Algorithms - a simple introduction in Python: Part Nine

The universe cannot be read until we have learned the language and become familiar with the characters in which it is written. It is written in mathematical language, and the letters are triangles, circles and other geometrical figures, without which means it is humanly impossible to comprehend a single word. Without these, one is wandering about in a dark labyrinth.

Galileo

Pascal’s Triangle

Most of you will be familiar with Pascal’s Triangle. This is a simply-described mathematical object with some interesting properties. For instance, when you raise an expression of the form (x + y) to an integer power, the triangle will tell you the coefficients of the expanded equation.

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

So, $(x+y)^2 = x^2 + 2xy + y^2$, and row 3 of the triangle (the top one is counted as row zero), holds the numbers 1, 2 and 1, which correspond to our equation - we could write it as:

$$\mathbf{1}x^2 + \mathbf{2}xy + \mathbf{1}y^2$$

Similarly, $(x+y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$.

There are many other interesting features of this sequence, for instance the third downwards diagonal contains the triangular numbers: 1, 3, 6, 10 etc.

Anyway, let’s have a look at how to calculate the number at any position in Pascal’s Triangle.

We are going to label each position in the triangle with a number for row and column. The 0th row has one element, which we label (0,0). The next row has two, (1, 0) and (1, 1). Each number in the triangle is the sum of the two numbers above it. You’ll see immediately that any number in column “0“ must equal 1 and any number which has the same value for row and column (eg. (3, 3)) must also be a 1.

Here’s some code.

```python
# pascal.py Pascal's Triangle

def main():
    print("*** Pascal's Triangle ***")
    a, b = input("Enter row and column, separated by a comma.\n? ").split(',')
    print(pascal(int(a), int (b)))

def pascal(r, c):
    """Takes row and column and returns the number at that position."""
    if c == 0 or r == c:
        return 1
    else:
        return pascal(r - 1, c - 1) + pascal(r - 1, c)
```

```
if __name__ == "__main__":
    main()
```

As you can see, the function uses a recursive process to find the values for any position in the triangle.

Now that we have a way to find each member of the triangle, we can adapt our code to print it out (although this version doesn't attempt to make the spacing look right!)

Here is a print_triangle() function, along with an alternative main().

```
def main():
    print("*** Pascal's Triangle ***")
    print_triangle(int(input("How many rows of the triangle: \n? ")))

def print_triangle(x):
    """Prints out x+1 rows of Pascal’s Triangle."""
    for r in range(x + 1):
        for c in range(r+1):
            print(pascal(r, c), end=" ")
        print()
```

Sample Output:

```
*** Pascal’s Triangle ***
How many rows of the triangle:
? 10
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
```

The nested loops in the print_triangle() function take care of generating row and column labels and our pascal() function happily prints them out!