

Lesson 7: Numbers and Variables in Python

Numbers and Math in Python, Simple Calculations

The two primary types of numbers in Python are called *integers* (whole numbers, including positive and negatives, like 7, -9, or 0) and *floating-point numbers* (numbers with decimals, like 1.0, 2.5, 0.9). 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.

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 \times) and the division operator (the forward slash, /, instead of \div). We'll get to know these operators better in future.

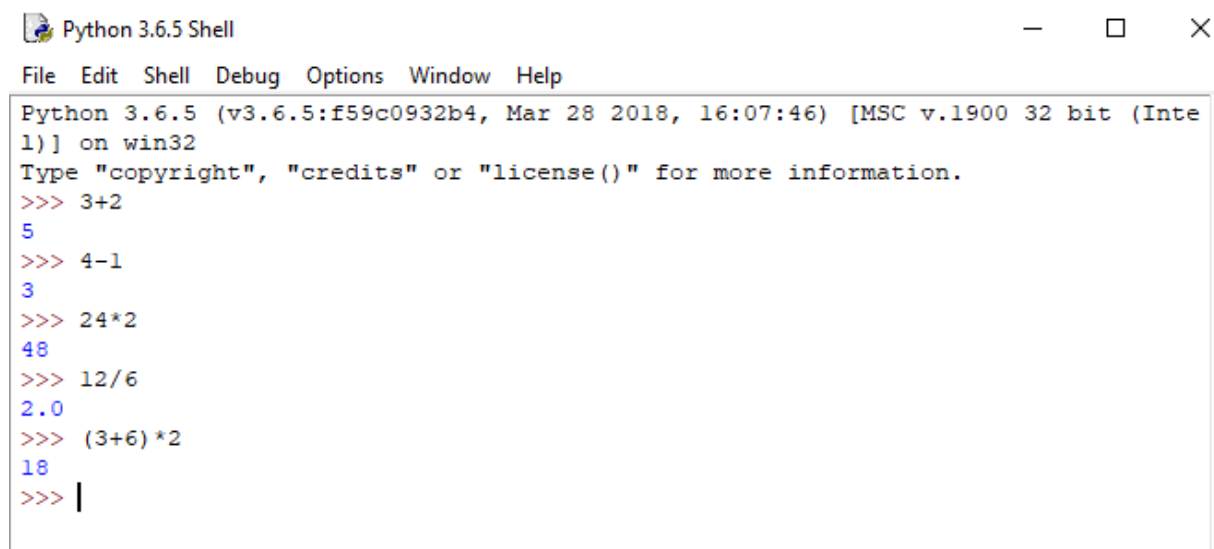
Summary for math operations:

Code Instruction		What it does		
Math symbol	Python operator	Operation	Example	Result
+	+	Addition	11+12	23
Math symbol	Python operator	Operation	Example	Result
-	-	Subtraction	25-10	15
Math symbol	Python operator	Operation	Example	Result
\times	*	Multiplication	12*3	36

Math symbol	Python operator	Operation	Example	Result
÷	/	Multiplication	12*3	36
Math symbol	Python operator	Operation	Example	Result
6 ²	**	Exponent of power	6**2	36
Math symbol	Python operator	Operation	Example	Result
()	()	Parentheses(grouping)	(8+3)*6	66

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.



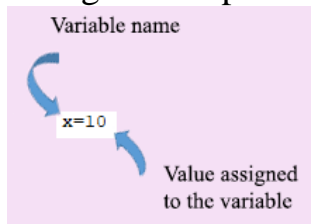
```

Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 3+2
5
>>> 4-1
3
>>> 24*2
48
>>> 12/6
2.0
>>> (3+6)*2
18
>>> |

```

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 assign a 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.

```
import turtle          import turtle
t=turtle.Turtle()     t=turtle.Turtle()
t.forward(100)        N=100
t.left(90)            t.forward(N)
t.forward(100)        t.left(90)
t.right(45)           t.forward(N)
t.forward(100)        t.right(45)
                     t.forward(N)
```

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

```
import turtle          import turtle
t=turtle.Turtle()     t=turtle.Turtle()
t.forward(50)         N=50
t.left(90)            t.forward(N)
t.forward(50)         t.left(90)
t.right(45)           t.forward(N)
t.forward(50)         t.right(45)
                     t.forward(N)
```

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

```

Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x=25
>>> print(x)
25
>>> x=19
>>> print(x)
19
>>> a=234
>>> print(a)
234
>>> score=25
>>> print(score)
25
>>>

```

We can use math operations with variables.

Syntax	Math	Operation Name
<code>a+b</code>	$a + b$	addition
<code>a-b</code>	$a - b$	subtraction
<code>a*b</code>	$a \times b$	multiplication
<code>a/b</code>	$a \div b$	division

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

```

Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x=3
>>> y=4
>>> print(x+y)
7
>>> print(2*x+10*y)
46
>>> print(34-x)
31
>>> print(10+x-2*y)
5
>>> score=7
>>> x+score
10
>>> x-score
-4
>>> print(y+score)
11
>>> y+score
11
>>> |

```

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.

```
n1=100
n2=50
print(n1+n2)
```

```
Python 3.6.5 (v3.6.5:f59c0932b4,
1)] on win32
Type "copyright", "credits" or ".
>>>
RESTART: C:/Users/Victor/Google
numbers(2).py
150
>>>
```

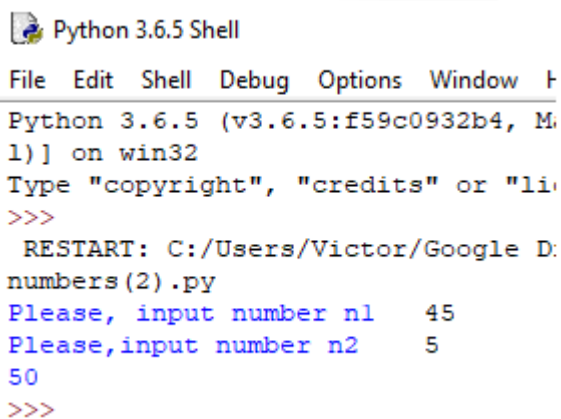
Now, we want to change our numbers n1 and n2. What to do? We have to change our code. However, introduce command input, that allows us to replace our numbers using keyboard control. Code is shown below.

```
n1=input('Please, input number n1')
n2=input('Please, input number n2')

n1=int(n1)
n2=int(n2)

print(n1+n2)
```

Result is presented on the Shell Screen



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window F
Python 3.6.5 (v3.6.5:f59c0932b4, M
1)] on win32
Type "copyright", "credits" or "li
>>>
RESTART: C:/Users/Victor/Google D:
numbers(2).py
Please, input number n1 45
Please, input number n2 5
50
>>>
```

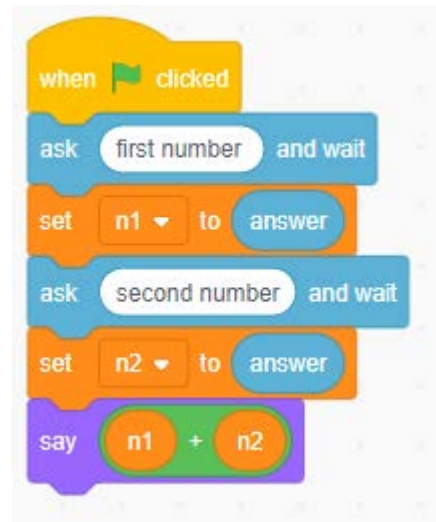
First code line prompts the user to input first number n1 and waits while you type the number. After you type first number, second code line prompts you to input second number and waits till you type it. Line 3 and 4 convert first number and second numbers into digital type (numbers typed according to 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

```
n1=input('Please, input number n1')
n2=input('Please,input number n2')

n1=int(n1)
n2=int(n2)

print(n1+n2)
```



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

1. Example #1(subtraction of two numbers)

```
n1=input('Please, input number n1')
n2=input('Please,input number n2')

n1=int(n1)
n2=int(n2)

print(n1-n2)
```

Result

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28
1)] on win32
Type "copyright", "credits" or "license
>>>
RESTART: C:/Users/Victor/Google Drive/I
numbers(2).py
Please, input number n1 36
Please,input number n2 23
13
>>>
```

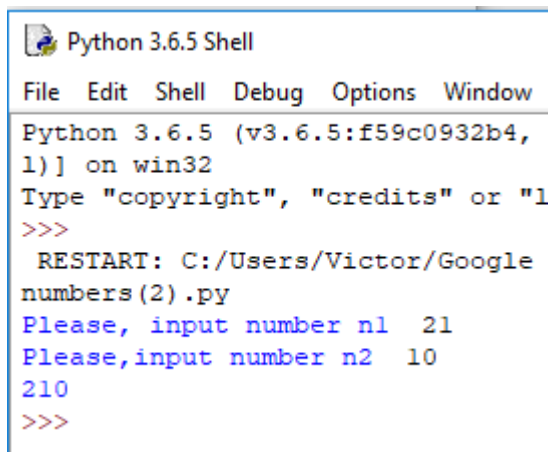
2. Example #2 (Product of two numbers)

```
n1=input('Please, input number n1')
n2=input('Please,input number n2')

n1=int(n1)
n2=int(n2)

print(n1*n2)
```

Result



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window
Python 3.6.5 (v3.6.5:f59c0932b4,
1)] on win32
Type "copyright", "credits" or "1
>>>
RESTART: C:/Users/Victor/Google
numbers(2).py
Please, input number n1  21
Please,input number n2  10
210
>>>
```

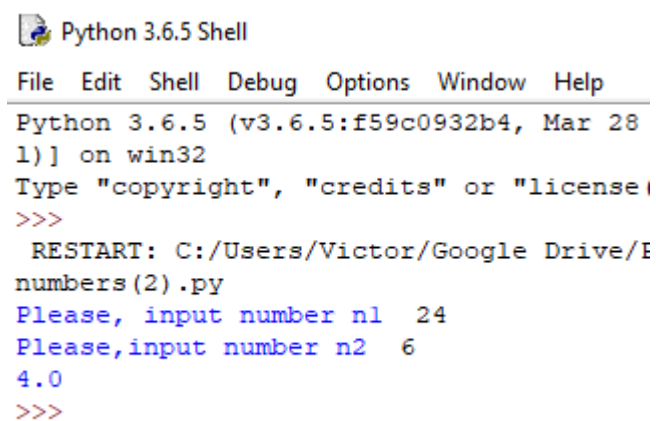
3. Example #3 (Division of two numbers)

```
n1=input('Please, input number n1')
n2=input('Please,input number n2')

n1=int(n1)
n2=int(n2)

print(n1/n2)
```

Result



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28
1)] on win32
Type "copyright", "credits" or "license
>>>
RESTART: C:/Users/Victor/Google Drive/E
numbers(2).py
Please, input number n1  24
Please,input number n2  6
4.0
>>>
```

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

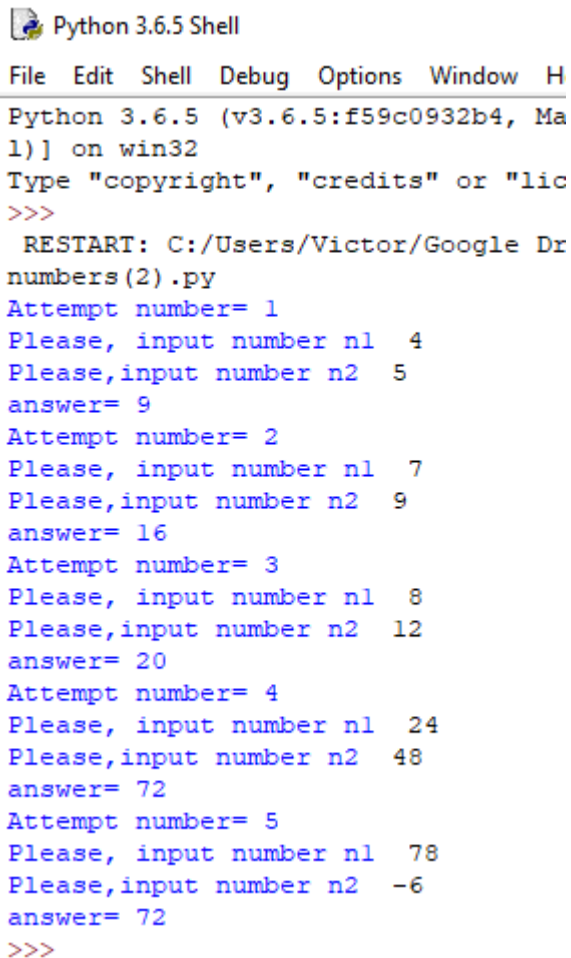
```
for i in range (5):
    print('Attempt number=', i+1)

    n1=input('Please, input number n1')
    n2=input('Please,input number n2')

    n1=int(n1)
    n2=int(n2)

    print('answer=',n1+n2)
```

Result (5 times)



Python 3.6.5 Shell

File Edit Shell Debug Options Window H

Python 3.6.5 (v3.6.5:f59c0932b4, Ma
l)] on win32

Type "copyright", "credits" or "lic
>>>

RESTART: C:/Users/Victor/Google Dr
numbers(2).py

Attempt number= 1
Please, input number n1 4
Please,input number n2 5
answer= 9

Attempt number= 2
Please, input number n1 7
Please,input number n2 9
answer= 16

Attempt number= 3
Please, input number n1 8
Please,input number n2 12
answer= 20

Attempt number= 4
Please, input number n1 24
Please,input number n2 48
answer= 72

Attempt number= 5
Please, input number n1 78
Please,input number n2 -6
answer= 72

>>>

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.

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

a.

```
import turtle
t = turtle.Turtle('turtle')
t.color('blue')
t.pensize(5)
x=20
for n in range (8):
    t.fd(x)
    t.left(45)
```



b.

```
import turtle
t = turtle.Turtle('turtle')
t.color('blue')
t.pensize(5)
x=35
for n in range (8):
    t.fd(x)
    t.left(45)
```



c.

```
import turtle
t = turtle.Turtle('turtle')
t.color('blue')
t.pensize(5)
x=35
for n in range (8):
    t.fd(x)
    t.left(45)
```



6. Example #2 (Hexagon, side length is equal 50, turn angle is variable $\text{angle}=60$)

```
import turtle
t = turtle.Turtle('turtle')
t.color('blue')
t.pensize(5)
angle=(60)
for n in range(6):
    t.fd(50)
    t.left(angle)
```



7. Example #3 (Changing distance between turtles)

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window
Python 3.6.5 (v3.6.5:f59c0932b4,
1) on win32
Type "copyright", "credits" or '
>>>
===== RESTART: C:/Users/Victor,
0
40
80
120
160
>>>

import turtle
t = turtle.Turtle('turtle')
t.color('blue')
t.shapesize(2)
t.up()
for m in range(5):
    t.fd(m*40)
    print(m*40)
    t.stamp()
```

Shell shows printed values

Code

Output

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 (fourth step), and m variable takes value equal 4 (fifth step).

According to our code distance between turtles is shown below



8. Example #4 (Changing of the turtle shape size)

- a. (Distance between turtles is $x=100$, size of the turtle changes in the loop; smallest one is equal 1, largest one is 4)

```

import turtle
t = turtle.Turtle('turtle')
t.color('blue')
t.pensize(5)
x=100
t.up()
t.hideturtle()
t.goto(-200,0)
t.showturtle()
for n in range (4):
    t.fd(x)
    t.shapesize(1+n)
    t.stamp()

```



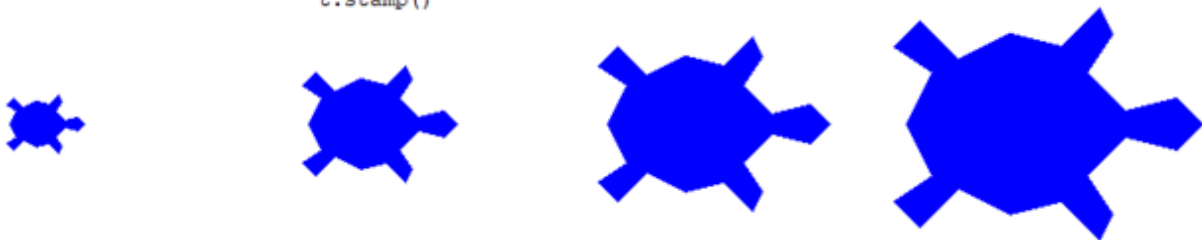
In this example shape size of the turtle changes according to the following law: $(1+n)$, where n is a variable. Let's explain. Firstly turtle has a shape size equal $(1+0)=1$ because the first number n of the loop $n=0$. During the second loop cycle turtle has a shape size equal $(1+1)=2$ because the 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 $n=3$

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

```

import turtle
t = turtle.Turtle('turtle')
t.color('blue')
t.pensize(5)
x=200
t.up()
t.hideturtle()
t.goto(-400,0)
t.showturtle()
for n in range (4):
    t.fd(x)
    t.shapesize(2*(1+n))
    t.stamp()

```



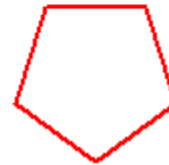
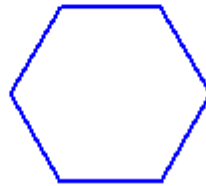
In this example shape size of the turtle changes (not distance between turtles) according to the following law: $2*(1+n)$. Let's explain. First turtle has a shape

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 $n=3$.

9. Example #5 (Number of polygons)

```
import turtle
t = turtle.Turtle()
t.color('blue')
t.pensize(2)
#t.hideturtle()
side_length=50
num_sides=6
angle=360/num_sides
for m in range(num_sides):
    t.fd(side_length)
    t.right(angle)

t.color('red')
num_sides=5
angle=360/num_sides
t.up()
t.goto(200,0)
t.down()
t.setheading(0)
for m in range(num_sides):
    t.fd(side_length)
    t.right(angle)
```



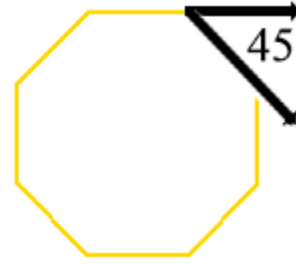
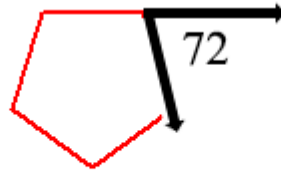
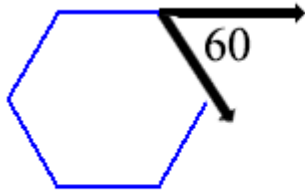
Let's explain this code. Code has three variables: the name of the first is `side_length`; second one is `num_sides` and third is `angle`, which is equal $360/\text{num_sides}$. If `num_sides=5` we draw pentagon, because for pentagon angle has to be $360/5=72$. If `num_sides=6` we draw hexagon, because 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.

```
for m in range(num_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.

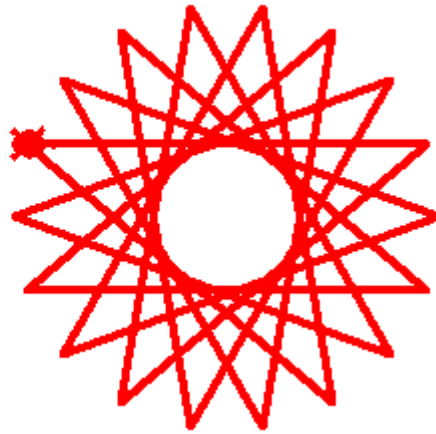


10.Example #6 (Number of Stars)

a.

```
import turtle
t = turtle.Turtle('turtle')
t.color('red')
t.pensize(4)
d=200
angle=220

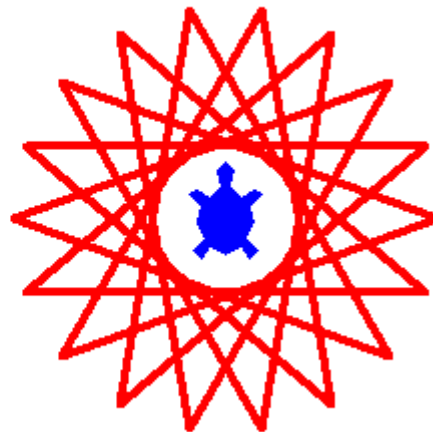
for i in range(18):
    t.fd(d)
    t.left(angle)
```



b.

```
import turtle
t = turtle.Turtle('turtle')
t.color('red')
t.pensize(4)
d=200
angle=220

for i in range(18):
    t.fd(d)
    t.left(angle)
t.up()
t.goto(100,-40)
t.setheading(90)
t.color('blue')
t.shapesize(2)
```



Challenges:

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

