



Lesson 9: Strings, Lists

As we discussed before (Lesson #7), Python can remember *values* of several types, including **number values** (like 7, 42, or even 98.6) and **strings**. A word- or any collection of letters and symbols- is called a string (letters, symbols, words, sentences). Strings can include letters, numbers, spaces, and symbols such as stops and commas. You define a string by enclosing the characters in single (' ') or double quotes (" "), like so. For example, variable a=' How are you' is a string.

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2016) on win32
Type "copyright", "credits" or "license()"
>>> a= 'How are you'
>>> print(a)
How are you
>>>
```

We can add two string variables. Example below shows how to do it. (Code – from left side, result is from right side)

```
a='How are you '
b='Good, Thanks'
c=a+b
print(c)

File Edit Shell Debug Options Window
Python 3.6.5 (v3.6.5:f59c0932b4, 1
1) on win32
Type "copyright", "credits" or "l.
>>>
===== RESTART: C:/Users/Victor/G
How are you Good, Thanks
>>>
```

When we add two string variable a to string variable b, we get new string variable, which is a combination of two variables a and b. **Keep in mind that you can't add string type data and number type data.**

```

a='How are you '
b=4
c=a+b
print(c)

```

```

File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16
1)] on win32
Type "copyright", "credits" or "license()" for m
>>>
===== RESTART: C:/Users/Victor/Google Drive/Pytl
Traceback (most recent call last):
  File "C:/Users/Victor/Google Drive/Python Proj
dule>
      c=a+b
TypeError: must be str, not int
>>>

```

We can add two numbers if both of them are string type, like this:

```

File Edit Shell Debug Options
Python 3.6.5 (v3.6.5:f59c0
1)] on win32
Type "copyright", "credits
>>>
===== RESTART: C:/Users/V
2314
>>>

```

In this example both '23' and '14' are string made of numbers, not number types. So, when you add them together you get a longer string '2314' that is combination of the two strings.

Sometimes you need to operate with numbers and string in one program. How to do it. Let's discuss the following example (**this code uses function called input**). Don't forget click button ENTER after input data. So, this code looks, like this:

```

Nick_age=input('How old is Nick?')
Vic_age=input('How old is Vic?')

print(Nick_age+Vic_age)

```

Code requests the age of Nick and Vic and must print the sum of their age. Let's see the result

```

Python 3.6.5 Shell
File Edit Shell Debug Options
Python 3.6.5 (v3.6.5:f59c(
1)] on win32
Type "copyright", "credits
>>>
===== RESTART: C:/Users/\
How old is Nick?12
How old is Vic?10
1210
>>>

```

Result is wrong because both input ages are string variables. How to fix this problem? Very simple. You need to convert string variable number 12 and string variable number 10 to the integer numbers.

```
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/G
How old is Nick?12
How old is Vic?10
22
>>>
```

If you want the input to be on its own line, then you could just add symbol `\n` as shown below:

```
Nick_age=input('How old is Nick?\n')
Vic_age=input('How old is Vic?\n')

Nick_age=int(Nick_age)
Vic_age=int(Vic_age)

print(Nick_age+Vic_age)
```

Result:

```
File Edit Shell Debug Options
Python 3.6.5 (v3.6.5:f59c0
1) on win32
Type "copyright", "credits
>>>
===== RESTART: C:/Users/V
How old is Nick?
12
How old is Vic?
10
22
>>>
```

If you want to add comments to the result you need modify code as:

```
Nick_age=input('How old is Nick?\n')
Vic_age=input('How old is Vic?\n')

Nick_age=int(Nick_age)
Vic_age=int(Vic_age)

print('Nick_age+Vic_age=',Nick_age+Vic_age)
```

Last line includes string data `'Nick_age+Vic_age='` and integer number `Nick_age+Vic_age`.

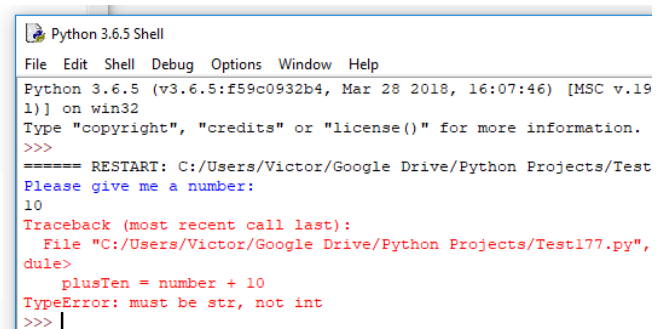
Result:

```
File Edit Shell Debug Options
Python 3.6.5 (v3.6.5:f59c
1)] on win32
Type "copyright", "credit
>>>
===== RESTART: C:/Users/
How old is Nick?
10
How old is Vic?
12
Nick_age+Vic_age= 22
>>>
```

Another example

```
print ("Please give me a number: ")
number = input()

plusTen = number + 10
print ("If we add 10 to your number, we get ", + plusTen)
```

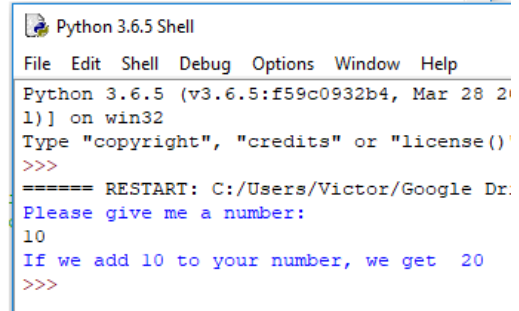


```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.19
1)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Victor/Google Drive/Python Projects/Test
Please give me a number:
10
Traceback (most recent call last):
  File "C:/Users/Victor/Google Drive/Python Projects/Test177.py",
    dule>
      plusTen = number + 10
TypeError: must be str, not int
>>> |
```

We can't get correct result because line #3 of the code adds string variable "number" with integer 10. Again, to fix this problem we have to convert string variable "number" to integer number. Below, we show corrected code and accurate result

```
print ("Please give me a number: ")
number = input()
number=int(number)

plusTen = number + 10
print ("If we add 10 to your number, we get ", + plusTen)
```



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 20
1)] on win32
Type "copyright", "credits" or "license()"
>>>
===== RESTART: C:/Users/Victor/Google Dr.
Please give me a number:
10
If we add 10 to your number, we get 20
>>>
```

Example: Bad or good weather with Python

```

answer=input ('Rainy?--yes or no\n')
if answer=='yes':
    print('bad weather')
else:
    print('good weather')

```

```

File Edit Shell Debug Optic
Python 3.6.5 (v3.6.5:f5
1)] on win32
Type "copyright", "cred
>>>
===== RESTART: C:\User
Rainy?--yes or no
yes
bad weather
>>>

```

Example: what year now → usual or intercalary

```

keep=True
while keep:

    year = input('what is the year now=\n')
    year=int(year)

    if year % 4 != 0:
        print("usual year")
    elif year % 100 == 0:
        if year % 400 == 0:
            print("intercalary year")
        else:
            print("usual year")
    else:
        print("intercalary year")

    answer=input('Dou you want to continue?, yes or no \n')
    if answer=='yes':
        keep=True
    else:
        break

what is the year now=
2019
usual year
Dou you want to continue?, yes or no
yes
what is the year now=
2018
usual year
Dou you want to continue?, yes or no
yes
what is the year now=
2016
intercalary year
Dou you want to continue?, yes or no
no
>>>

```

Example: Winter, Spring, Summer, Fall

```
keep=True
while keep:

    a = input('Input month number any number from 1 to 12\n')
    b=int(a)
    if b>= 1 and b<=2 or b==12:
        print('Winter')
    elif b>= 3 and b<=5 :
        print('Spring')
    elif b>= 6 and b<=8 :
        print('Summer')
    elif b>=9 and b<=11 :
        print('Fall')
    else:
        break
```

```
===== RESTART: C:\Users\Victor\Google Drive
Input month number any number from 1 to 12
12
Winter
Input month number any number from 1 to 12
5
Spring
Input month number any number from 1 to 12
6
Summer
Input month number any number from 1 to 12
8
Summer
Input month number any number from 1 to 12
14
>>>
```

You know that 10 multiplied by 3 is equal to 30, of course. But what's 10 multiplied by '3' ('3' is a string). Here's Python's answer:

```
>>> 10*'3'
'3333333333'
>>>
```

We see that the result is 10 times of digit 3.

Example: Heart pattern with strings

```
#CODE
print('RESULT')

print(' ', '8'*2, ' ', '8'*2)
print('8', ' ', '8', ' ', '8')
print('8', ' ', ' ', ' ', '8')
print(' ', '8', ' ', ' ', '8')
print(' ', '8', ' ', ' ', '8')
print(' ', ' ', '8')

print(' ', '*'*2, ' ', '*'*2)
print('*', ' ', '*', ' ', '*')
print('*', ' ', ' ', ' ', '*')
print(' ', '*', ' ', ' ', '*')
print(' ', '*', ' ', ' ', '*')
print(' ', ' ', '*')

>>>
```

```
RESULT
  88 88
8 8 8
8 8
8 8
8 8
  8
  ** **
* * *
* * *
* *
  *
>>>
```

Example: Find prime number

```
def prime_number(x):
    global n
    if x >= 2:

        for y in range(2,x):

            k=x*y

            if k==0:
                n=0
                print('usual')
                break

            n=1
forward=True

while forward:
    answer=input("Do you want to find prime number? yes or no\n")

    if answer=='no':
        break

    q= input("Please enter a number:\n")
    a=int(q)
    prime_number(a)

    if n==1:
        print('prime')
```

```
Do you want to find prime number? yes or no
yes
Please enter a number:
163
prime
Do you want to find prime number? yes or no
yes
Please enter a number:
12345763
usual
Do you want to find prime number? yes or no
yes
Please enter a number:
9301
usual
Do you want to find prime number? yes or no
no
>>>
```

Now, Let's demonstrate example how to turn turtle with input code

1. Example #1 (Turtle turns 45 degrees)

Code:

```
1 import turtle
2 t1=turtle.Turtle('turtle')
3 t1.shapesize(5)
4 t1.color('red')
5
6 a=input('Please input turtle angle?=\n')
7 b=int(a)
8 print('turtle angle=',b)
9 t1.setheading(b)
```

Result:

```
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\Google
Please input turtle angle?=
45
turtle angle= 45
>>>
```



2. Example #2

```
1 import turtle
2 t1=turtle.Turtle('turtle')
3 t1.shapesize(5)
4 t1.color('red')
5
6 a=input('Please input turtle angle?=\n')
7 b=int(a)
8 print('turtle angle=',b)
9 t1.setheading(b)
```

```
Please input turtle angle?=
135
turtle angle= 135
>>>
```



```
Please input turtle angle?=
245
turtle angle= 245
>>>
```



```
Please input turtle angle?=
330
turtle angle= 330
>>>
```



You can include input code line in the loop. In this case program requests you to input data a few times and you will see the results for each request:


```

import turtle
t1=turtle.Turtle('turtle')
t1.shapesize(5)
t1.color('red')

for i in range (10):

    a=input('Please input turtle angle?=\n')
    b=int(a)
    print('turtle angle=',b)
    t1.left(b)

```

USING LISTS

In addition to strings and number values, variables can also contain lists. A list is a group of values, separated by commas, between square brackets, []. We can store any value types I lists, including numbers and strings. Let's go to list example, called `color_list`:

```
color_list=['red', 'blue', 'gold', 'green', 'yellow', 'pink']
```

This list consists of 6 string variables. If you want to do anything with one of the words from that list, you can use something called the index- the position of the item in the list. So, a list is a structure in Python where items are kept in order. Each entry is given a number that you can use to refer back to it.

```
color_list[0]= 'red'
```

```
color_list[1]= 'blue'
```

```
color_list[2]= 'gold'
```

```
color_list[3]= 'green'
```

```
color_list[4]= 'yellow'
```

```
color_list[5]= 'pink'
```

Below it is shown a list that includes integer numbers

```
number_list=[3,5,7,11,13,17,19,23,29,31]
```

Now we create our first program with a list. Let's calculate the following sums:

$$S1=3+5+7+11+13+17+19+23+29+31$$

$$S2=3^2+5^2+7^2+11^2+13^2+17^2+19^2+23^2+29^2+31^2$$

$$S3=3^4+5^4+7^4+11^4+13^4+17^4+19^4+23^4+29^4+31^4$$

All of these integers are prime numbers. Code, shown below, calculates these values S1, S2, and S3. To make calculations we use list called *num*.

3. Example #3 (Math calculations using List with integer values)

```
num=[3,5,7,11,13,17,19,23,29,31]

S1=0
S2=0
S3=0

for n in range(10):
    S1=S1+num[n]
    S2=S2+num[n]**2
    S3=S3+num[n]**4

print('S1=',S1)
print('S2=',S2)
print('S3=',S3)
```

```
S1= 158
S2= 3354
S3= 2170794
>>>
```

Result

4. Example #4 (Sort list numbers in the order from smallest item to largest values)

Code:

```
list1 = [10, 20, 4, 45, 99,15,44,38,72,125,1,999,348]

# sorting the list
list1.sort()
print(list1)
print('Minimum number in the list=',list1[0])
```

Result:

```
===== RESTART: C:/Users/Victor/Google Drive/Python Project
[1, 4, 10, 15, 20, 38, 44, 45, 72, 99, 125, 348, 999]
Minimum number in the list= 1
>>>
```

In this program we used line `list1.sort` that sorts all numbers in order from minimum to maximum value.

5. Example #5 (Find smallest and greatest items in the list with numbers)

Code:

```
list = [10, 20, 4, 45, 99,15,44,38,72,125,1,999,348]

# sorting the list
a=min(list)
b=max(list)

print(list)
print('Smallest number in the list=',a)
print('Greatest number in the list=',b)
```

Result:

```
===== RESTART: C:/Users/Victor/Google Drive/Python Projec
[10, 20, 4, 45, 99, 15, 44, 38, 72, 125, 1, 999, 348]
Smallest number in the list= 1
Greatest number in the list= 999
>>>
```

If you want to calculate the number of items in the list you have to use command `len(list)`, for example, as shown below

6. Example #6 (Calculate the number of items in the list)

Code:

```
list = [10, 20, 4, 45, 99,15,44,38,72,125,1,999,348]

# sorting the list
a=min(list)
b=max(list)
c=len(list)

print(list)

print('Smallest number in the list=',a)
print('Greatest number in the list=',b)
print('Number of list items=',c)
```

Result:

```
===== RESTART: C:/Users/Victor/Google Drive/Python Pr
[10, 20, 4, 45, 99, 15, 44, 38, 72, 125, 1, 999, 348]
Smallest number in the list= 1
Greatest number in the list= 999
Number of list items= 13
>>>
```

7. Example #7 (Sorting of the list with strings variables)

```
#CODE
list = ['g','c','h','a','f','i','b','e','d']
list.sort()
print('-----')
print('RESULT')
print(list)

-----
RESULT
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
>>>
```

All letters are sorted according to the alphabet.

8. Example #8 (Add to the end of a list additional items)

You can add any item numbers to the end of the list whatever you want, for example,

Code:

```
list=[]
list1=[]
list2=[]

for i in range(10):
    list.append(i)
    list1.append(i*i)
    list2.append(i**3)

print('list=',list)
print('list1=',list1)
print('list2=',list2)
```

Result:

```
list= [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
list1= [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
list2= [0, 1, 8, 27, 64, 125, 216, 343, 512, 729]
>>>
```

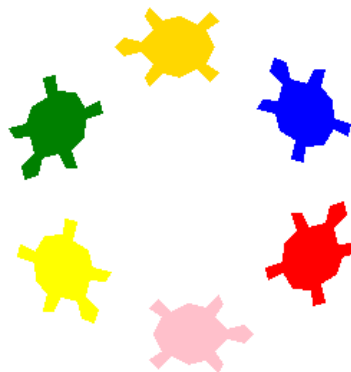
In this example we have three lists: list; list1; and list2. All of them initially are empty. However using lines “list.append(i)”, “list1.append(i*i)”, and “list2.append(i**3)” we add to the end of each list integer numbers equal
i for list,
i*i for list1,
i**3 for list2.

9. Example #9 (Number of different colour turtles, code uses list with string variables)

```
import turtle
t=turtle.Turtle('turtle')
t.shapesize(3)
t.up()

color_list=['red','blue','gold','green','yellow','pink']

for i in range(6):
    t.color(color_list[i])
    t.circle(100, 60)
    t.stamp()
```



Here we set up the colour of turtle with a line t.color(color_list[i]), where i is an integer number that takes sequentially the following values 0,1,2,3,4,5.

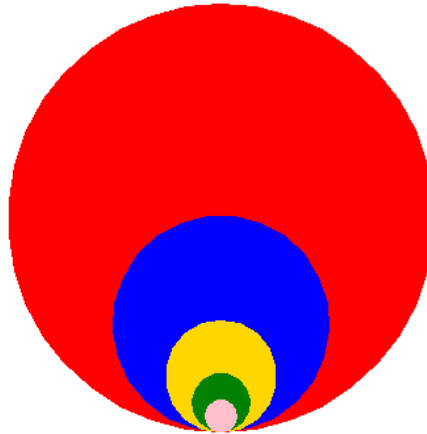
10.Example #10 (Example shows the combination of number and string lists)

a.

```
import turtle
t=turtle.Turtle('turtle')
t.pensize(5)

color_list=['red','blue','gold','green','pink']
num=[32,16,8,4,2]

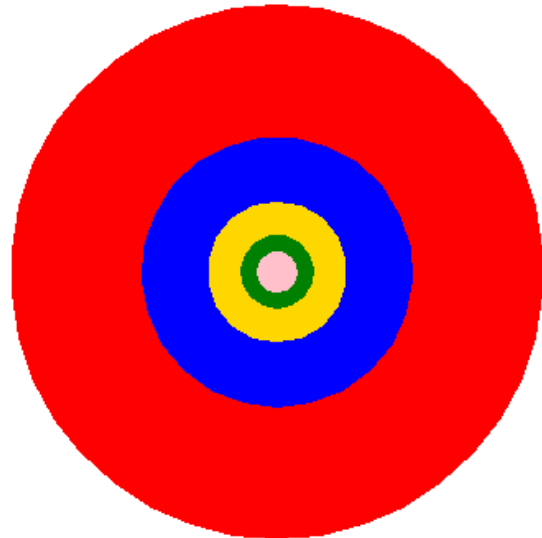
for i in range (5):
    t.color(color_list[i])
    t.begin_fill()
    t.circle(5*num[i])
    t.end_fill()
t.hideturtle()
```



```
import turtle
t=turtle.Turtle('turtle')
t.pensize(5)

color_list=['red','blue','gold','green','pink']
num=[32,16,8,4,2]

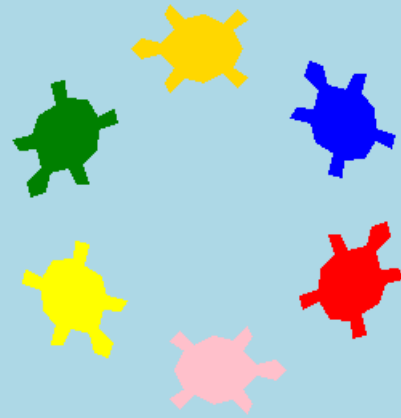
for i in range (5):
    t.color(color_list[i])
    t.up()
    t.goto(0,-5*num[i])
    t.down()
    t.begin_fill()
    t.circle(5*num[i])
    t.end_fill()
t.hideturtle()
```



11.Example #11 (Colour turtles with colour background)

```
import turtle
t=turtle.Turtle('turtle')
t.shapesize(3)
t.up()
turtle.bgcolor('light blue')
color_list=['red','blue','gold','green','yellow','pink']

for i in range(6):
    t.color(color_list[i])
    t.circle(100, 60)
    t.stamp()
```



Line `turtle.bgcolor('light blue')` creates background light blue colour.

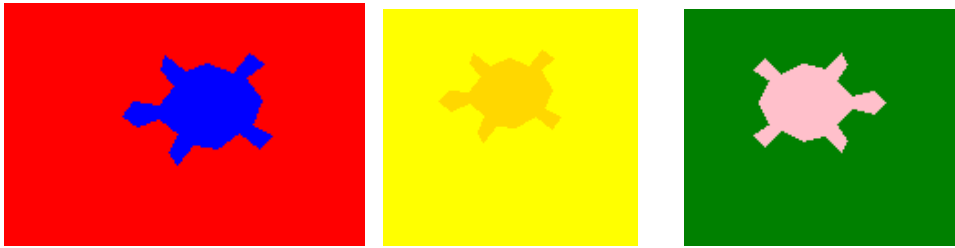
12.Example #12 (Colour turtles with changing colour background)

```
import turtle
t=turtle.Turtle('turtle')
t.shapesize(3)
t.up()

clr=['red','yellow','green']
color_list=['blue','gold','pink']

for m in range (3):

    turtle.bgcolor(clr[m])
    t.color(color_list[m])
    t.circle(100)
```



13.Example #13 (Drawing with colour background)

We can set up black back-ground colour with the line
 turtle.bgcolor('black')

a.

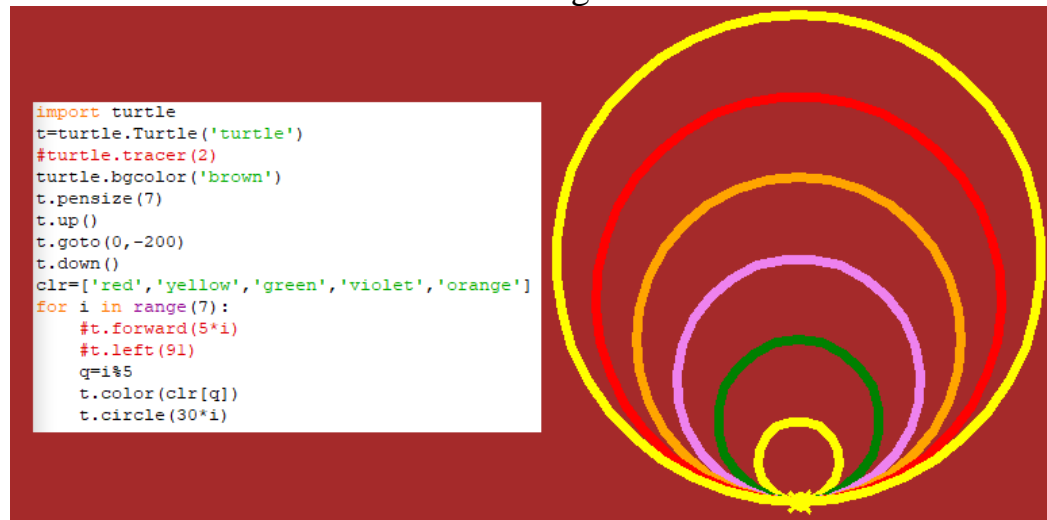
```
import turtle
t=turtle.Turtle('turtle')
turtle.bgcolor('black')
clr=['red', 'yellow', 'green', 'violet', 'orange', \
     'purple', 'blue', 'gold', 'pink', 'light blue']

for i in range (10):
    t.color(clr[i])
    for m in range(4):
        t.fd(10+25*i)
        t.left(90)
```

b.

```
import turtle
t=turtle.Turtle('turtle')
turtle.bgcolor('black')
clr=['red', 'yellow', 'green', 'violet', 'orange']
for i in range (5):
    t.color(clr[i])
    for m in range(4):
        t.circle(10+25*i)
        t.left(90)
```

You can use any colour, you want to use with Python code. For, example, code shown below uses brown colour background



c.

Code (Blue background)

```
import turtle
t=turtle.Turtle('turtle')
turtle.bgcolor('blue')
t.pensize(5)
clr=['red','yellow','green','violet','orange']
for i in range(100):
    t.forward(5*i)
    t.left(91)
    q=i%5
    t.color(clr[q])
```

Variable $q=i\%5$ (line nine in the code) tells Python that we use the only 5 colours in the colour list, numbered 0 to 4, and rotate them every I changes. In this case, our colour list has five colours, so, we'll rotate through these five colours over and over.

Result

