1. Write a program called **sums()** that asks the user for a positive integer and prints out all the ways that the integer can be made by adding together two *smaller* positive integers (excluding 0). Hint: keep track of two variables called `first` and `second`, as you go through the loop. Your output should look similar to this:

```
>>> sums()
Enter a positive number: 6
1 plus 5 equals 6
2 plus 4 equals 6
3 plus 3 equals 6
4 plus 2 equals 6
5 plus 1 equals 6
```

2. In class, we wrote the program oddRecips() to add up the terms of the series 1/1 + 1/3 + 1/5 + 1/7 + 1/9 + … , which does *not* converge as more and more terms are added. Modify this program to add up the *even* reciprocals, starting from 1/2. That is: 1/2 + 1/4 + 1/6 + 1/8 + 1/10 + … Call the new version of your program **evenRecips()**. How much did you have to change the original code? Does this series converge as you add together more terms?

```
>>> evenRecips()
How many terms to add up? 5
Total is 1.1416666666666666
```

3. Write a program called **doubleRecips()** that adds up the terms of the series 1/1 + 1/2 + 1/4 + 1/8 + 1/16 + … where the denominator of each new term doubles. Unlike the series of odd reciprocals, this series converges to a specific value as more and more terms are added together. Use your program to determine which value.

4. The following series, described by Archimedes, also converges: 1/1 + 1/4 + 1/16 + 1/64 + 1/256 + … where each denominator goes up by a factor of 4. What number does this series converge to? Write a program called **archimedes()** to find out. As usual, your program should ask the user for the number of terms to add up.

5. The great Swiss mathematician Leonhard Euler proved that the infinite sum of the reciprocals of the squares, namely: 1/1 + 1/4 + 1/9 + 1/16 + 1/25 + 1/36 + … is equal to the value $\pi^2/6$. Write a program called **squareRecips()** to verify this. For comparison, your program should also compute and print the value of $\pi^2/6$ using the constant `math.pi` available in the `math` library (don't forget to put `import math` at the top of your code).

6. Consider this infinite sum of fractions: 1/2 + 2/3 + 3/4 + 4/5 + 5/6 + 6/7 + 7/8 + 8/9 + 9/10 + …
Write a program called **fractions()** that asks the user for the number of terms to add up, and then prints the sum. As more and more terms are added together, does this series converge to a particular value, or not? Hint: use a variable `numer` to keep track of the current numerator value, and a variable `denom` to keep track of the denominator value, as you go through the loop.

7. Write a program called **power()** that asks the user for a base value and an integer exponent and then computes the value of the base raised to the specified power. For example, if the base is 2 and the exponent is 3, your program should compute 2 × 2 × 2. If the base is 5 and the exponent is 4, it should compute 5 × 5 × 5 × 5. You are not allowed to use the `**` operator or the `math.pow` function for this problem. Instead, use a for-loop to perform the necessary multiplications, and a variable called `total` to accumulate the product. Examples:

```
>>> power()
Enter a base value: 2
Enter the power: 3
2 raised to the power 3 is 8

>>> power()
Enter a base value: 5
Enter the power: 4
5 raised to the power 4 is 625
```

8. Write a program called **pi()** that approximates the value of π by summing the terms of the following series: 4/1 – 4/3 + 4/5 – 4/7 + 4/9 – 4/11 + . . . Your program should ask the user for the number of terms to add up. Keep track of the numerator and denominator in the variables `numer` and `denom`. After printing the final total at the end, also print out the value of the `math` library constant `math.pi` for comparison. Examples:

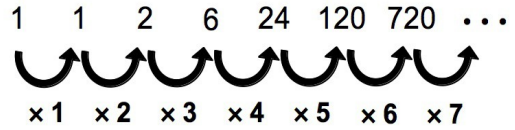
```
>>> pi()
How many terms to add up? 1
pi is approximately 4.0
The true value is 3.141592653589793

>>> pi()
How many terms to add up? 3
pi is approximately 3.466666666666667
The true value is 3.141592653589793
```

9. The following series is very similar to the double-reciprocals series from Exercise 3, except that it starts with 1/2 instead of 1/1, and the signs alternate for subsequent terms: 1/2 – 1/4 + 1/8 – 1/16 + 1/32 – 1/64 + ... Write a program called **alternating()** to find out which value this series converges to.

10. The value of the mathematical constant *e* can be calculated by summing the infinite series below:

e = 1/1 + 1/1 + 1/2 + 1/6 + 1/24 + 1/120 + 1/720 + …

Here each denominator goes up by a multiplicative factor that increases by one each time, starting from 1:



Write a program called **e()** that asks the user for the number of terms to add up, and computes an approximation to *e* using the above series. Also print out the value of the `math` library constant `math.e` for comparison.

11. The Fibonacci sequence is an infinite sequence of numbers where each successive number is the sum of the previous two: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... Write a program called **fib()** that computes the *n*th Fibonacci number, where *n* is the position in the sequence counting from 1. For example, if $n = 6$, then the result is 8, because the 6th number in the sequence is 8. Hint: use two separate variables called `a` and `b` to keep track of the two most recent values in the sequence, starting with 1 and 1. Use a for-loop to move forward through the sequence, incrementing the variables appropriately, until you reach the position you are interested in. Make sure your program gives correct answers for the first two positions as well ($n = 1$ and $n = 2$).

```
>>> fib()
Which position? 6
The Fibonacci number at that position is 8

>>> fib()
Which position? 10
The Fibonacci number at that position is 55

>>> fib()
Which position? 1
The Fibonacci number at that position is 1

>>> fib()
Which position? 2
The Fibonacci number at that position is 1
```