a = 5
b = 3
c = a + b
print( "c: " + str(c) )
a = 5
b = 3
c = a + b
print( "c: " + str(c) )  c: 8

b = 30
a = b
print( a )
a = 5
b = 3
c = a + b
print( "c: " + str(c) )  
c: 8

b = 30
a = b
print( a )  
30

print( d )
```python
a = 5
b = 3
c = a + b
print( "c: " + str(c) )  # c: 8

b = 30
a = b
print( a )  # 30

print( d )  # error

4 = a
```
a = 5
b = 3
c = a + b
print("c: " + str(c))  # c: 8

b = 30
a = b
print(a)  # 30

print(d)  # error

4 = a  # error
a = 5
b = 3
c = a + b
c = "hello"
print( b + c )
```python
a = 5
b = 3
c = a + b
c = "hello"
print( b + c )
```
a = 5
b = 3
c = a + b
c = "hello"
print( b + c )

print( ??? )
a = 5
b = 3
c = a + b
c = "hello"
print( b + c )
print(str(b) + " " + c )) 3 hello
# --- DRILL ---
# write an expression that prints
# I am <age> years old and I live in <city>.

age = 53
city = "Berkeley"
# --- DRILL ---
# write an expression that prints
# I am <age> years old and I love in <city>.

age = 53
city = "Berkeley"

print( "I am " + str(age) + " years old and I live in " + city + "." )

I am 53 years old and I live in Berkeley.
# --- DRILL ---
# write an expression that prints
# I am <age> years old and I love in <city>.  
# but now subtract 5 years from your age  
# before printing

age = 53
city = "Berkeley"
# --- DRILL ---
# write an expression that prints
# I am <age> years old and I love in <city>.
# but now subtract 5 years from your age
# before printing

age = 53
city = "Berkeley"

print( "I am " + str(age-5) + " years old and I live in " + city + "." )

I am 48 years old and I live in Berkeley.
import drawSvg as draw

def draw_frame(y):
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Circle(0, y, 25, fill='lime'))
    return d

# bounce the ball up and down
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(100)
    anim.draw_frame(90)
    anim.draw_frame(80)
    anim.draw_frame(70)
    anim.draw_frame(60)
    anim.draw_frame(50)
    anim.draw_frame(40)
    anim.draw_frame(30)
    anim.draw_frame(20)
    anim.draw_frame(10)
    anim.draw_frame(0)
    anim.draw_frame(10)
    anim.draw_frame(20)
    anim.draw_frame(30)
    anim.draw_frame(40)
    anim.draw_frame(50)
    anim.draw_frame(60)
    anim.draw_frame(70)
    anim.draw_frame(80)
    anim.draw_frame(90)
    anim.draw_frame(100)
# rotate a square around a circular trajectory

\[(0,0) \to (R \times \cos(t), R \times \sin(t))\]
# rotate a square around a circular trajectory

(R x cos(t), R x sin(t))

import math

# convert to radians: deg*math.pi/180
math.cos(t)
math.sin(t)
# rotate a square around a circular trajectory

```python
import drawSvg as draw

# Draw a frame of the animation
def draw_frame(t):
    r = 100  # radius of circular trajectory
    x = r * math.cos(t*math.pi/180)  # x-on circle
    y = r * math.sin(t*math.pi/180)  # y-on circle
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Rectangle(x, y, 25, 25, fill='red'))
    return d

# rotate the square
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(0)
    anim.draw_frame(10)
    anim.draw_frame(20)
    anim.draw_frame(30)
```

# rotate a square around a circular trajectory

import drawSvg as draw

# Draw a frame of the animation
def draw_frame(t):
    r = 100  # radius of circular trajectory
    x = r * math.cos(t*math.pi/180)  # x-on circle
    y = r * math.sin(t*math.pi/180)  # y-on circle
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Rectangle(x, y, 25, 25, fill='red'))
    return d

# rotate the square
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(0)
    anim.draw_frame(10)
    anim.draw_frame(20)
    anim.draw_frame(30)
# rotate a square around a circular trajectory

```python
from drawSvg import draw

# Draw a frame of the animation

def draw_frame(t):
    r = 100  # radius of circular trajectory
    x = r * math.cos(t * math.pi / 180)  # x-on circle
    y = r * math.sin(t * math.pi / 180)  # y-on circle
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Rectangle(x, y, 25, 25, fill='red'))
    return d

# rotate the square

with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(0)
    anim.draw_frame(10)
    anim.draw_frame(20)
    anim.draw_frame(30)
```
# rotate a square around a circular trajectory

```python
import drawSvg as draw

def draw_frame(t):
    r = 100  # radius of circular trajectory
    x = r * math.cos(t*math.pi/180)  # x-on circle
    y = r * math.sin(t*math.pi/180)  # y-on circle
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Rectangle(x, y, 25, 25, fill='red'))
    return d

with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(0)
    anim.draw_frame(10)
    anim.draw_frame(20)
    anim.draw_frame(30)
```

# rotate a square around a circular trajectory

import drawSvg as draw

# Draw a frame of the animation
def draw_frame(t):
    r = 100  # radius of circular trajectory
    x = r * math.cos(t*math.pi/180)  # x-on circle
    y = r * math.sin(t*math.pi/180)  # y-on circle
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Rectangle(x, y, 25, 25, fill='red'))
    return d

# rotate the square
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(0)
    anim.draw_frame(10)
    anim.draw_frame(20)
    anim.draw_frame(30)
# rotate a square around a circular trajectory

```python
import drawSvg as draw

# Draw a frame of the animation

def draw_frame(t):
    r = 100  # radius of circular trajectory
    x = r * math.cos(t*math.pi/180)  # x-on circle
    y = r * math.sin(t*math.pi/180)  # y-on circle
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Rectangle(x, y, 25, 25, fill='red'))
    return d

# rotate the square

with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(0)
    anim.draw_frame(10)
    anim.draw_frame(20)
    anim.draw_frame(30)
```
import drawSvg as draw

def draw_frame(t):
    r = 100  # radius of circular trajectory
    x = r * math.cos(t*math.pi/180)  # x-on circle
    y = r * math.sin(t*math.pi/180)  # y-on circle
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Rectangle(x, y, 25, 25, fill='red'))
    return d

with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(0)
    anim.draw_frame(10)
    anim.draw_frame(20)
    anim.draw_frame(30)
While Loop

```
while condition:
    body
```
While Loop

```
while condition:
    body
```

an expression that evaluates to a boolean value (True/False)
While Loop

while condition:
    body

one or more lines of code (indented, just like a function definition)
Conditional Operators

x = 5
print x < 4
Conditional Operators

x = 5
print x < 4

output: False
Conditional Operators

x = 5
x = x - 2
print x < 4
Conditional Operators

x = 5
x = x - 2
print x < 4

output: True
Conditional Operators

c = 5 < 6
print c
Conditional Operators

c = 5 < 6
print c

output: True
Conditional Operators

<  less than
>  greater than
== equal to
>= greater than or equal to
<= less than or equal to
!= not equal to
Conditional Operators

=  !=  ==

x = 5

print x == 4
False

print x = 5
Error
While Loop

In [1]:
   t = 0
   while( t < 180 ):
       t = t + 10
       print(t)

10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
# --- DRILL ---
# rotate a square around a circular trajectory
# using a while loop

import drawSvg as draw

def draw_frame(t):
    r = 100  # radius of circular trajectory
    x = r * math.cos(t*math.pi/180)  # x-on circle
    y = r * math.sin(t*math.pi/180)  # y-on circle
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Rectangle(x, y, 25, 25, fill='red'))
    return d

# rotate the square
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    # your code here