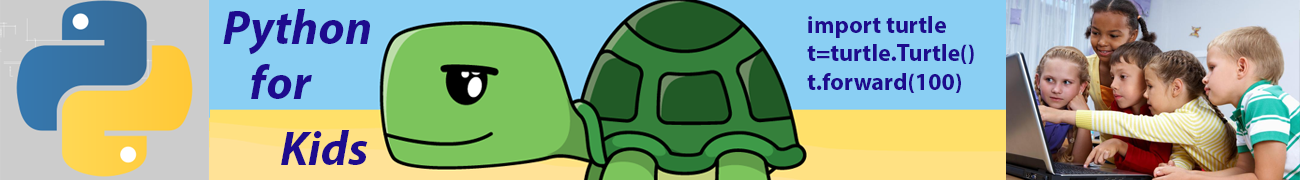
****

**Lesson 7: Numbers and Variable in Python**

**Commands Print and Input**

Numbers and Math Operators in Python, Simple Calculations

There are two primary numeric types that we’ll be using in Python: integers and floats. First type are integers: positive (for example, 3,7,9,783,23 etc.) or negatives as -4, -12, -342, -781 etc. Integers, or whole numbers, are useful for counting and for basic math (2 + 2 = 4). We usually state our age in whole numbers, so when you say you’re 5 or 16 or 42, you’re using an integer. When you count to 10, you’re using integers.

Floating numbers, or simply floats, are numbers that can have whole and fractional parts and are written using decimal points, for example, 3.47, 62.34, 761.23 etc.

Math Operators

The math symbols like + (plus) and - (minus) are called *operators* because they operate, or perform calculations, on the numbers in our programs. When we say “4 + 2” aloud or enter it on our calculator, we want to perform addition on the numbers 4 and 2 to get their sum, 6. Python uses most of the same operators that you would use in a math class, including +, -, and parentheses, (), as shown in Table below. However, some operators are different from what you may have used in school, like the multiplication operator (the asterisk, \*, instead of ×) and the division operator (the forward slash, /, instead of ÷). We’ll get to know these

operators better in future.

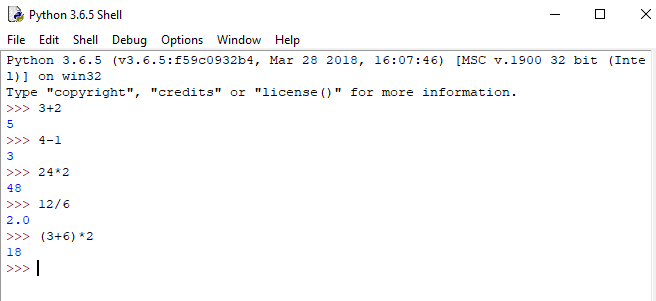
**Summary for math operations:**

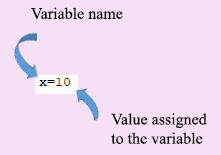
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code Instruction** | | **What it does** | | |
| Math symbol | Python operator | Operation | Example | Result |
| + | + | Addition | 11+12 | 23 |
|  |  |  |  |  |
| **-** | **-** | Subtraction | 25-10 | 15 |
|  |  |  |  |  |
| x | \* | Multiplication | 12\*3 | 36 |
|  |  |  |  |  |
| ÷ | / | Multiplication | 12\*3 | 36 |
|  |  |  |  |  |
| 62 | \*\* | Exponent of power | 6\*\*2 | 36 |
|  |  |  |  |  |
| () | () | Parentheses(grouping) | (8+3)\*6 | 66 |
|  |  |  |  |  |
| % | % | Modulus of operation\* | 14%5  12%5 | 4  2 |

\*In computing, the **modulo operation** finds the remainder after division of one number by another (called the **modulus** of the **operation**).

Math in the Python Shell.

Let’s use the Python shell this time. The Python shell gives you direct access to Python’s power without writing a whole program. It’s sometimes called the *command line* because you can type commands line by line and instantly see the result. For example, you can type a math problem (called an *expression* in programming) like 4 + 2 directly at the command prompt (the >>> symbol with the flashing cursor) in the Python shell, and when you press ENTER, you’ll see the *result* of the expression, or the answer to the math problem. Try typing some of the examples listed in Table below and see what Python says. Feel free to try your own math problems as well.



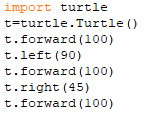
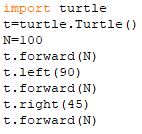
A ***variable***is something you want the computer to remember while your program is running. When Python “remembers” something, it’s storing that information in the computer’s memory. Variable is like box where data can be stored and labelled. Python can remember *values* of several types, including **number values** (like 7, 42, or even 98.6) and **strings** (letters, symbols, words, sentences). In Python, as in most modern programming languages, we assigna value to a variable with the equal sign (=). An assignment like x = 10 tells the computer to remember the number 10 and give it back to us anytime we call out x. So, the assignment operator = assigns the value on the right to a variable on the left. 

The command to assign a variable in Python does the same job as this block-based Scratch

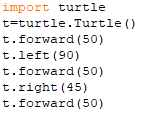
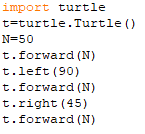


We can use any letter or word to use as a variable. For example, v=23 assigns the value of integer 23 to the variable v. In the case score=5 we assign the integer value 5 to the variable score.

**What is the reason to use variable instead of number? Let’s show this with an example. Below two simple codes: one from left side created with number and from right side code uses variable X.**

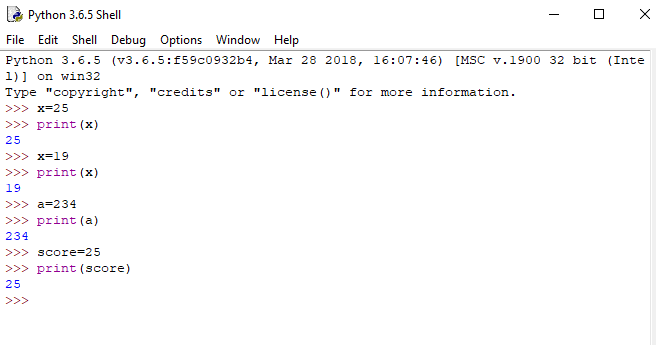
Both of them gave the same result. However, if you want to change the number 100 by the other value from left side you must replace three lines, from right side replace only one line. For example, let’s use number 50 instead of 100. In this case codes from left and right sides are shown below

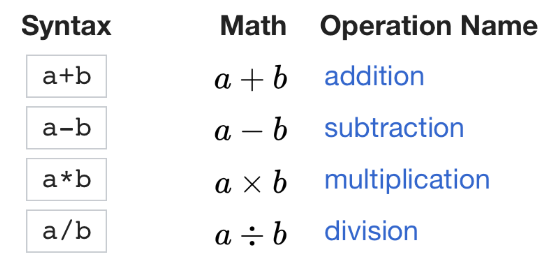
Here N is a variable.

Printing Variable

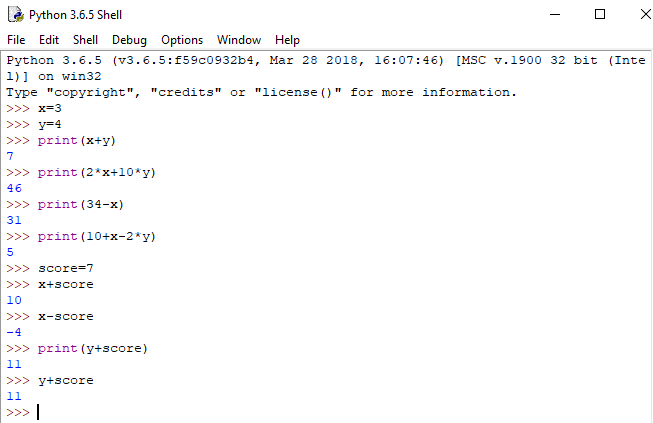
The ‘print’ command is used to show something on the screen. It has nothing to do with the printer. You can use it to show the value of a variable



We can use math operations with variables.



Only in the program code we have to assign number to each variable that use in the program. Examples are shown below

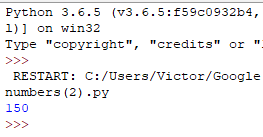


When you assign some value to the variable, computer remember this variable value in the memory and when you use variable with certain name for math operation computer takes the value assigned to the variable and process math operation.

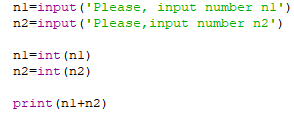
Command *input*

Very convenient and important command, can be used when math calculations.

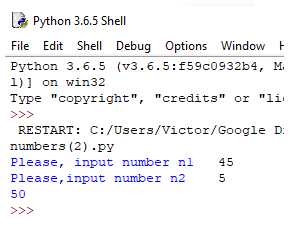
Let’s create code that adds two positive integer numbers n1 and n2. Code is shown below from left side, result—from right side.

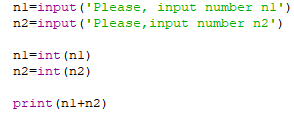
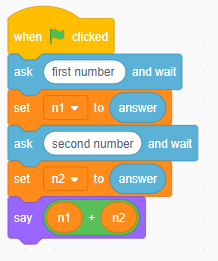
Now, we want to change our numbers n1 and n2. What to do? We have to change our code. However, introduce command input, that allows us to replace our numbers using keyboard control. Code is shown below.



Result is presented on the Shell Screen

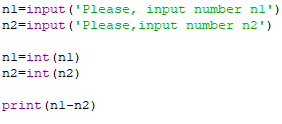


First code line prompts the user to input first number n1 and waits while you type the number. After you type first number, second code line prompts you to input second number and waits till you type it. Line 3 and 4 convert first number and second numbers into digital type (numbers typed according to input request are string variable, and we will discuss such kind of variables a little bit later). Last line print results on the Shell screen. What is the reason to use input data option? Program became flexible and we can repeat it a lot of times without code changes. This input option is similar to the “ask and wait” block in Scratch program. So, input() is opposite of the print(). It lets the user give instructions or data to the program by typing them in. Code from left side is written with Python, code from right side –with Scratch

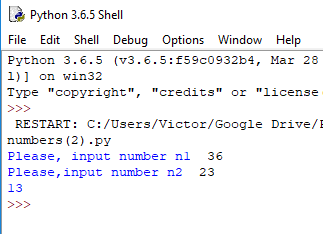
 

Now a few simple programs that demonstrate “input()”🡪”print” application.

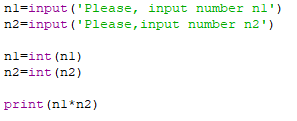
1. **Example #1**(subtraction of two numbers)



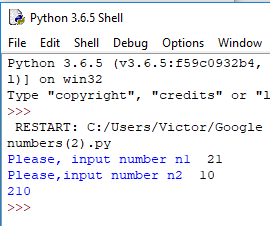
**Result**



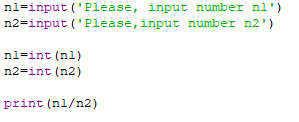
1. **Example #2** ( Product of two numbers)



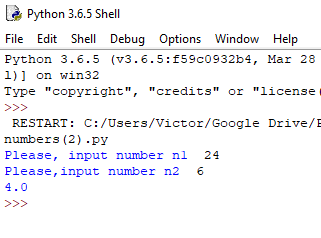
**Result**



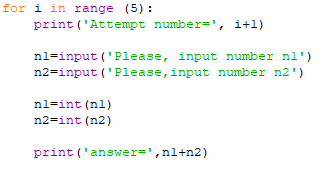
1. **Example #3 (**Division of two numbers)



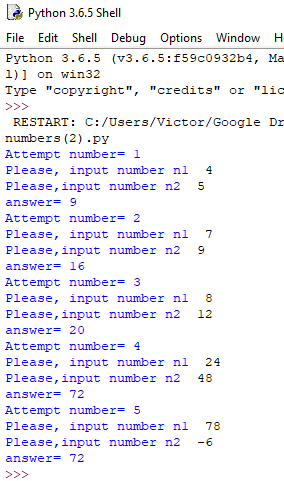
**Result**



1. **Example #4** (A few attempts to add two numbers)



**Result** (5 times)

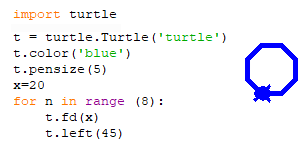


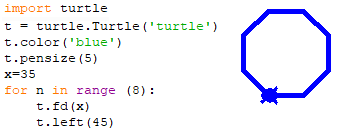
**Attention for students!**

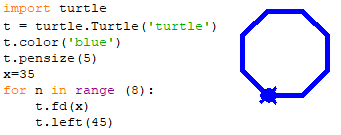
If you consider that the material, presented above is too complicated, please skip it.

Let’s write a few programs that use variables and math operations with our lovely turtle library.

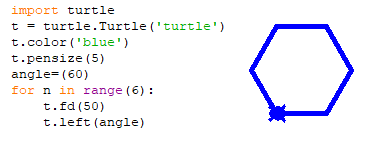
1. **Example #1** (Octagon, length is equal 20, side length of an octagon is a variable x=20)

****

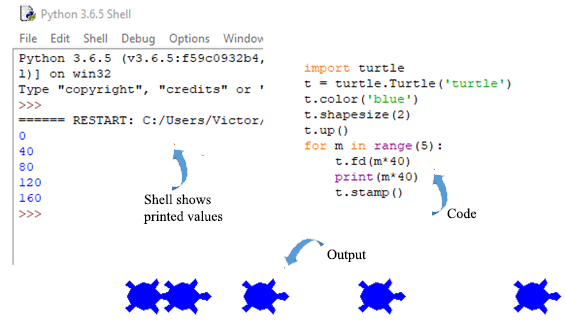
****

****

1. **Example #2** (Hexagon, side length is equal 50, turn angle is variable angle=60)

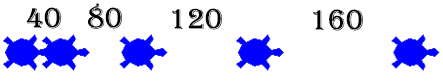


1. **Example #3** (Changing distance between turtles)

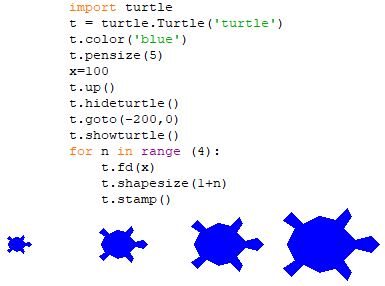


The ‘print’ command is used to show something on the Shell screen. It has nothing to do with the printer. You can use it to show the value of a variable. According to our code *m* variable takes value equal 0(first step), *m* variable takes value equal 1(second step), *m* variable takes value equal 2(third step), *m* variable takes value equal 3(forth step), and *m* variable takes value equal 4(fifth step).

According to our code ddistance between turtles is shown below

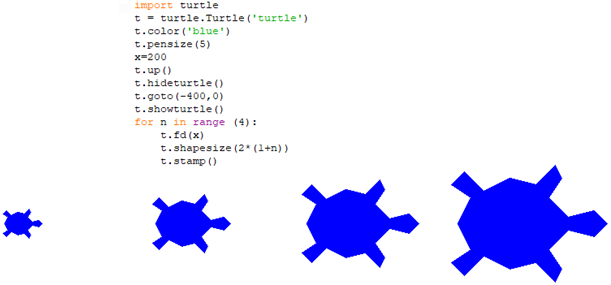


1. **Example #4** (Changing of the turtle shape size)
2. (Distance between turtles is x=100, size of the turtle changes in the loop; smallest one is equal 1, largest one is 4)

****

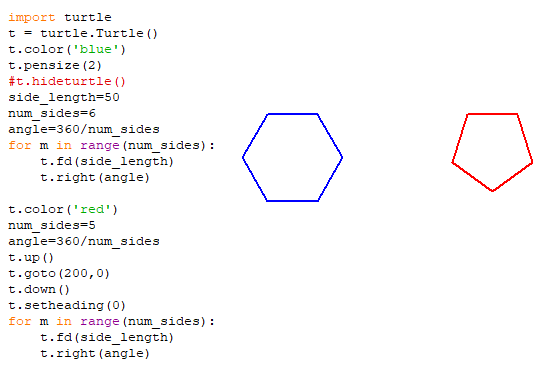
In this example shape size of the turtle changes according to the following law: (1+n), where n is a variable. Let’s explain. Firstly turtle has a shape size equal (1+0)=1 because the first number *n* of the loop n=0. During the second loop cycle turtle has a shape size equal (1+1)=2 because the second number of the loop n=1; during the third loop cycle turtle has a shape size equal (1+2)=3 because the third number of the loop n=2 and last cycle step: turtle has a shape size equal (1+3)=4 because the first number of the loop n=3

1. (Distance between turtles is x=200, size of the turtle changes in the loop; smallest one is equal 2, largest one is 6)



In this example shape size of the turtle changes (not distance between turtles) according to the following law: 2\*(1+n). Let’s explain. First turtle has a shape size equal 2\*(1+0)=2 because the first number of the loop n=0. Second turtle has a shape size equal 2\*(1+1)=4 because the second number of the loop n=1; third turtle has a shape size equal 2\*(1+2)=6 because the third number of the loop n=2 and fourth turtle has a shape size equal 2\*(1+3)=8 because the fourth number of the loop n=3.

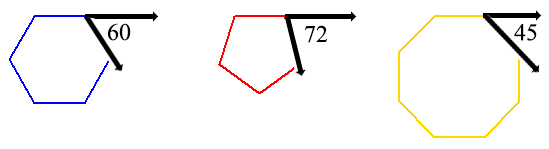
1. **Example #5** (Number of polygons)



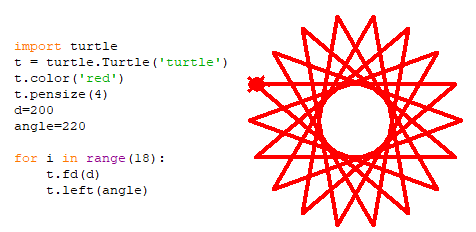
Let’s explain this code. Code has three variables: the name of the first is side\_length; second one is num\_sides and third is angle, which is equal 360/num\_sides. If num\_sides=5 we draw pentagon, because for pentagon angle has to be 360/5=72. If num\_sides=6 we draw hexagon, because for hexagon angle has to be 360/6=60. If num\_sides=8 we draw octagon, because for octagon angle has to be 360/8=45.

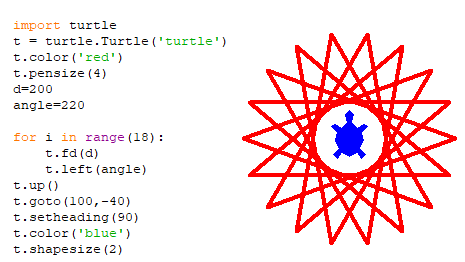
The number of loop cycles is equal of the number of sides. 

If this number is less than the number of sides geometry shape is not completely drawn, it could be more than the number of sides, however it is useless.



1. **Example #6** (Number of Stars)





Challenges:

1. Modify code of an example #7 to get Square, Triangle and Octagon
2. Build a code for the following outputs

