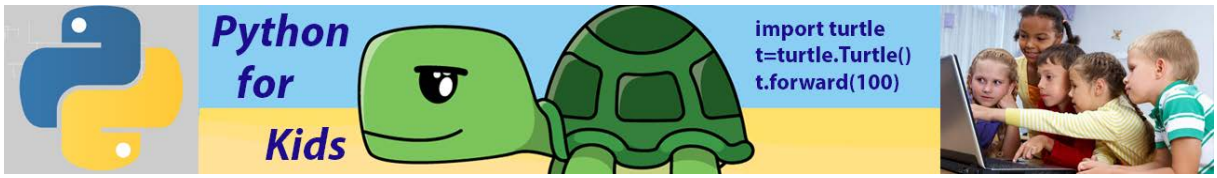


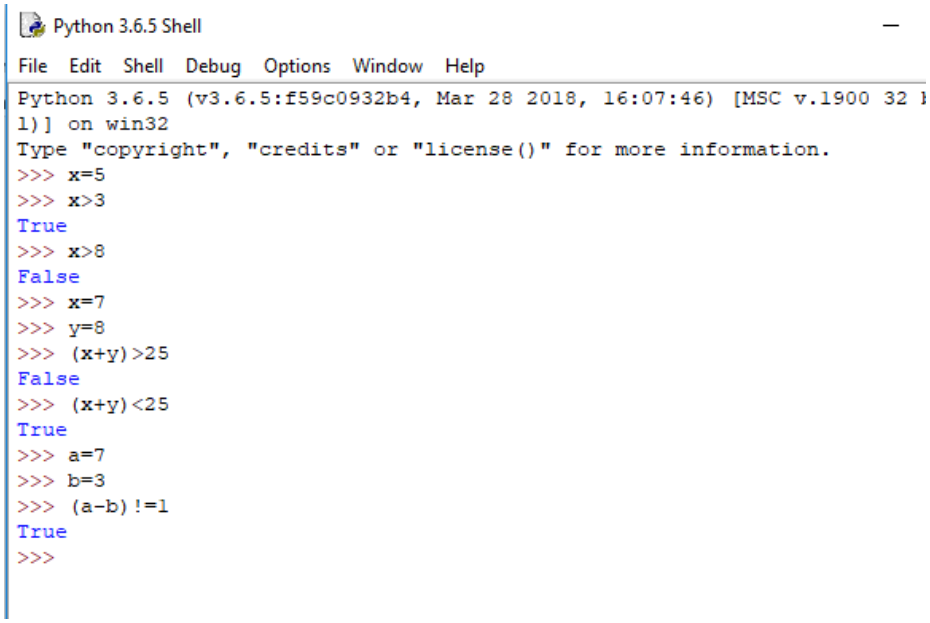
Lesson 8: Logical Operators, Conditional Statements, while loop

Logical operators are used to compare variables against numbers or against other variables. The resulting answer is either True or False.

Summary for Comparison Operations:

Code Instruction		What it does		
Math symbol	Python operator	Meaning	Example	Result (Boolean Values)
>	>	Greater than	12>11	True
<	<	Less than	25<10	False
≤	<=	Less than or equal to	1<=2	True
≥	>=	Greater or equal to	1>=3	False
=	= =	Equal to	5= =10	False
≠	!=	Not equal to	5 !=10	True

Let's go to the Python Shell and try entering some of the expressions shown in the Summery.



Every conditional expression will evaluate to either True or False in Python. Those are only two Boolean values, and the capital T in True and capital F in False are requared.

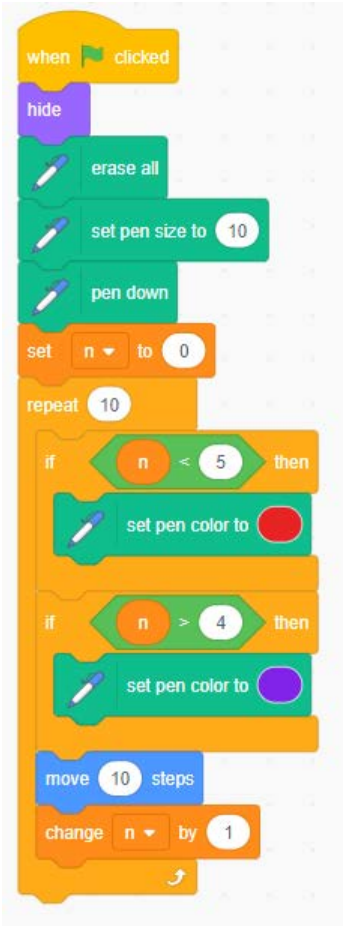
Conditional statement *if*

The ***if*** command means that if a condition is True, then the program runs a block of commands. If the condition isn't True, the block is skipped. The block after the ***if*** command is always indented by four spaces.

Let's go to example:

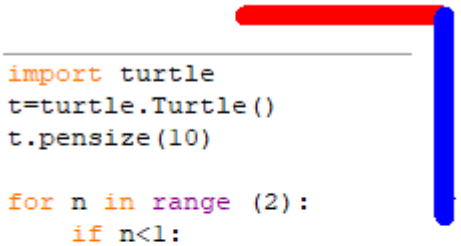
1. **Example #1** (From left side we show Python code, from right side- Scratch code for comparison)

```
import turtle
t=turtle.Turtle()
t.pensize(10)

for n in range (10):
    if n<5:
        t.color('red')

    if n>=5:
        t.color('blue')

    t.fd(20)
```



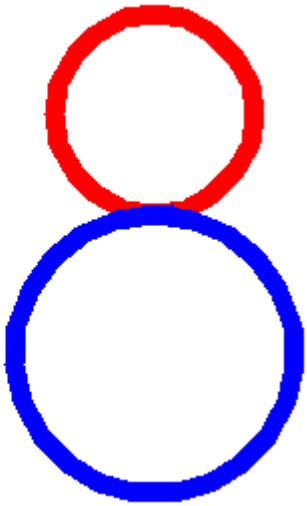
As we can see program (in a ***for*** loop) assigns the following values 0,1,2,3,4,5,6,7,8,9 to the variable n. Very important than if variable n<5 colour of line is red, if variable >=5 colour of line is blue.

2. **Example #2** (Two Lines)



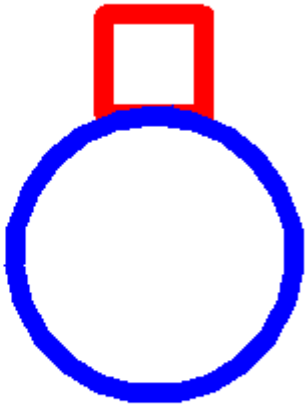
```
import turtle
t=turtle.Turtle()
t.pensize(10)

for n in range (2):
    if n<1:
        t.color('red')

    if n>0:
        t.color('blue')
        t.right(90)

    t.fd(100)
```

3. **Example #3 (**Two circles)



```
import turtle
t=turtle.Turtle()
t.pensize(10)

for n in range (2):
    if n<1:
        t.color('red')
        t.circle(50)

    if n>0:
        t.color('blue')
        t.right(180)
        t.circle(70)
```

4. **Example #4** (Square and Circle)



```
import turtle
t=turtle.Turtle()
t.pensize(10)

for n in range (2):
    if n<1:
        t.color('red')
        for q in range(4):
            t.fd(50)
            t.left(90)

    if n>0:
        t.color('blue')
        t.right(180)
        t.up()
        t.fd(-25)
        t.down()
        t.circle(70)
```

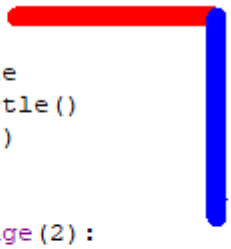
5. **Example #5** (Square, circle, and Turtle)

```
import turtle
t=turtle.Turtle()
t.hideturtle()

for n in range (3):

    t.pensize(10)
    if n==0:
        t.color('red','red')
        t.begin_fill()
        for q in range(4):
            t.fd(50)
            t.left(90)
        t.end_fill()

    if n==1:
        t.color('blue','blue')
        t.right(180)
        t.up()
        t.fd(-25)
        t.down()
        t.begin_fill()
        t.circle(70)
        t.end_fill()
        t.hideturtle()

    if n==2:
        t=turtle.Turtle('turtle')
        t.up()
        t.color('gold')
        t.goto(25,-70)
        t.setheading(90)
        t.shapesize(3)
        t.showturtle()
```

<u>Conditional statement</u> ***if-else***

The ***if*** command can be combined with an ***else*** command. This combination means that if something is True, one thing happens, and if not else happens. Let's see our Example #3 and build the same program using ***if-else*** code

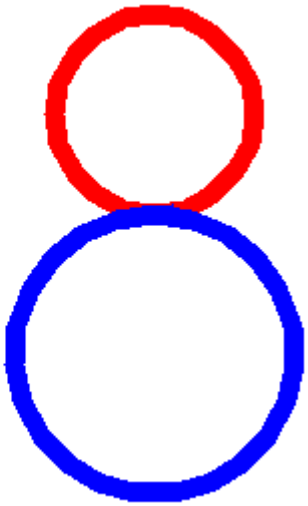
6. **Example #6** (Example #2 created with condition ***if-else***)



```
import turtle
t=turtle.Turtle()
t.pensize(10)


for n in range(2):
    if n<1:
        t.color('red')
    else:
        t.color('blue')
        t.right(90)
    t.fd(100)
```

7. **Example #7** (Example #3 with condition ***if-else***)

```
import turtle
t=turtle.Turtle()
t.pensize(10)


for n in range(2):
    if n<1:
        t.color('red')
        t.circle(50)
    else:
        t.color('blue')
        t.right(180)
        t.circle(70)
```

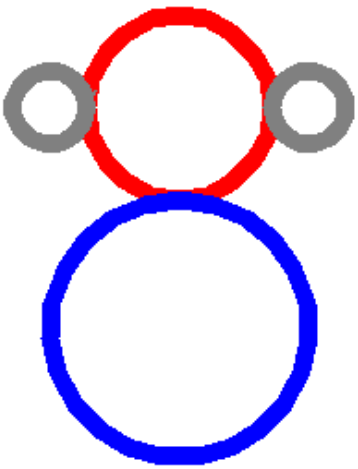


<u>Conditional statement</u> ***if-elif-else***

The ***elif*** command is short for ***else-if.*** It means that if something is True, do one thing; otherwise, check if something else is True and do something else if it is. The following program (Example #5 written using this conditional statement command)

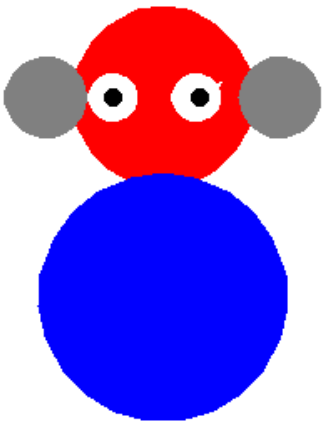
8. **Example #8** (Toy)

```
import turtle
t=turtle.Turtle()
t.pensize(10)
for n in range(4):
    if n==0:
        t.color('red')
        t.circle(50)

    elif n==1:
        t.color('blue')
        t.right(180)
        t.circle(70)

    elif n==2:
        t.up()
        t.goto(50,50)
        t.color('grey')
        t.down()
        t.setheading(-90)
        t.circle(20)

    else:
        t.up()
        t.goto(-50,50)
        t.color('grey')
        t.down()
        t.setheading(-90)
        t.circle(-20)
```



9. Example #9

```
import turtle
t=turtle.Turtle()
#turtle.tracer(3)
t.pensize(10)
for n in range(6):
    if n==0:

        t.color('red','red')
        t.begin_fill()
        t.circle(50)
        t.end_fill()
    elif n==1:
        t.color('blue')
        t.right(180)
        t.begin_fill()
        t.circle(70)
        t.end_fill()

    elif n==2:
        t.up()
        t.goto(50,50)
        t.color('grey')
        t.down()
        t.setheading(-90)
        t.begin_fill()
        t.circle(20)
        t.end_fill()

    elif n==3:
        t.up()
        t.goto(-50,50)
        t.color('grey')
        t.down()
        t.setheading(-90)
        t.begin_fill()
        t.circle(-20)
        t.end_fill()

    elif n==4:
        t.up()
        t.goto(-20,50)
        t.color('white')
        t.down()
        t.setheading(-90)
        t.begin_fill()
        t.circle(-10)
        t.end_fill()

    else:
        t.up()
        t.goto(30,50)
        t.color('white')
        t.down()
        t.setheading(-90)
        t.begin_fill()
        t.circle(-10)
        t.end_fill()

t=turtle.Turtle('circle')
t.up()
t.goto(-30,50)
t.color('black')
t.shapesize(0.5)
t.stamp()

t.goto(22,50)
t.color('black')
```



While loops

For loops are useful when you know how many times a task needs to be repeated. But sometimes you need a loop to keep repeating until something changes. A while loop keeps on going around as many times as it needs to. A *while* loop keeps repeating as long as certain condition is true. This condition is called the loop condition and is either true or false.

10.Example #6

```
1  import turtle
2  t=turtle.Turtle()
3  t.hideturtle()
4  t.color('gold')
5  t.pensize(10)
6
7  n=0
8  while n<5:
9      t.fd(150)
10     t.left(90)
11     n=n+1
```


Code (Example #6) creates square using *while* loop. We start by creating a variable (line 7) named n which is set to 0. Code line 8: While loop checks for condition n<5. The current value of n is 0. Condition is true. Flow of control enters into while loop. Turtle draws horizontal line and turns heading by 90 degrees. Code line 11: n is incremented by1. Flow of control goes back to line 8. Now the value of n is 1 which is less then 5. The condition is true, and again the while loop is executed. This continues till n becomes 5, and the while condition becomes false. Execution of this code is completed.

while True: This loop runs forever. If you set the condition in a while loop to be True, it can never be false and the loop will never end. In some situations, it can be very useful.

11.Example #7 (This program will never end)

```
import turtle
t=turtle.Turtle()
t.hideturtle()
t.color('gold')
t.pensize(2)

while True:
    t.left(11.33)
    for n in range (18):
        t.fd(20)
        t.left(20)
```