W3.2 Applied

Leap Years

Write a Python program that calculates whether a given (CE) year is a leap year. The code should:

- input a positive integer `year` representing the year (assumed to be CE), and
- output `True` if that year was a leap year and `False` otherwise.

When collecting the input, please prompt the user with the following message:

"Please enter a year: "

Hint: All years that are divisible by four are leap years, unless they are also divisible by 100, in which case they are only leap years if they are also divisible by 400.

Flipping

Write a Python program that inputs a binary string `binary_string` (i.e., a string containing only 1s and 0s) and outputs a new string where every bit has been flipped.

For example, if the user inputs `00101111`, running the Python code should output `11010000`.

When collecting the input, please prompt the user with the following message:

"Please enter a binary string: "

Triangular Numbers

A triangular number is any number obtained by continued summation of the natural numbers $1, 2, 3, 4, 5, \ldots$, i.e., a number from the sequence $1, 3, 6, 10, 15, \ldots$

Write a Python program that inputs an integer `num` and outputs the smallest triangular number that is equal to or larger than the given number `num`.

When collecting the input, please prompt the user with the following message:

"Please enter a natural number: "

For example, the correct Python code outputs `28` for user input `num=25` as the triangular number 21 ($1 + 2 + 3 + 4 + 5 + 6 = 21$) is smaller than 25, and so the next triangular number 28 ($1 + 2 + 3 + 4 + 5 + 6 + 7 = 28$), which is bigger than 25, is the next triangular number in the series.

Return of Euclid

In the applied session of week 2, you have manually computed the greatest common divisor of two integers `num_1` and `num_2` with a little help from Python. Now, you will apply this week's knowledge to fully automate the task of implementing Euclid's algorithm for any valid user input `num_1` and `num_2`. You can assume `num_1` is greater than or equal to `num_2`.

When collecting the input, please prompt the user with the following messages:

"Please enter the value of number 1: "

"Please enter the value of number 2: "

Additional Question: Does your code work when `num_1<num_2`? If it works, what happens to the values of `num_1` and `num_2` in the first iteration of the algorithm?

Predict the outcome of the following computations by hand

In this activity, please attempt to predict the outcome of the following code *without using the Python interpreter*.

Question 1 *Submitted Mar 15th 2022 at 11:49:58 am*

```
(2+7)/(5%3)
```

- ○ 4
- ◉ 4.5
- ○ 9
- ○ 9.0

Question 2 *Submitted Mar 15th 2022 at 11:50:13 am*

```
not True or False and True
```

- ○ True
- ◉ False

Question 3 *Submitted Mar 15th 2022 at 11:50:33 am*

```
response = 'unknown'
x = 15
if x%2 == 0:
    response = 'is even'
elif x%3 == 0 or x%5 == 0 or x%7 == 0:
    response = 'is composite'
elif x < 13:
    response = 'is prime'
else:
    response = 'is too big'
print(response)
```

- is even
- is composite
- is prime
- is too big

Question 4 *Submitted Mar 15th 2022 at 11:50:53 am*

```
i = 0
word = 'anyway'
letter = 'e'
while i < len(word):
    if word[i] == letter:
        print(True)
    i += 1
print(False)
```

- True
- False
- True
- False

Question 5 *Submitted Mar 15th 2022 at 11:51:01 am*

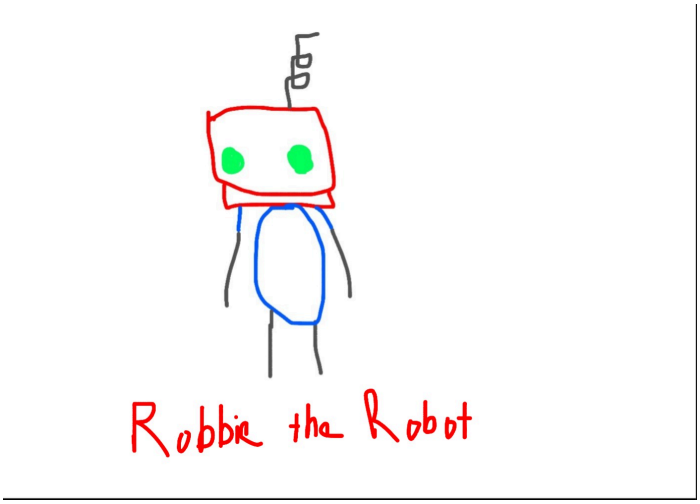
```
i = 0
word = 'anyway'
letter = 'a'
count = 0
while i < len(word):
    if word[i] == letter:
        count += 1
    i += 1
print(count)
```

- [ ] 0
- [ ] 1
- [x] 2
- [ ] 3

How long does Robbie take to get there?

In the last week's applied, you have successfully written some code in Python to predict the final position x_{final} of Robbie the Robot (i.e., visualized below), given its initial location $x_{initial}$, velocity v, acceleration a and movement duration t based on the following Physics-based formula:

$$x_{final} = x_{initial} + vt + \frac{1}{2}at^2$$



This week, you are going to (again) help Robbie using Python! Specifically, you need to write an algorithm to help Robbie compute its movement duration t, given its final location x_{final}, initial location $x_{initial}$, velocity v and acceleration a. But how can you solve this problem?

Turns out, solving this problem in general corresponds to solving the problem of *finding the real roots of a quadratic equation*! Let's move onto the next section to start solving this problem.

Part 1: Write an algorithm

Write an algorithm (in pseudocode) in a file named `pseudo_real_roots.txt` for finding the real roots of a quadratic equation

$$AX^2 + BX + C = 0$$

for arbitrary real coefficients A, B, and C. Things to note:

1. The quadratic formula is

$$X = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

2. Begin by considering the inputs and outputs of this algorithm.
3. There exist 0, 1, or 2 real roots in any quadratic equation, depending on what is yielded by

$$B^2 - 4AC$$

4. The number of real roots affects how the quadratic formula should be used.
5. Determining how many real roots you are trying to find should occur somewhere in the algorithm.

We now need to figure out what are the values of X, A, B, and C based on the previously presented Physics formula. So let's do that next!

Part 2: What are X, A, B and C?

Given the quadratic equation

$$AX^2 + BX + C = 0$$

and the Physics equation

$$x_{final} = x_{initial} + vt + \frac{1}{2}at^2$$

can you figure out the values of X, A, B, and C in terms of Robbie's movement duration t, final location x_{final}, initial location $x_{initial}$, velocity v and acceleration a?

Question 1 *Submitted Mar 15th 2022 at 11:51:36 am*

The value of X is:

- x_{final}
- $x_{initial}$
- **(selected)** t
- a

Question 2 *Submitted Mar 15th 2022 at 11:51:59 am*

The value of A is:

- a
- **(selected)** $0.5a$
- v
- $x_{initial}$

Question 3 *Submitted Mar 15th 2022 at 11:52:14 am*

The value of B is:

- $x_{initial}$
- x_{final}
- v
- a

Question 4 *Submitted Mar 15th 2022 at 11:52:26 am*

The value of C is:

- 0
- $x_{initial}$
- x_{final}
- $x_{final} - x_{initial}$
- $x_{initial} - x_{final}$

Part 3: Implement the algorithm in Python

Now you are ready to help Robbie calculate its movement duration. Specifically, you will build on your previous work from Part 1 and Part 2, and calculate Robbie's movement duration t, given its final location x_{final}, initial location $x_{initial}$, velocity v and acceleration a based on:

- your pseudocode from Part 1, and
- your definitions of X, A, B and C in terms of t, x_{final}, $x_{initial}$, v and a from Part 2.

Implement your algorithm in Python in a file named `real_roots.py` which

1. computes the values of A, B and C given the user inputs x_{final}, $x_{initial}$, v and a, and
2. computes and prints all the roots of a quadratic equation given its three coefficients A, B and C.

When collecting the input, please prompt the user with the following messages:

"Please enter the final location: "

"Please enter the initial location: "

"Please enter the velocity: "

"Please enter the acceleration: "

If there are two real roots `D` and `E`, the code should display the following message:

"The real roots of the quadratic function are: D and E"

Similarly, if there is a single real root `D`, the code should display the following message:

"The real root of the quadratic function is: D"

Finally, if there are no roots, the code should display the following message:

"No real roots for the quadratic function!"

Feedback

Question 1 *Submitted Mar 21st 2022 at 8:53:50 pm*

Feedback

What worked best in this lesson?

-

Question 2 *Submitted Mar 21st 2022 at 8:53:52 pm*

Feedback

What needs improvement most?

-