Functions
Defining Functions With `def`

The `def` statement ends in a colon (":")

```python
def printMyName(name):
    print("***************")
    print(name)
    print("***************")
```

All statements that are part of the function definition must be indented. This is how the Python interpreter knows that these statements are part of the definition of the function (and not part of some other part of your program).
Placement of Functions in Your File

# define all of your functions
def <function>():
    def <function>():
        def <function>():
            def <function>():
                def <function>():

# define your main program
def main():
    def main():

# call your main program to start execution
main()
Exercise:

A triangular number is an integer that corresponds to the number of objects arranged in an equilateral triangle, as shown in the picture.

The $n$th triangular number $T_n$ is the number of dots in the triangular arrangement with $n$ dots on a side, and is equal to the sum of the $n$ natural numbers from 1 to $n$.

Write a function `triangle(n)` that accepts as an argument an `int`, and returns the $n$th triangular number.

Then write a main program that makes a table showing the first 20 triangular numbers.
Types of Arguments/Parameters: Positional

Using *positional* arguments means that the arguments must be passed in the same order as their respective parameters in the function definition.

Example:

```python
def extremes(x, y, z):
    biggest = max(x, y, z)
    smallest = min(x, y, z)
    return biggest, smallest
```

**function call:**

```python
largest, smallest = extremes(first, second, third)
```

*x maps to* first, *y maps to* second, *and z maps to* third because that’s the order they appear in both the function definition and the function call.
Using *keyword* arguments means that each argument in the function call is written in the form `parameter = value`, where `parameter` is the parameter that you want to map the value to. This enables the arguments to appear in any order since each argument is labeled.

Example:

```python
def extremes(x, y, z):
    ...
```

Function call:

```python
largest, smallest = extremes(z=third, x=first, y=second)
```

*x maps to first, y maps to second, and z maps to third* because we explicitly make those associations using keyword parameters.
Types of Arguments/Parameters: Default Arguments

You can make an argument optional by providing a default value for the function to use if the user doesn’t provide a value in the function call.

Example:

```python
def printArea(width=1, height=2):
    area = width * height
    print("width:", width, "\theight:", height,
          "\tarea:", area)
```

function calls:
```
printArea()                  # width=1, height=2
printArea(4, 2.5)            # width=4, height=2.5
printArea(height=4, width=3) # width=3, height=4
printArea(width=7)           # width=7, height=2
printArea(height=6)          # width=1, height=6
```