for converting between degrees Celsius and degrees Fahrenheit can be found on the internet.

Exercise - Advanced

The value of π can be approximated by the following infinite series:

$$\pi \approx 3 + \frac{4}{2 \times 3 \times 4} - \frac{4}{4 \times 5 \times 6} + \frac{4}{6 \times 7 \times 8} - \frac{4}{8 \times 9 \times 10} + \frac{4}{10 \times 11 \times 12} - \dots$$

Write a program that displays 15 approximations of π . The first approximation should make use of only the first term from the infinite series. Each additional approximation displayed by your program should include one more term in the series, making it a better approximation of π than any of the approximations displayed previously.

Lists and Collections

A List is [] things pointing to other things

A Set is one without duplicates

*(Dictionaries Later)***

All of these have methods

Let's Try!



Exercise - Basic

In this exercise, you will create a program that reads words from the user until the user enters a blank line. After the user enters a blank line your program should display each word entered by the user exactly once. The words should be displayed in the same order that they were entered. For example, if the user enters:

```
first  
second  
first  
third  
second
```

then your program should display:

```
first  
second  
third
```

Exercise - Advanced

The Sieve of Eratosthenes is a technique that was developed more than 2,000 years ago to easily find all of the prime numbers between 2 and some limit, say 100. A description of the algorithm follows:

Write down all of the numbers from 0 to the limit

Cross out 0 and 1 because they are not prime

Set p equal to 2

While p is less than the limit **do**

Cross out all multiples of p (but not p itself)

Set p equal to the next number in the list that is not crossed out

Report all of the numbers that have not been crossed out as prime

The key to this algorithm is that it is relatively easy to cross out every n th number on a piece of paper. This is also an easy task for a computer—a for loop can simulate this behavior when a third parameter is provided to the `range` function. When a number is crossed out, we know that it is no longer prime, but it still occupies space on the piece of paper, and must still be considered when computing later prime numbers. As a result, you should **not** simulate crossing out a number by removing it from the list. Instead, you should simulate crossing out a number by replacing it with 0. Then, once the algorithm completes, all of the non-zero values in the list are prime.

Create a Python program that uses this algorithm to display all of the prime numbers between 2 and a limit entered by the user. If you implement the algorithm correctly you should be able to display all of the prime numbers less than 1,000,000 in a few seconds.

This algorithm for finding prime numbers is not Eratosthenes' only claim to fame. His other noteworthy accomplishments include calculating the circumference of the Earth and the tilt of the Earth's axis. He also served as the Chief Librarian at the Library of Alexandria.

Exercise - Basic

When writing out a list of items in English, one normally separates the items with commas. In addition, the word “and” is normally included before the last item, unless the list only contains one item. Consider the following four lists:

apples

apples and oranges

apples, oranges and bananas

apples, oranges, bananas and lemons

Write a function that takes a list of strings as its only parameter. Your function should return a string that contains all of the items in the list formatted in the manner described previously as its only result. While the examples shown previously only include lists containing four elements or less, your function should behave correctly for lists of any length. Include a main program that reads several items from the user, formats them by calling your function, and then displays the result returned by the function.

Exercise - Advanced

One of the first known examples of encryption was used by Julius Caesar. Caesar needed to provide written instructions to his generals, but he didn't want his enemies to learn his plans if the message slipped into their hands. As result, he developed what later became known as the Caesar Cipher.

The idea behind this cipher is simple (and as a result, it provides no protection against modern code breaking techniques). Each letter in the original message is shifted by 3 places. As a result, A becomes D, B becomes E, C becomes F, D becomes G, etc. The last three letters in the alphabet are wrapped around to the beginning: X becomes A, Y becomes B and Z becomes C. Non-letter characters are not modified by the cipher.

Write a program that implements a Caesar cipher. Allow the user to supply the message and the shift amount, and then display the shifted message. Ensure that your program encodes both uppercase and lowercase letters. Your program should also support negative shift values so that it can be used both to encode messages and decode messages.

Exercise - Advanced

A roulette wheel has 38 spaces on it. Of these spaces, 18 are black, 18 are red, and two are green. The green spaces are numbered 0 and 00. The red spaces are numbered 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30, 32, 34 and 36. The remaining integers between 1 and 36 are used to number the black spaces.

Many different bets can be placed in roulette. We will only consider the following subset of them in this exercise:

- Single number (1 to 36, 0, or 00)
- Red versus Black
- Odd versus Even (Note that 0 and 00 do **not** pay out for even)
- 1 to 18 versus 19 to 36

The spin resulted in 13...

Pay 13
Pay Black
Pay Odd
Pay 1 to 18

If the simulation results in 0 or 00 then your program should display Pay 0 or Pay 00 without any further output.

Exercise - Advanced

A roulette wheel has 38 spaces on it. Of these spaces, 18 are black, 18 are red, and two are green. The green spaces are numbered 0 and 00. The red spaces are numbered 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30, 32, 34 and 36. The remaining integers between 1 and 36 are used to number the black spaces.

Many different bets can be placed in roulette. We will only consider the following subset of them in this exercise:

- Single number (1 to 36, 0, or 00)
- Red versus Black
- Odd versus Even (Note that 0 and 00 do **not** pay out for even)
- 1 to 18 versus 19 to 36

The spin resulted in 13...

Pay 13
Pay Black
Pay Odd
Pay 1 to 18

If the simulation results in 0 or 00 then your program should display Pay 0 or Pay 00 without any further output.

Exercise - Super Advanced!

- We'll see this later:
 - Import Math
 - Math.Random()

BETS	PAYOUT	PROBABILITY
Outside Bets	1:1	
Even/Odd	1:1	46.37%
Black/Red	1:1	46.37%
1-18 (Low)	1:1	46.37%
19-36 (High)	1:1	46.37%
1-12 (Dozen)	2:1	31.58%
13-24 (Dozen)	2:1	31.58%
25-36 (Dozen)	2:1	31.58%
Inside Bets		
Single (1 number)	35:1	2.63%
Split (2 numbers)	17:1	5.26%
Street (3 numbers)	11:1	7.89%
Corner (4 numbers)	8:1	10.53%
Six Line (6 numbers)	5:1	15.70%
Combination of 1,2,3,0,00	6:1	13.16%

A roulette wheel has 38 spaces on it. Of these spaces, 18 are black, 18 are red, and two are green. The green spaces are numbered 0 and 00. The red spaces are numbered 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30 32, 34 and 36. The remaining integers between 1 and 36 are used to number the black spaces.

Many different bets can be placed in roulette. We will only consider the following subset of them in this exercise:

- Single number (1 to 36, 0, or 00)
- Red versus Black
- Odd versus Even (Note that 0 and 00 do **not** pay out for even)
- 1 to 18 versus 19 to 36

Write a program that simulates a spin of a roulette wheel by using Python's random number generator. Display the number that was selected and all of the bets that must be payed. For example, if 13 is selected then your program should display:

The spin resulted in 13...

Pay 13
Pay Black
Pay Odd
Pay 1 to 18

If the simulation results in 0 or 00 then your program should display Pay 0 or Pay 00 without any further output.

Functional Roulette!