Exposure CS 2021 for CS1

Chapter 18 Section 6-7 Slides

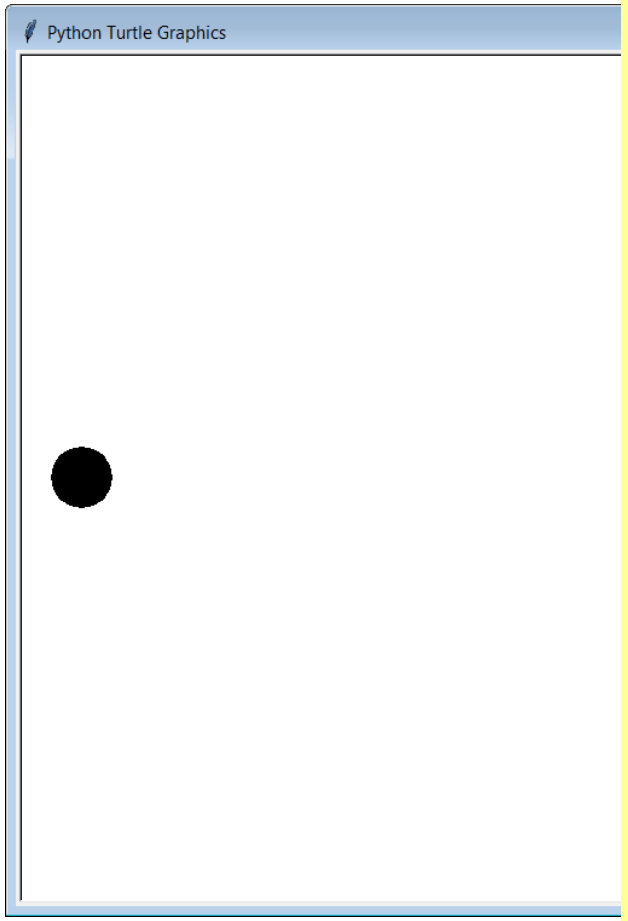
More Graphics:
Computer Animation & The Last Word

PowerPoint Presentation
created by:
Mr. John L. M. Schram
and Mr. Leon Schram
Authors of Exposure
Computer Science

Section 18.6

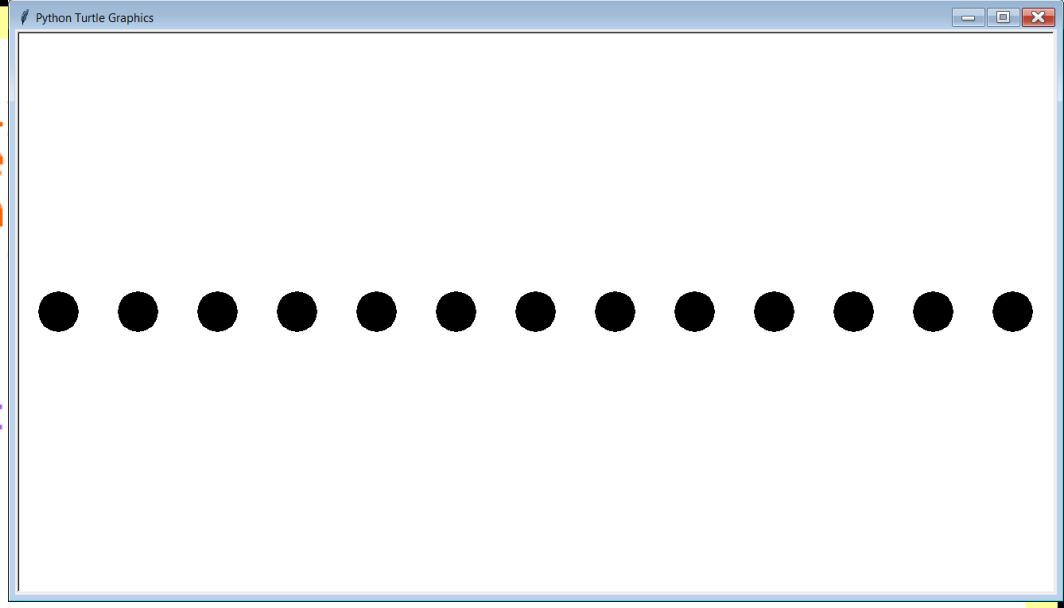
COMPUTER ANIMATION

```
1  # Animation01.py
2  # This program simply draws a solid black circle.
3  # This will eventually be "animated" as we go
4  # through the next few programs.
5
6
7  from Graphics import *
8
9
10 beginGrfx(1300,700)
11
12 fillCircle(50,350,25)
13
14 endGrfx()
15
```

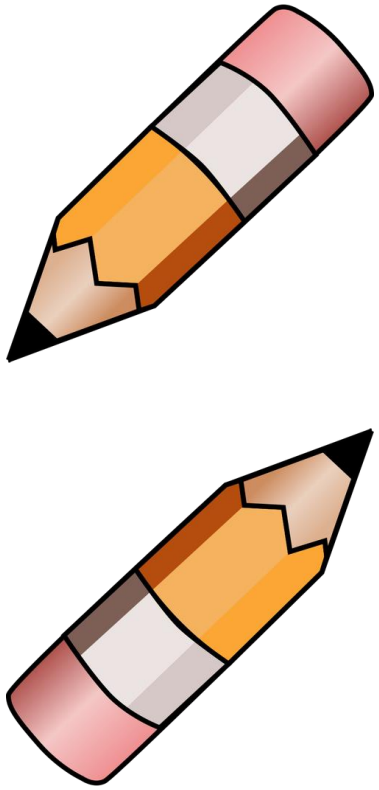


```
 1 # Animation02.py
 2 # This program simply draws the solid black
 3 # circle at 13 different locations.
 4 # This is not yet animation because we see
 5 # all 13 circles at the same time.
 6
 7
 8 from Graphics import *
 9
10
11 beginGrfx(1300,700)
12
13 fillCircle(50,350,25)
14 fillCircle(150,350,25)
15 fillCircle(250,350,25)
16 fillCircle(350,350,25)
17 fillCircle(450,350,25)
18 fillCircle(550,350,25)
19 fillCircle(650,350,25)
20 fillCircle(750,350,25)
21 fillCircle(850,350,25)
22 fillCircle(950,350,25)
23 fillCircle(1050,350,25)
24 fillCircle(1150,350,25)
25 fillCircle(1250,350,25)
26
27 endGrfx()
```

```
 1 # Animation02.py
 2 # This program simpl
 3 # circle at 13 diffe
 4 # This is not yet an
 5 # all 13 circles at
 6
 7
 8 from Graphics import
 9
10
11 beginGrfx(1300,700)
12
13 fillCircle(50,350,25)
14 fillCircle(150,350,25)
15 fillCircle(250,350,25)
16 fillCircle(350,350,25)
17 fillCircle(450,350,25)
18 fillCircle(550,350,25)
19 fillCircle(650,350,25)
20 fillCircle(750,350,25)
21 fillCircle(850,350,25)
22 fillCircle(950,350,25)
23 fillCircle(1050,350,25)
24 fillCircle(1150,350,25)
25 fillCircle(1250,350,25)
26
27 endGrfx()
```

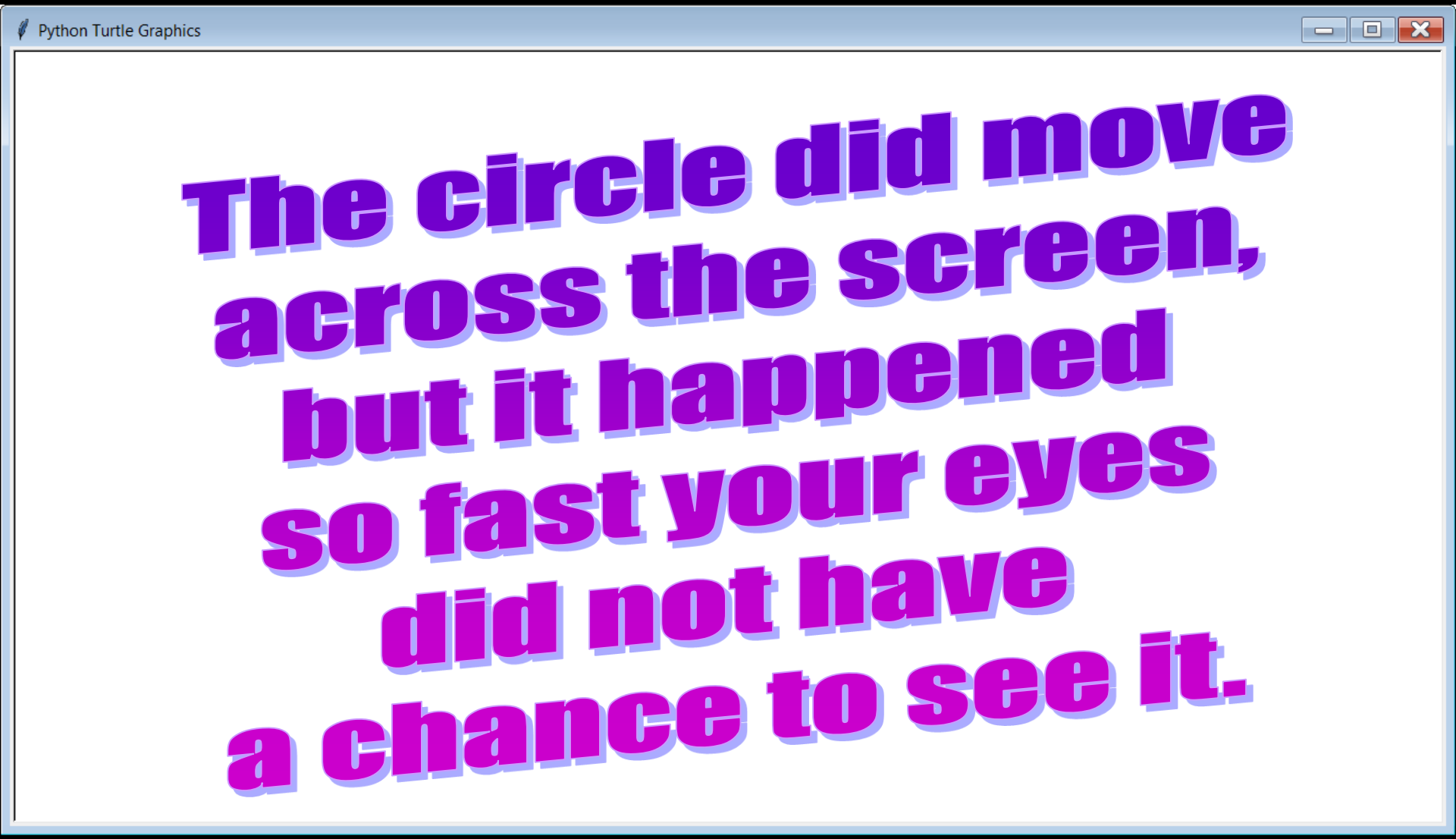


```
# Animation03.py
# This program erases every circle before it draws the
# next one.  This is necessary for animation; however,
# the process happens so fast that you cannot see the
# circle move.
# NOTE: On a Mac, you may see some circle outlines left
# behind because it did not completely erase the circles.


from Graphics import *


beginGrfx(1300,700)

setColor("black")
fillCircle(50,350,25)
setColor("white")
fillCircle(50,350,25)

setColor("black")
fillCircle(150,350,25)
setColor("white")
fillCircle(150,350,25)

setColor("black")
fillCircle(250,350,25)
setColor("white")
fillCircle(250,350,25)

setColor("black")
fillCircle(350,350,25)
setColor("white")
fillCircle(350,350,25)
```

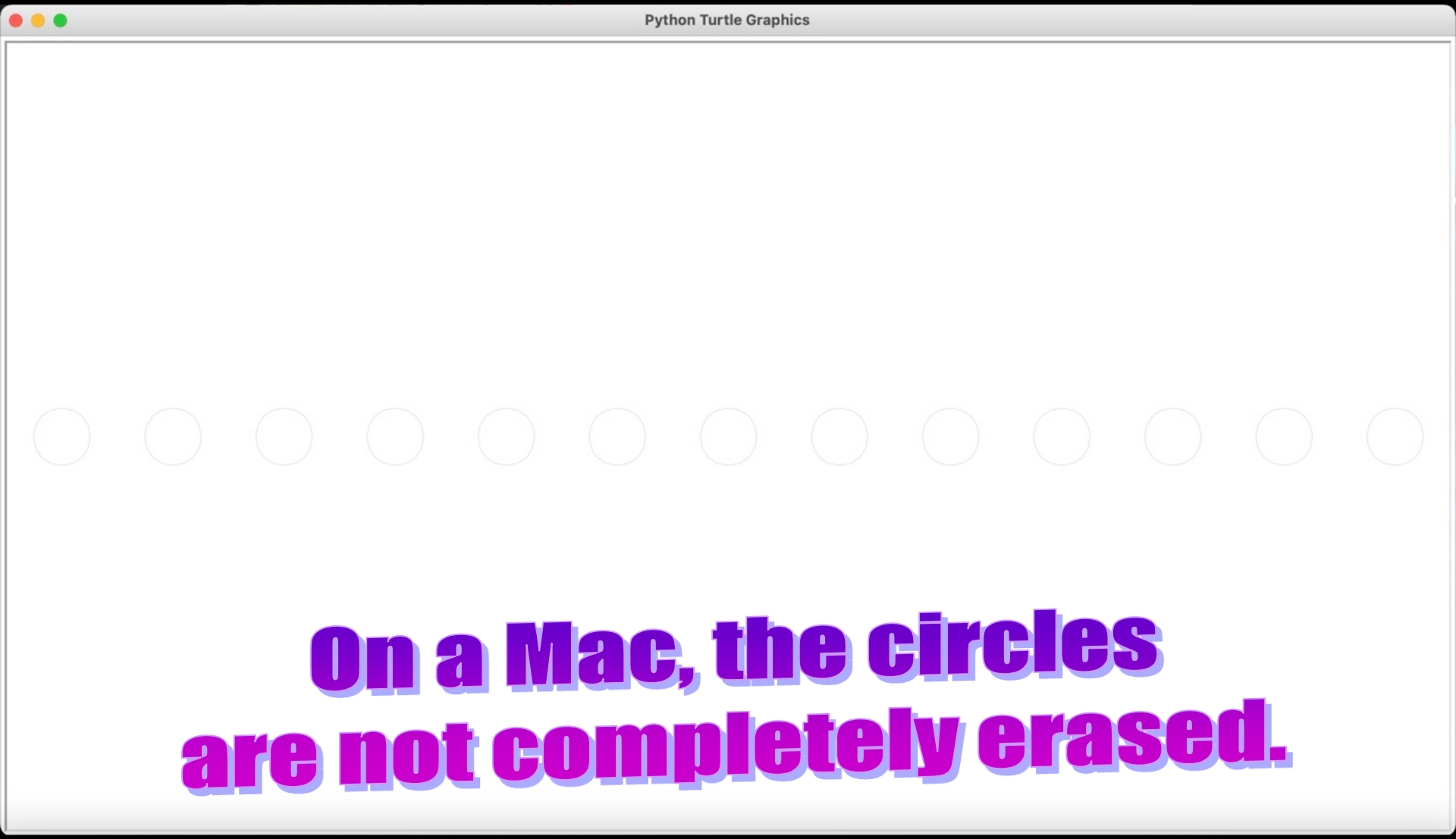


```
setColor("black")
fillCircle(450,350,25)
setColor("white")
fillCircle(450,350,25)

setColor("black")
fillCircle(550,350,25)
setColor("white")
fillCircle(550,350,25)

setColor("black")
fillCircle(650,350,25)
setColor("white")
fillCircle(650,350,25)

setColor("black")
fillCircle(750,350,25)
setColor("white")
fillCircle(750,350,25)

setColor("black")
fillCircle(850,350,25)
setColor("white")
fillCircle(850,350,25)

setColor("black")
fillCircle(950,350,25)
setColor("white")
fillCircle(950,350,25)

setColor("black")
fillCircle(1050,350,25)
setColor("white")
fillCircle(1050,350,25)

setColor("black")
fillCircle(1150,350,25)
setColor("white")
fillCircle(1150,350,25)

setColor("black")
fillCircle(1250,350,25)
setColor("white")
fillCircle(1250,350,25)

endGrfx()
```

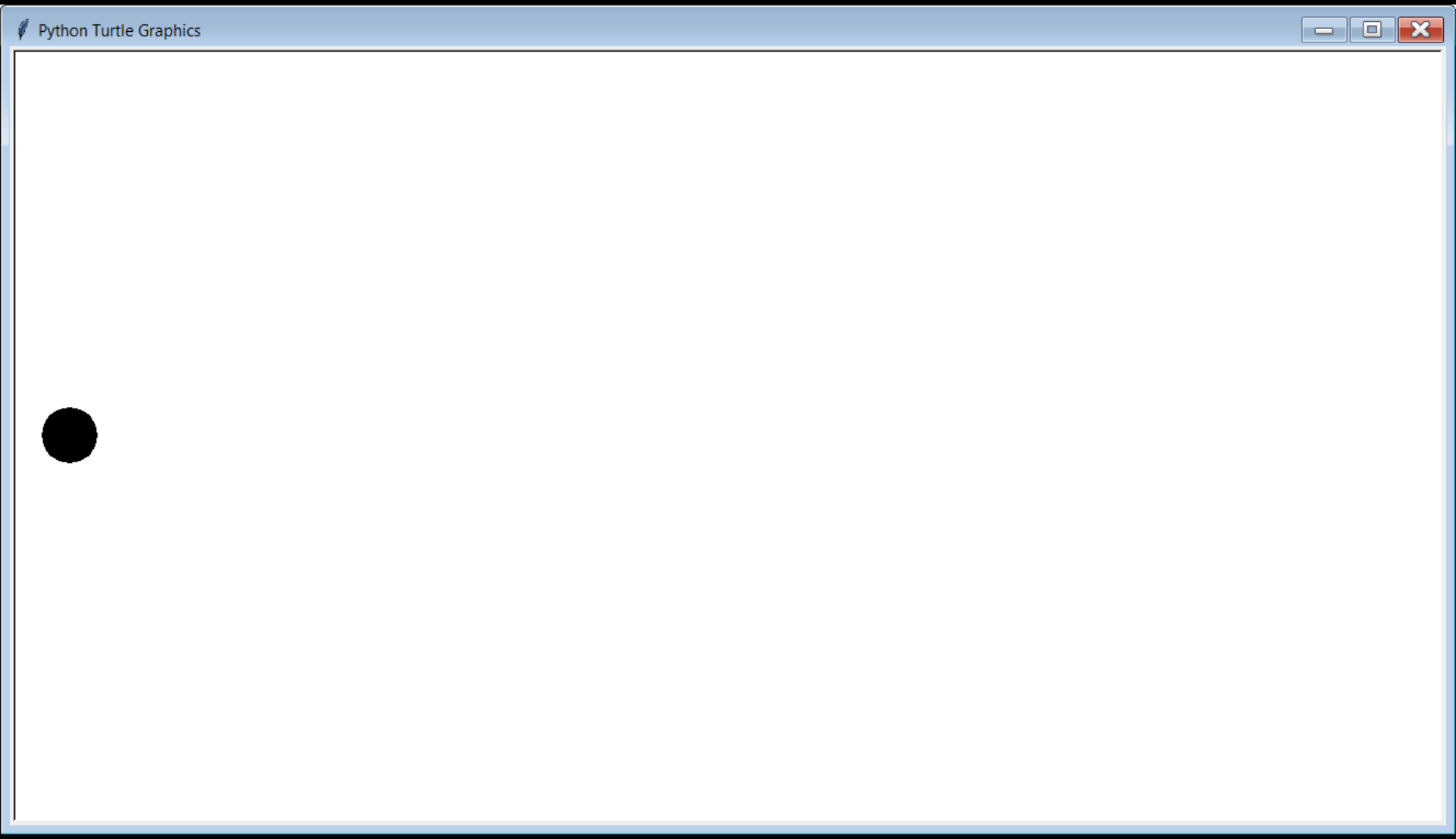
Windows PC Output:

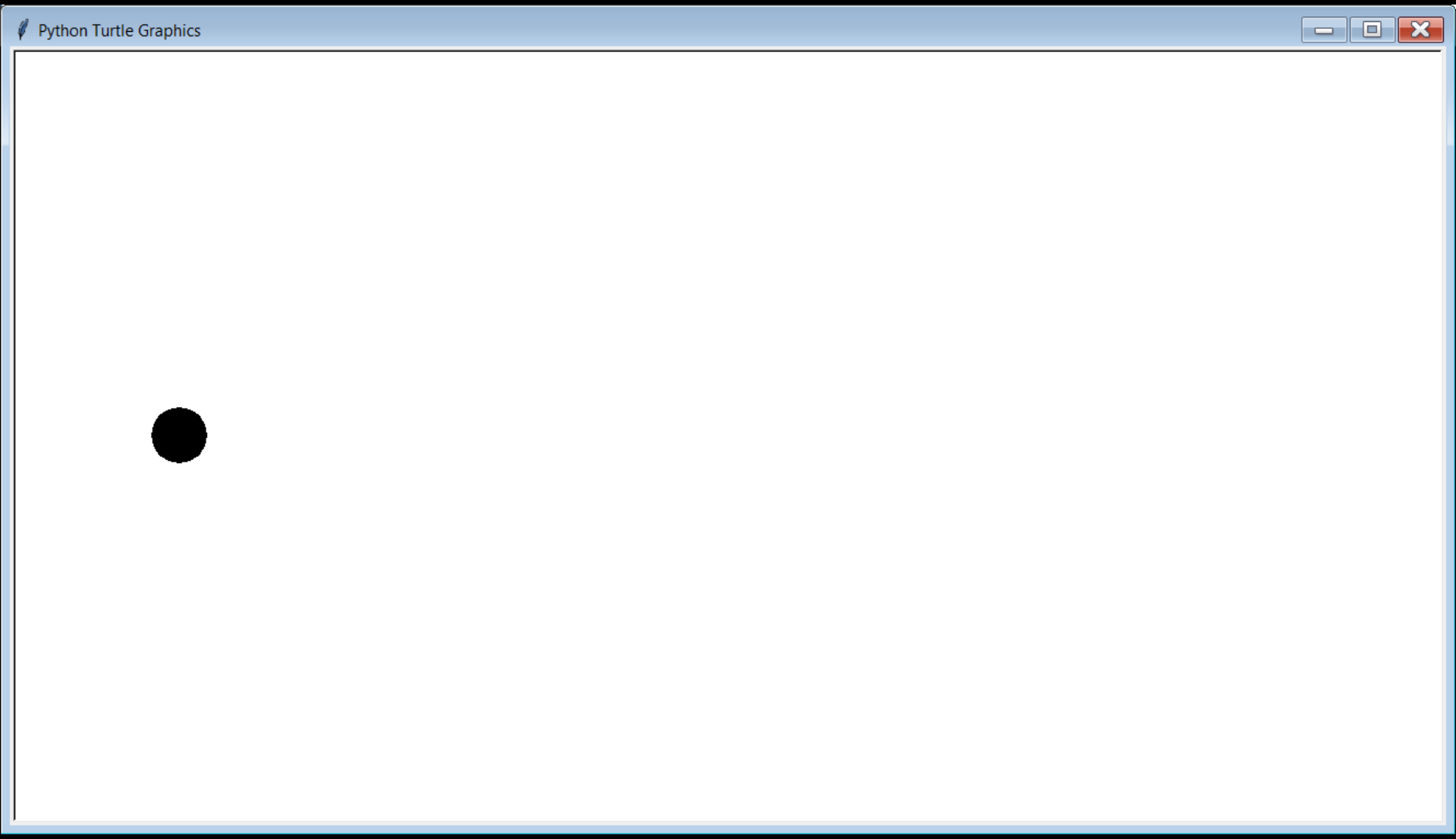


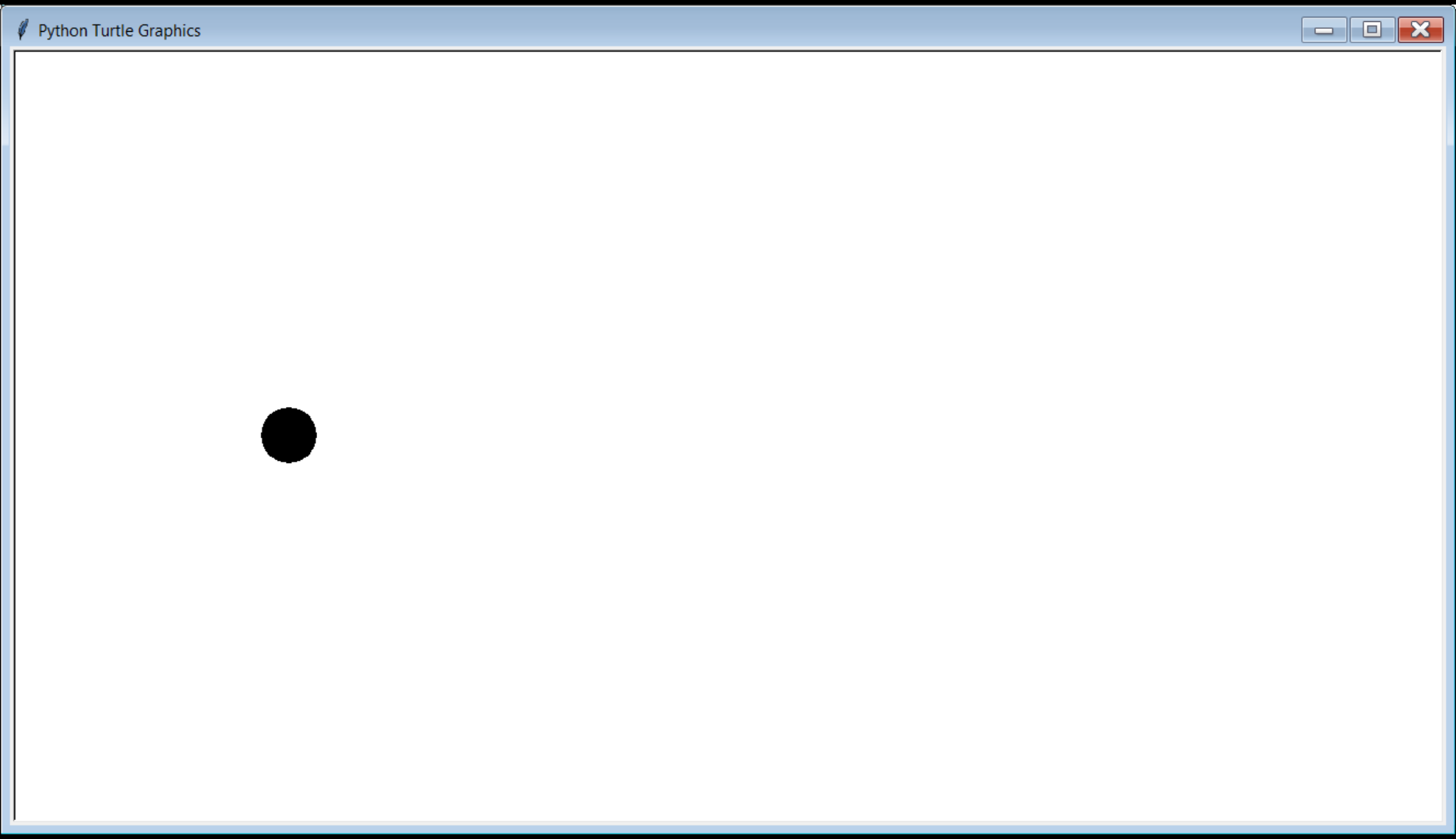
Mac Output:

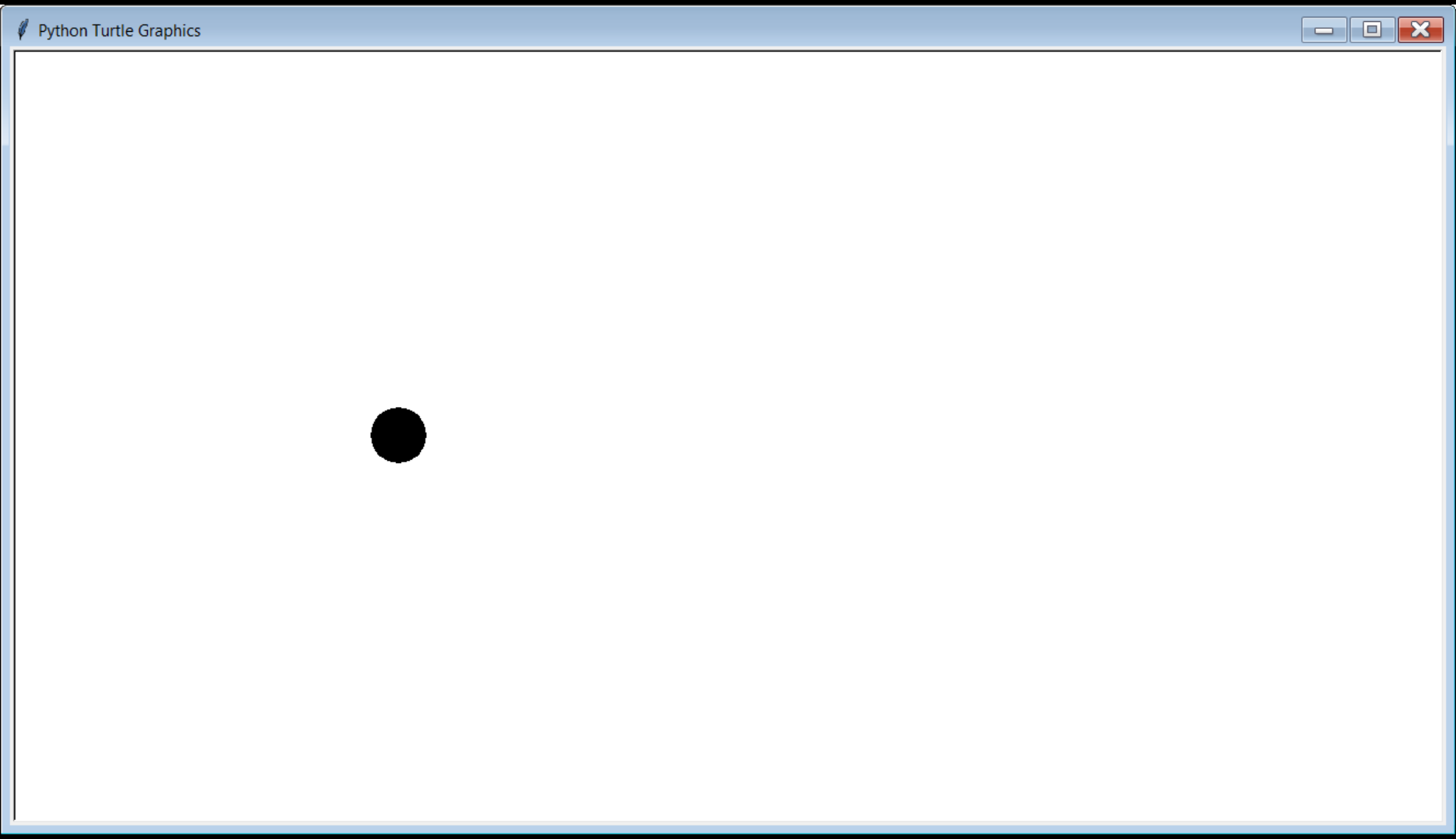

Python Turtle Graphics
On a Mac, the circles
are not completely erased.

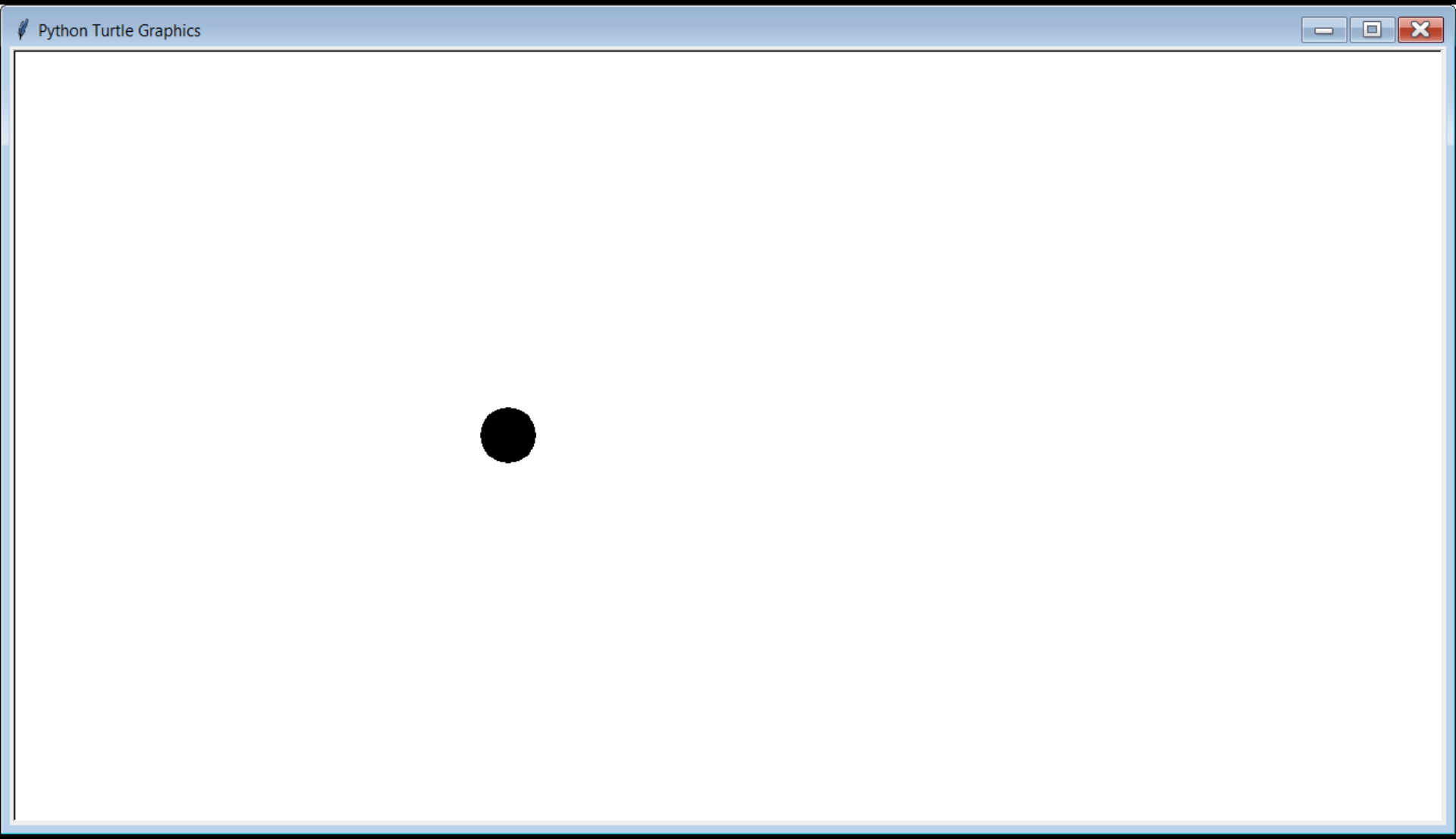
```
# Animation04.py
# This program adds a short 1 second delay
# after each circle is drawn to give you a
# chance to see it before it is erased.
# NOTE: This is called "Draw-And-Erase Animation".
# ALSO: The issue of the circle outlines on a Mac
#       was fixed by making the radii of the
#       erasing circles a little bigger.


from Graphics import *


beginGrfx(1300,700)

setColor("black")
fillCircle(50,350,25)
delay(1000)
setColor("white")
fillCircle(50,350,27)

setColor("black")
fillCircle(150,350,25)
delay(1000)
setColor("white")
fillCircle(150,350,27)

setColor("black")
fillCircle(250,350,25)
delay(1000)
setColor("white")
fillCircle(250,350,27)

setColor("black")
fillCircle(350,350,25)
delay(1000)
setColor("white")
fillCircle(350,350,27)

setColor("black")
fillCircle(450,350,25)
delay(1000)
setColor("white")
fillCircle(450,350,27)

setColor("black")
fillCircle(550,350,25)
delay(1000)
setColor("white")
fillCircle(550,350,27)

setColor("black")
fillCircle(650,350,25)
delay(1000)
setColor("white")
fillCircle(650,350,27)

setColor("black")
fillCircle(750,350,25)
delay(1000)
setColor("white")
fillCircle(750,350,27)

setColor("black")
fillCircle(850,350,25)
delay(1000)
setColor("white")
fillCircle(850,350,27)

setColor("black")
fillCircle(950,350,25)
delay(1000)
setColor("white")
fillCircle(950,350,27)

setColor("black")
fillCircle(1050,350,25)
delay(1000)
setColor("white")
fillCircle(1050,350,27)

setColor("black")
fillCircle(1150,350,25)
delay(1000)
setColor("white")
fillCircle(1150,350,27)

setColor("black")
fillCircle(1250,350,25)
delay(1000)
setColor("white")
fillCircle(1250,350,27)

endGrfx()
```

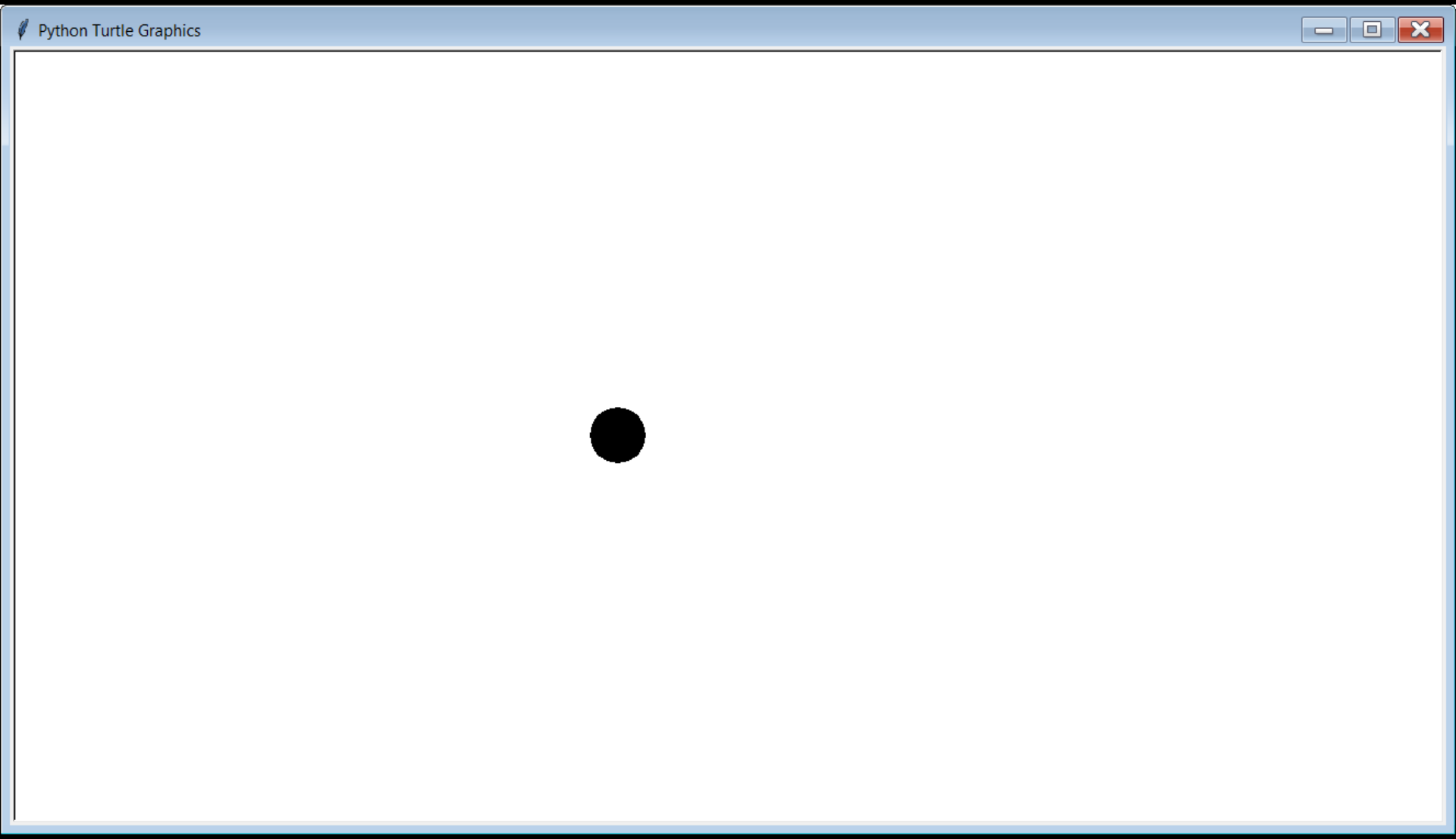

Python Turtle Graphics

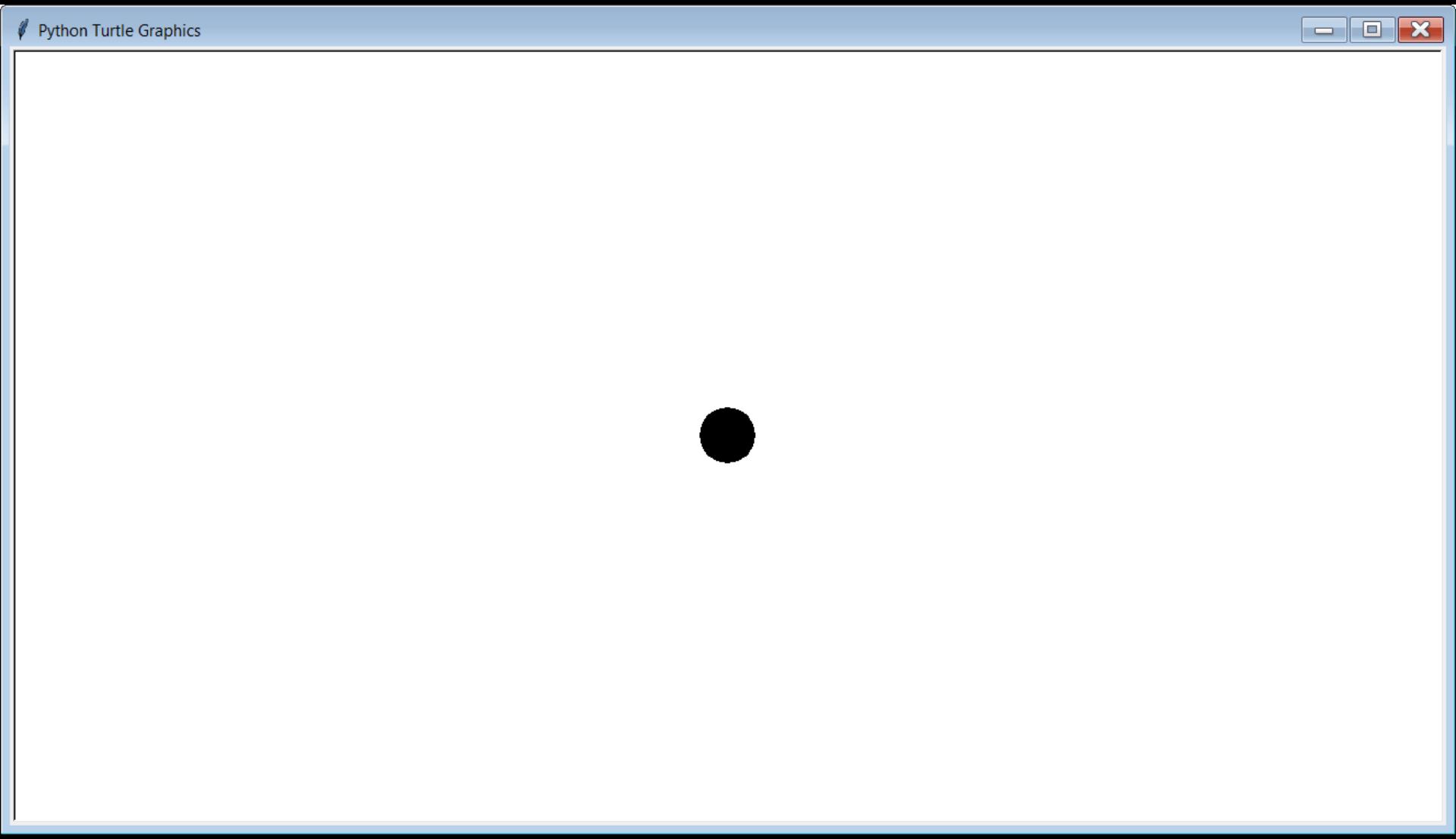

Python Turtle Graphics

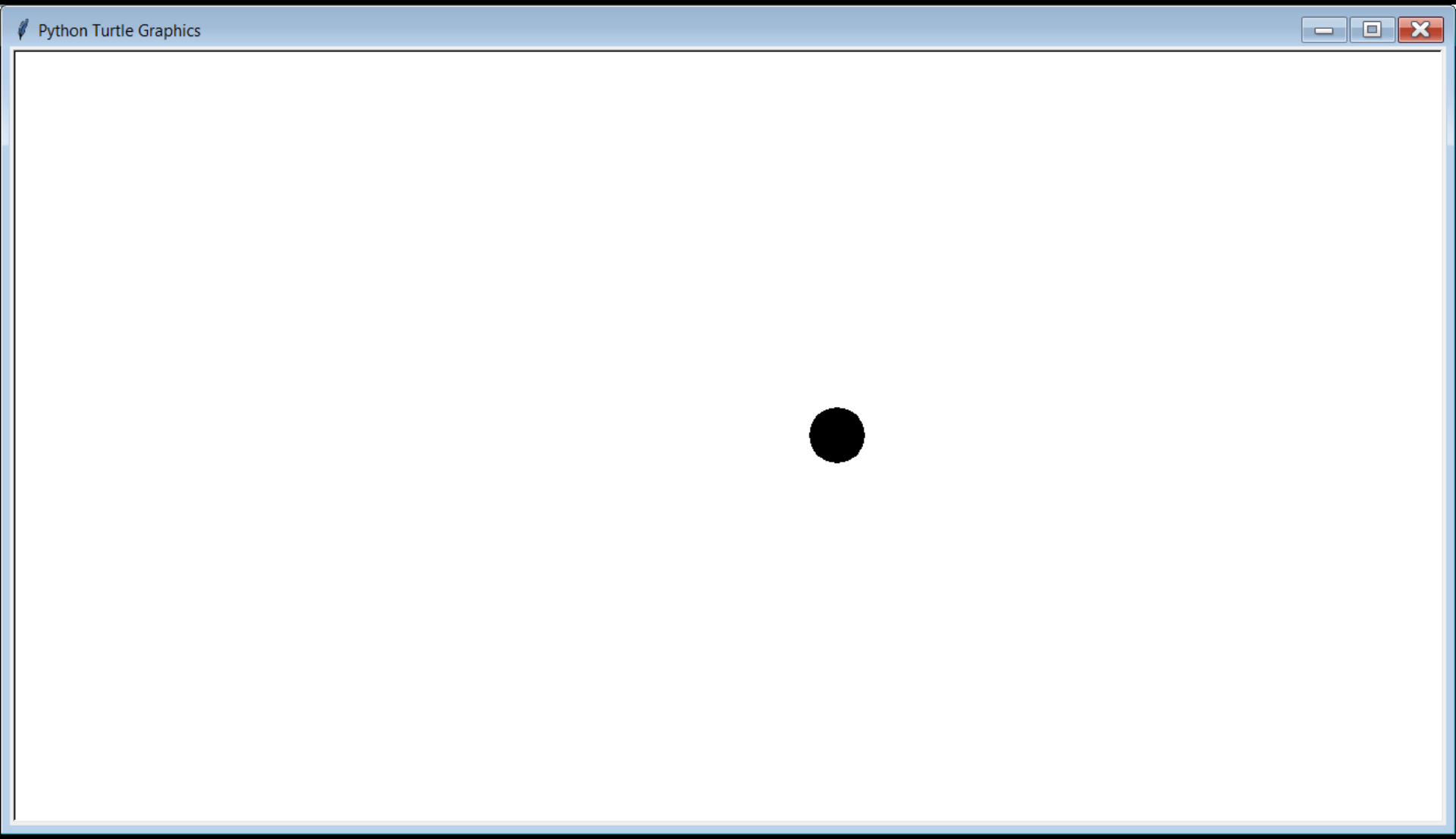

Python Turtle Graphics

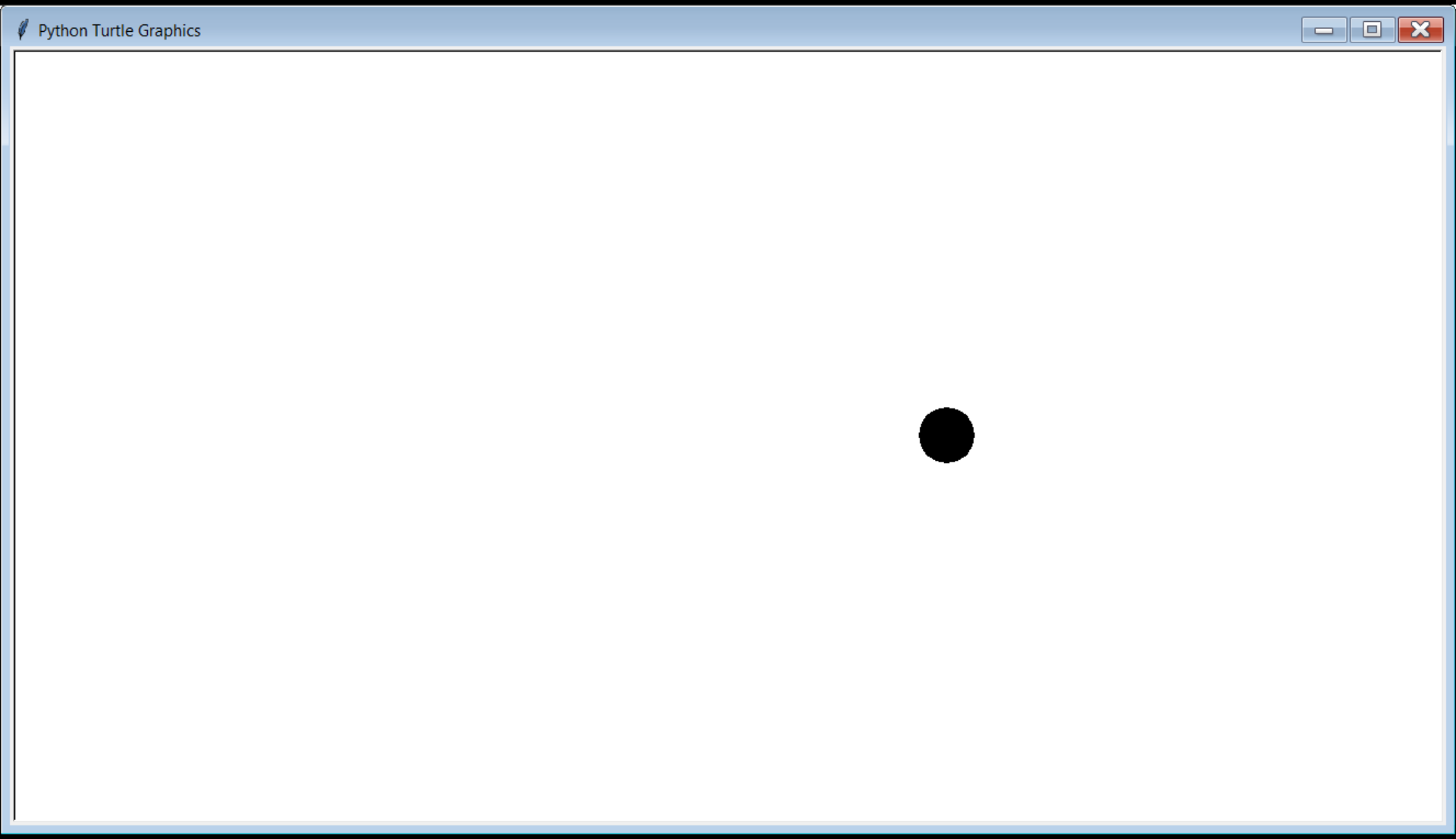

Python Turtle Graphics

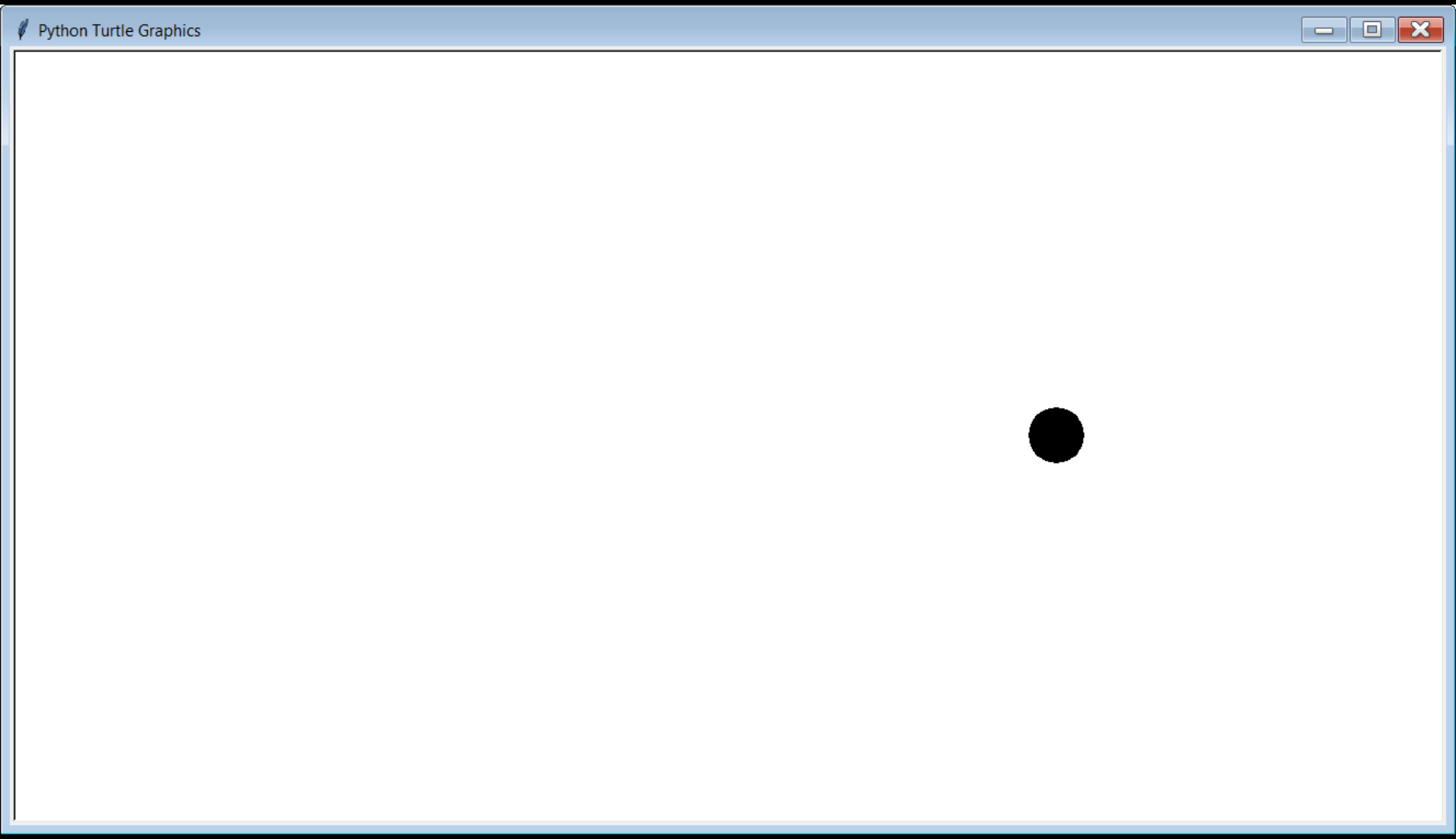

Python Turtle Graphics


Python Turtle Graphics

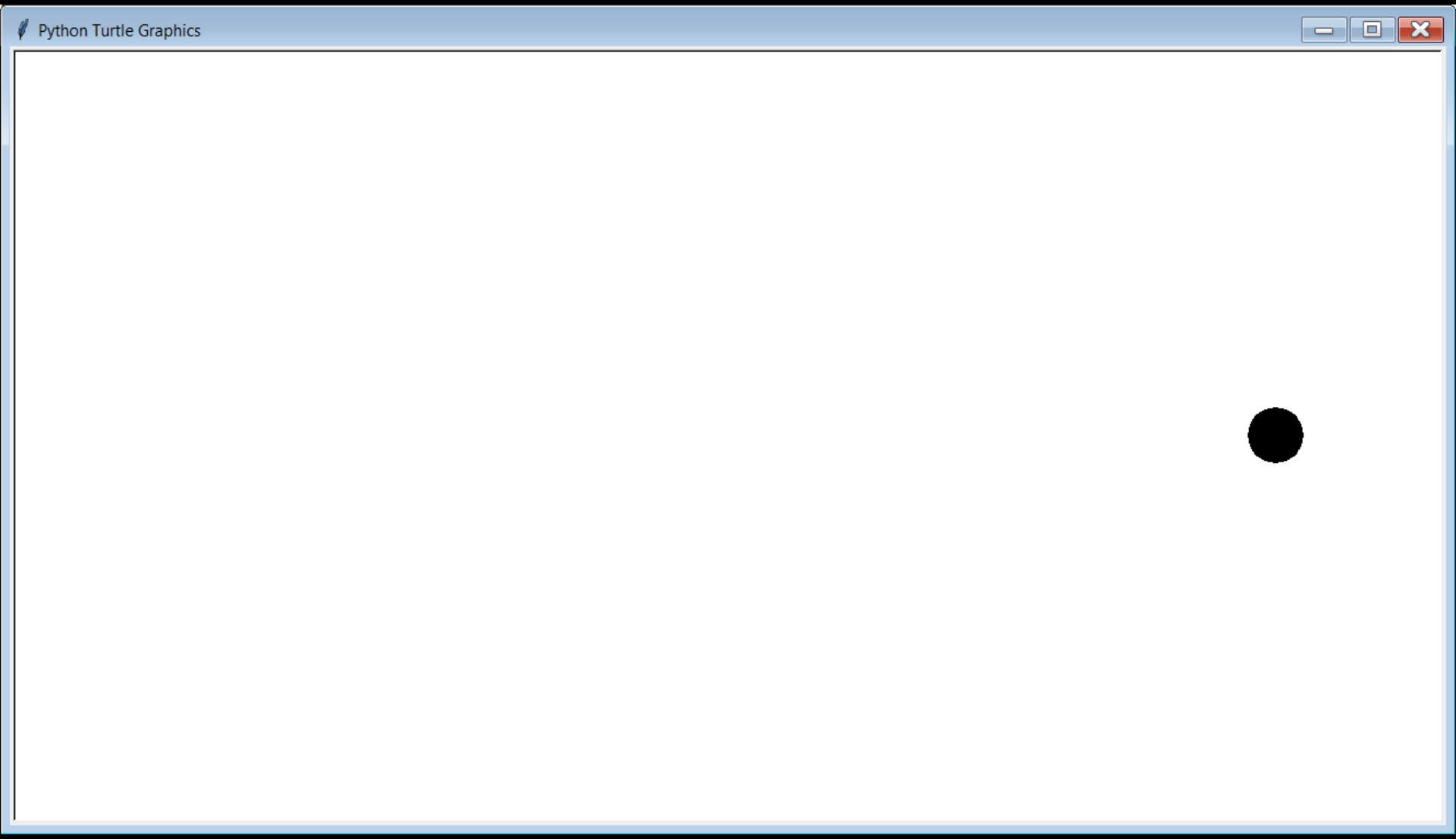

Python Turtle Graphics

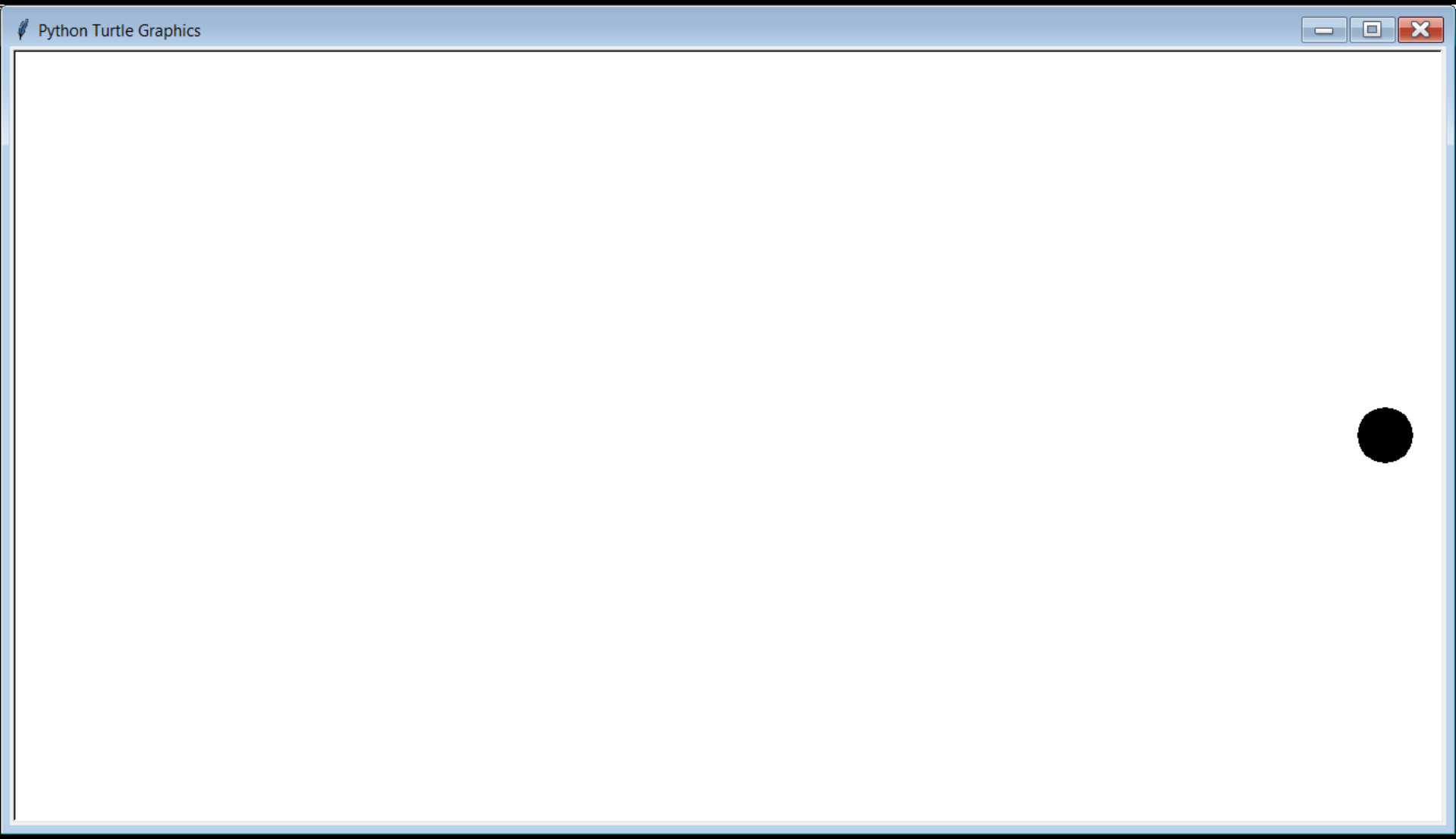

Python Turtle Graphics

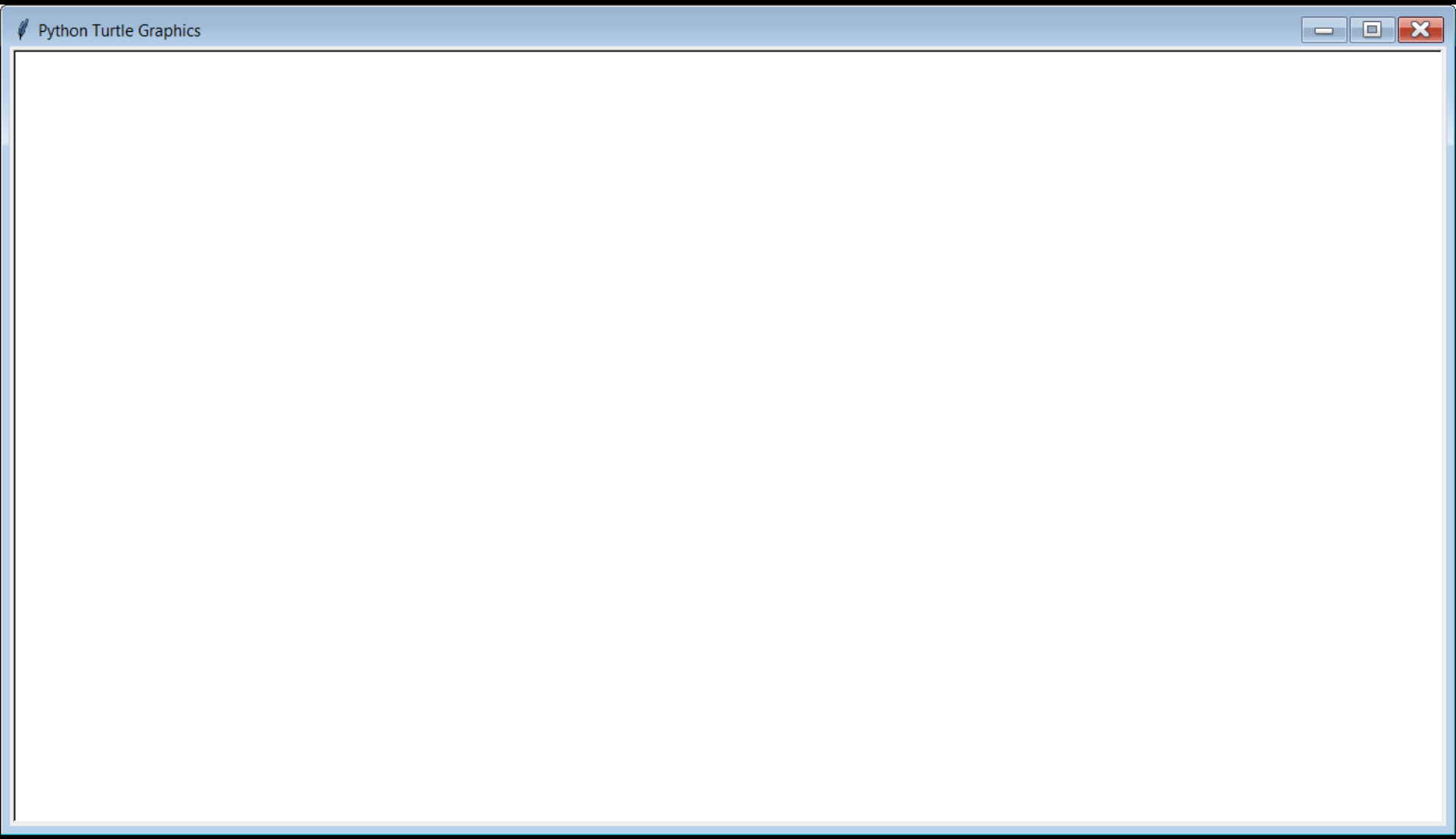

Python Turtle Graphics


Python Turtle Graphics


Python Turtle Graphics


Python Turtle Graphics


Python Turtle Graphics


Python Turtle Graphics

Program Note

The output of program Animation04.py was simulated with PowerPoint.

It will not be possible to simulate the output of the remaining programs.

You need to execute them on your own computer to truly see what they are doing.

Do you see a pattern here?

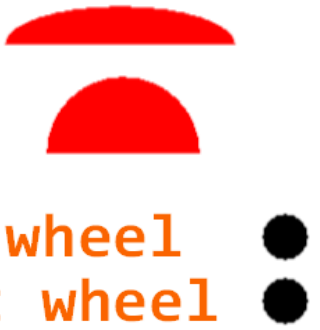
```
# Animation04.py
#
#
#
#
#
#
#

from Graphics import *


beginGrfx(1300,700)

setColor("black")
fillCircle(50,350,25)
delay(1000)
setColor("white")
fillCircle(50,350,27)

setColor("black")
fillCircle(150,350,25)
delay(1000)
setColor("white")
fillCircle(150,350,27)

setColor("black")
fillCircle(250,350,25)
delay(1000)
setColor("white")
fillCircle(250,350,27)

setColor("black")
fillCircle(350,350,25)
delay(1000)
setColor("white")
fillCircle(350,350,27)

setColor("black")
fillCircle(450,350,25)
delay(1000)
setColor("white")
fillCircle(450,350,27)

setColor("black")
fillCircle(550,350,25)
delay(1000)
setColor("white")
fillCircle(550,350,27)

setColor("black")
fillCircle(650,350,25)
delay(1000)
setColor("white")
fillCircle(650,350,27)

setColor("black")
fillCircle(750,350,25)
delay(1000)
setColor("white")
fillCircle(750,350,27)

setColor("black")
fillCircle(850,350,25)
delay(1000)
setColor("white")
fillCircle(850,350,27)

setColor("black")
fillCircle(950,350,25)
delay(1000)
setColor("white")
fillCircle(950,350,27)

setColor("black")
fillCircle(1050,350,25)
delay(1000)
setColor("white")
fillCircle(1050,350,27)

setColor("black")
fillCircle(1150,350,25)
delay(1000)
setColor("white")
fillCircle(1150,350,27)

setColor("black")
fillCircle(1250,350,25)
delay(1000)
setColor("white")
fillCircle(1250,350,27)

endGrfx()
```

```
1  # Animation05.py
2  # This program has the same output as the
3  # previous program.  The program is now much
4  # shorter because it uses a <for> loop.
5
6
7  from Graphics import *
8
9
10 beginGrfx(1300,700)
11
12 for x in range(50,1251,100):
13     setColor("black")
14     fillCircle(x,350,25)
15     delay(1000)
16     setColor("white")
17     fillCircle(x,350,27)
18
19 endGrfx()
```

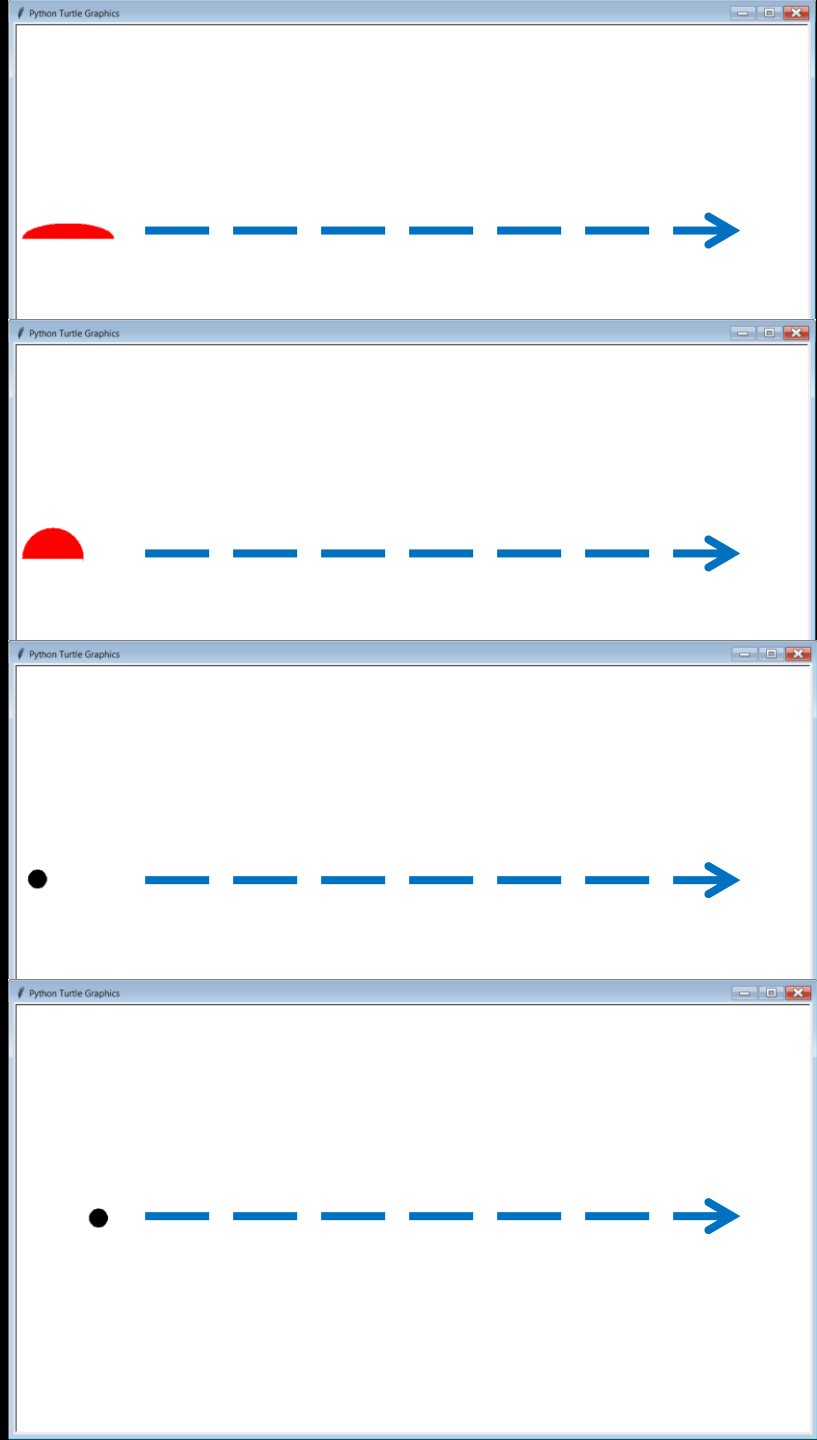
```
 1 # Animation06.py
 2 # This program makes the animation smoother
 3 # by using a smaller delay and a smaller
 4 # increment in the <for> loop.
 5
 6
 7 from Graphics import *
 8
 9
10 beginGrfx(1300,700)
11
12 for x in range(50,1251,10):
13     setColor("black")
14     fillCircle(x,350,25)
15     delay(100)
16     setColor("white")
17     fillCircle(x,350,27)
18
19 endGrfx()
```

```
 1 # Animation07.py
 2 # This program makes the animation as smooth
 3 # as possible by having an increment of just
 4 # 1 pixel in the <for> loop.
 5 # The delay is also made smaller.
 6
 7
 8 from Graphics import *
 9
10
11 beginGrfx(1300,700)
12
13 for x in range(50,1251,1):
14     setColor("black")
15     fillCircle(x,350,25)
16     delay(10)
17     setColor("white")
18     fillCircle(x,350,27)
19
20 endGrfx()
```

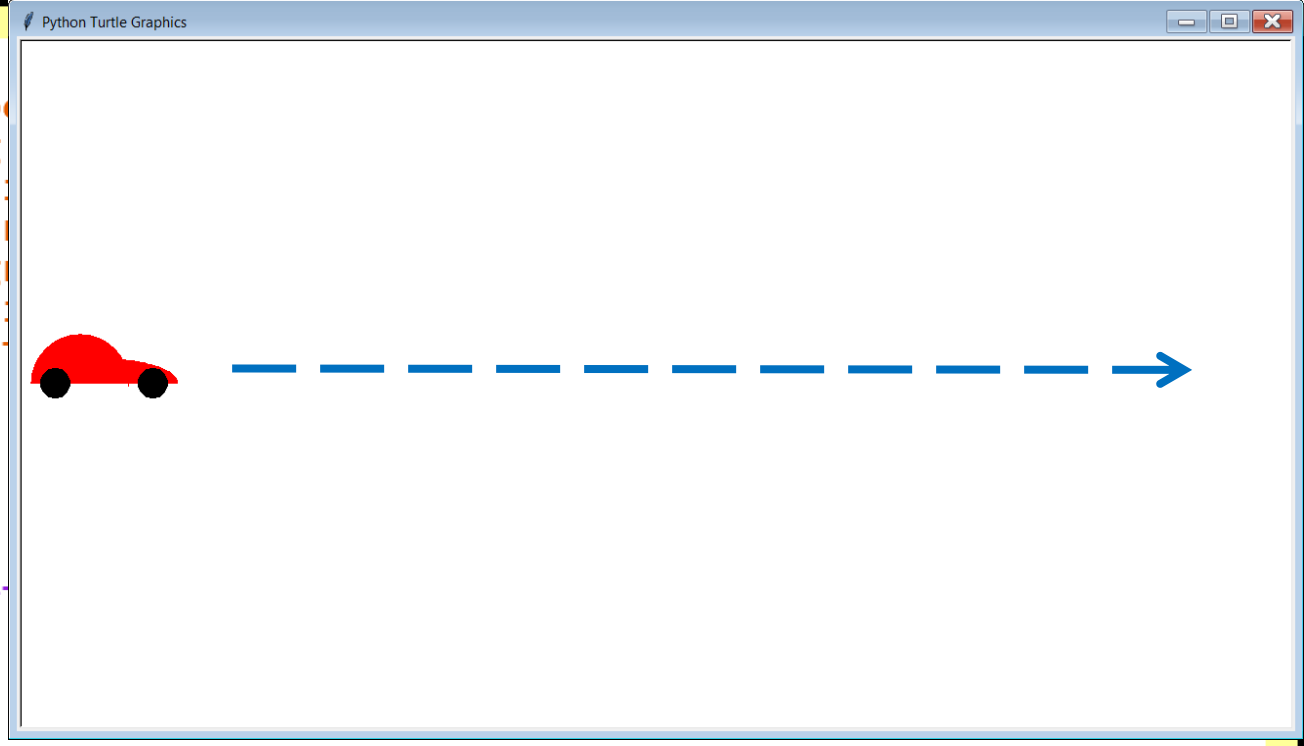
```
 1 # Animation08.py
 2 # This program draws the car that will
 3 # be animated in the next few programs.
 4 # The car is an overlapping composition
 5 # of 4 filled shapes, specifically the
 6 # 2 arcs and 2 circles.
 7
 8
 9 from Graphics import *
10
11
12 beginGrfx(1300,700)
13
14 setColor("red")
15 fillArc(85,350,75,25,270,90)   # body
16 fillArc(60,350,50,50,270,90)   # hood
17 setColor("black")
18 fillCircle(35,350,15)          # rear wheel
19 fillCircle(135,350,15)         # front wheel
20
21 endGrfx()
```



```
1  # Animation09.py
2  # This program demonstrates a common logic error
3  # when students try to move complicated images
4  # across the screen. The problem is each part of
5  # the car is in its own separate loop.  This means
6  # each part of the car moves across the screen one
7  # at a time instead of all together.  On top of this,
8  # on a Mac each piece may leave behind a trail.
9
10
11 from Graphics import *
12
13
14 beginGrfx(1300,700)
15
16 for x in range(85,1226,10):          # body
17    setColor("red")
18    fillArc(x,350,75,25,270,90)
19    delay(50)
20    setColor("white")
21    fillArc(x,350,75,25,270,90)
22
23 for x in range(60,1201,10):          # hood
24    setColor("red")
25    fillArc(x,350,50,50,270,90)
26    delay(50)
27    setColor("white")
28    fillArc(x,350,50,50,270,90)
29
30 for x in range(35,1176,10):          # rear wheel
31    setColor("black")
32    fillCircle(x,350,15)
33    delay(50)
34    setColor("white")
35    fillCircle(x,350,15)
36
37 for x in range(135,1276,10):        # front wheel
38    setColor("black")
39    fillCircle(x,350,15)
40    delay(50)
41    setColor("white")
42    fillCircle(x,350,15)
43
44 endGrfx()
```



```
 1 # Animation10.py
 2 # This program properly animates the car across the
 3 # screen by placing all 4 shapes in the same <for> loop.
 4 # A little algebra is required since the location of 3
 5 # of the shapes is relative to the location of the rear
 6 # wheel.  This program also introduces another issue.
 7 # As more and more images are drawn/erased on/from the
 8 # screen, the output gets slower and slower and eventually
 9 # drags to a crawl.  And on top of all of this, on a Mac
10 # the car may leave behind a trail.
11
12
13
14 from Graphics import *
15
16
17 beginGrfx(1300,700)
18
19 for x in range(35,1176,5):
20     setColor("red")
21     fillArc(x+50,350,75,25,270,90) # draw body
22     fillArc(x+25,350,50,50,270,90) # draw hood
23     setColor("black")
24     fillCircle(x,350,15)           # draw rear wheel
25     fillCircle(x+100,350,15)       # draw front wheel
26     delay(10)
27     setColor("white")
28     fillArc(x+50,350,75,25,270,90) # erase body
29     fillArc(x+25,350,50,50,270,90) # erase hood
30     fillCircle(x,350,15)           # erase rear wheel
31     fillCircle(x+100,350,15)       # erase front wheel
32
33 endGrfx()
```



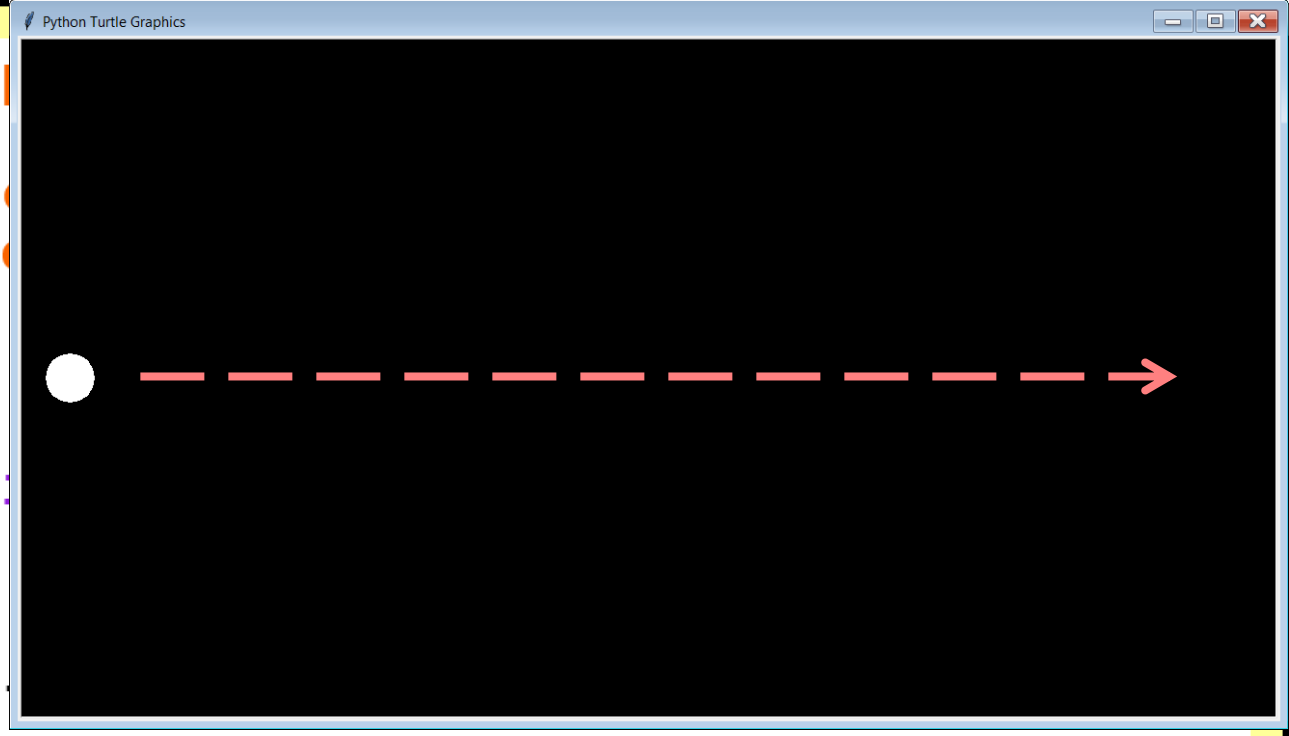

```
# Animation10.py
# This program prop
# screen by placing
# A little algebra
# of the shapes is
# wheel.  This prog
# As more and more
# screen, the outpu
# drags to a crawl.
# the car may leave



from Graphics impor


beginGrfx(1300,700)

for x in range(35,1176,5):
    setColor("red")
    fillArc(x+50,350,75,25,270,90) # draw body
    fillArc(x+25,350,50,50,270,90) # draw hood
    setColor("black")
    fillCircle(x,350,15)           # draw rear wheel
    fillCircle(x+100,350,15)       # draw front wheel
    delay(10)
    setColor("white")
    fillArc(x+50,350,75,25,270,90) # erase body
    fillArc(x+25,350,50,50,270,90) # erase hood
    fillCircle(x,350,15)           # erase rear wheel
    fillCircle(x+100,350,15)       # erase front wheel

endGrfx()
```

```
 1 # Animation11.py
 2 # This program fixes all of the issues of the
 3 # previous program by using the <clear> command
 4 # to completely erase everything on the screen.
 5
 6
 7 from Graphics import *
 8
 9
10 beginGrfx(1300,700)
11
12 for x in range(35,1176,5):
13     setColor("red")
14     fillArc(x+50,350,75,25,270,90) # draw body
15     fillArc(x+25,350,50,50,270,90) # draw hood
16     setColor("black")
17     fillCircle(x,350,15)           # draw rear wheel
18     fillCircle(x+100,350,15)       # draw front wheel
19     delay(10)
20     clear()                        # erase everything
21
22 endGrfx()
```

```
# Animation12.py
# This program shows an alternate way to
# animate the car across the screen.
# While this way does not require Algebra
# and is less complicated, the code is much
# longer, and the output is not as smooth.


from Graphics import *


beginGrfx(1300,700)

setColor("red")
fillArc(85,350,75,25,270,90)
fillArc(60,350,50,50,270,90)
setColor("black")
fillCircle(35,350,15)
fillCircle(135,350,15)
delay(1000)
clear()

setColor("red")
fillArc(185,350,75,25,270,90)
fillArc(160,350,50,50,270,90)
setColor("black")
fillCircle(135,350,15)
fillCircle(235,350,15)
delay(1000)
clear()

setColor("red")
fillArc(285,350,75,25,270,90)
fillArc(260,350,50,50,270,90)
setColor("black")
fillCircle(235,350,15)
fillCircle(335,350,15)
delay(1000)
clear()

setColor("red")
fillArc(385,350,75,25,270,90)
fillArc(360,350,50,50,270,90)
setColor("black")
fillCircle(335,350,15)
fillCircle(435,350,15)
delay(1000)
clear()

setColor("red")
fillArc(485,350,75,25,270,90)
fillArc(460,350,50,50,270,90)
setColor("black")
fillCircle(435,350,15)
fillCircle(535,350,15)
delay(1000)
clear()

setColor("red")
fillArc(585,350,75,25,270,90)
fillArc(560,350,50,50,270,90)
setColor("black")
fillCircle(535,350,15)
fillCircle(635,350,15)
delay(1000)
clear()

setColor("red")
fillArc(685,350,75,25,270,90)
fillArc(660,350,50,50,270,90)
setColor("black")
fillCircle(635,350,15)
fillCircle(735,350,15)
delay(1000)
clear()

setColor("red")
fillArc(785,350,75,25,270,90)
fillArc(760,350,50,50,270,90)
setColor("black")
fillCircle(735,350,15)
fillCircle(835,350,15)
delay(1000)
clear()

setColor("red")
fillArc(885,350,75,25,270,90)
fillArc(860,350,50,50,270,90)
setColor("black")
fillCircle(835,350,15)
fillCircle(935,350,15)
delay(1000)
clear()

setColor("red")
fillArc(985,350,75,25,270,90)
fillArc(960,350,50,50,270,90)
setColor("black")
fillCircle(935,350,15)
fillCircle(1035,350,15)
delay(1000)
clear()

setColor("red")
fillArc(1085,350,75,25,270,90)
fillArc(1060,350,50,50,270,90)
setColor("black")
fillCircle(1035,350,15)
fillCircle(1135,350,15)
delay(1000)
clear()

setColor("red")
fillArc(1185,350,75,25,270,90)
fillArc(1160,350,50,50,270,90)
setColor("black")
fillCircle(1135,350,15)
fillCircle(1235,350,15)
delay(1000)
clear()

endGrfx()
```

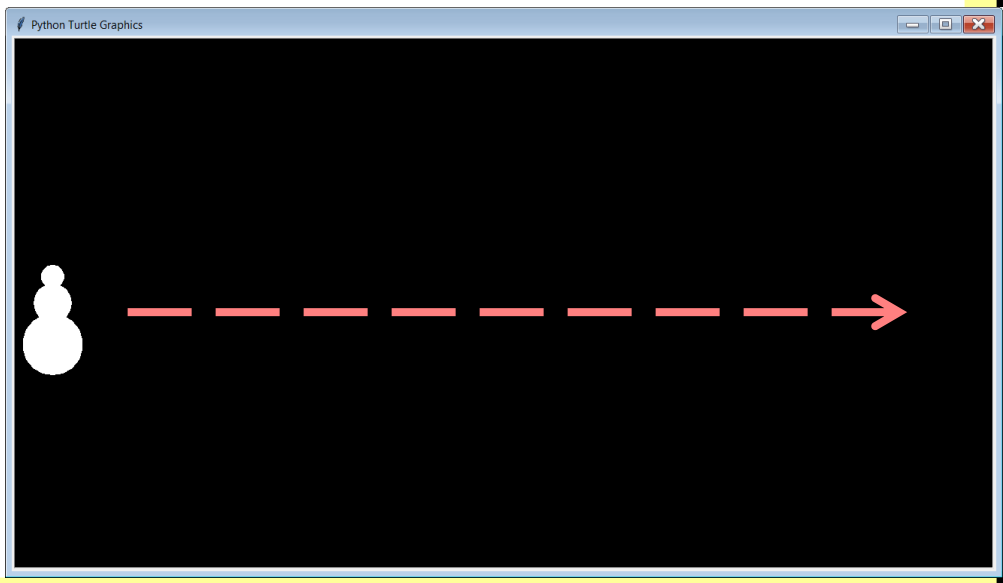
```
 1 # Animation13.py
 2 # This program is similar to Animation07.py,
 3 # but now the circle is white and the
 4 # background color is black.  This means
 5 # <"black"> is now the "erasing color".
 6
 7
 8 from Graphics import *
 9
10
11 beginGrfx(1300,700)
12
13 setBackground("black")
14
15 for x in range(50,1251,1):
16     setColor("white")
17     fillCircle(x,350,25)
18     delay(10)
19     setColor("black")
20     fillCircle(x,350,27)
21
22 endGrfx()
```

```
1  # Animation13.p
2  # This program
3  # but now the
4  # background c
5  # <"black"> is
6
7
8  from Graphics
9
10
11 beginGrfx(1300,
12
13 setBackground("black")
14
15 for x in range(50,1251,1):
16     setColor("white")
17     fillCircle(x,350,25)
18     delay(10)
19     setColor("black")
20     fillCircle(x,350,27)
21
22 endGrfx()
```



```
 1 # Animation14.py
 2 # This program animates a snowman across the screen
 3 # by placing 3 circles in the same <for> loop.
 4 # Note how the "slowing down" problem returns.
 5
 6
 7 from Graphics import *
 8
 9
10 beginGrfx(1300,700)
11
12 setBackground("black")
13
14 for x in range(50,1251,1):
15     setColor("white")
16     fillCircle(x,315,15)
17     fillCircle(x,350,25)
18     fillCircle(x,405,40)
19     delay(10)
20     setColor("black")
21     fillCircle(x,315,17)
22     fillCircle(x,350,27)
23     fillCircle(x,405,42)
24
25 endGrfx()
```

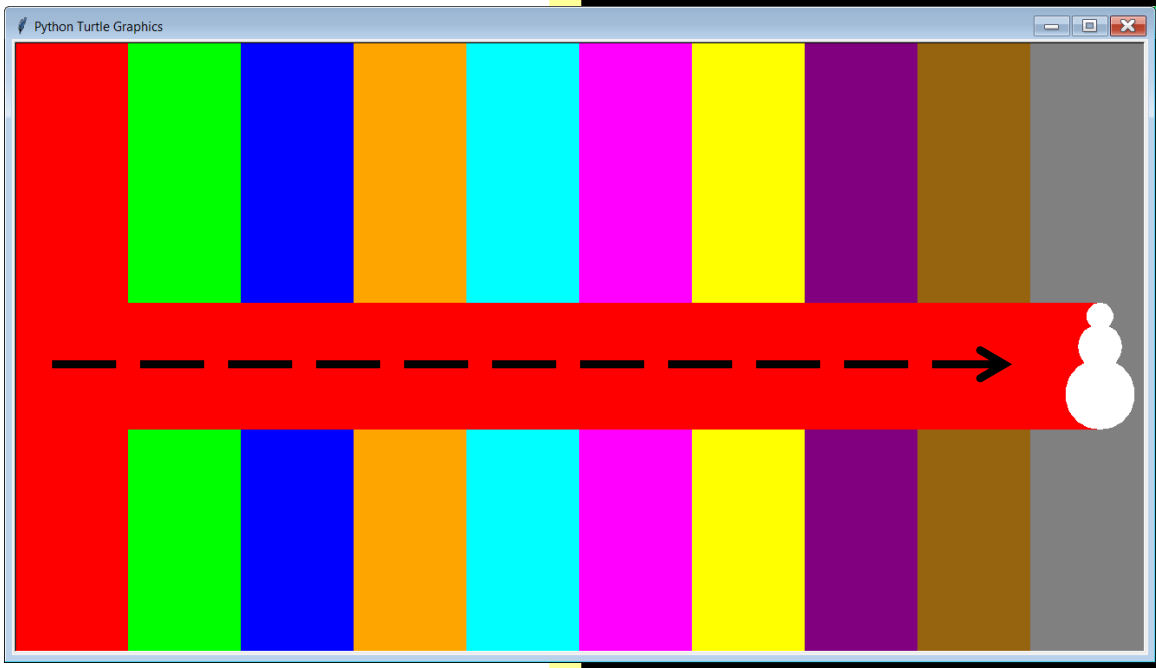
```
 1 # Animation14.py
 2 # This program animates a snowman across the screen
 3 # by placing 3 circles in the same <for> loop.
 4 # Note how the "slowing down" problem returns.
 5
 6
 7 from Graphics import *
 8
 9
10 beginGrfx(1300,700)
11
12 setBackground("black")
13
14 for x in range(50,1251,1):
15     setColor("white")
16     fillCircle(x,315,15)
17     fillCircle(x,350,25)
18     fillCircle(x,405,40)
19     delay(10)
20     setColor("black")
21     fillCircle(x,315,17)
22     fillCircle(x,350,27)
23     fillCircle(x,405,42)
24
25 endGrfx()
```



```
 1 # Animation15.py
 2 # This program fixes the "slowing down" problem
 3 # of the previous program by using <clear>.
 4 # This does require that the background color
 5 # is set immediately after the <clear> command.
 6 # NOTE: This is called "Draw-And-Redraw Animation".
 7
 8
 9 from Graphics import *
10
11
12 beginGrfx(1300,700)
13
14 for x in range(50,51,3):
15     clear()
16     setBackground("black")
17     setColor("white")
18     fillCircle(x,315,15)
19     fillCircle(x,350,25)
20     fillCircle(x,405,40)
21     delay(10)
22
23 endGrfx()
```

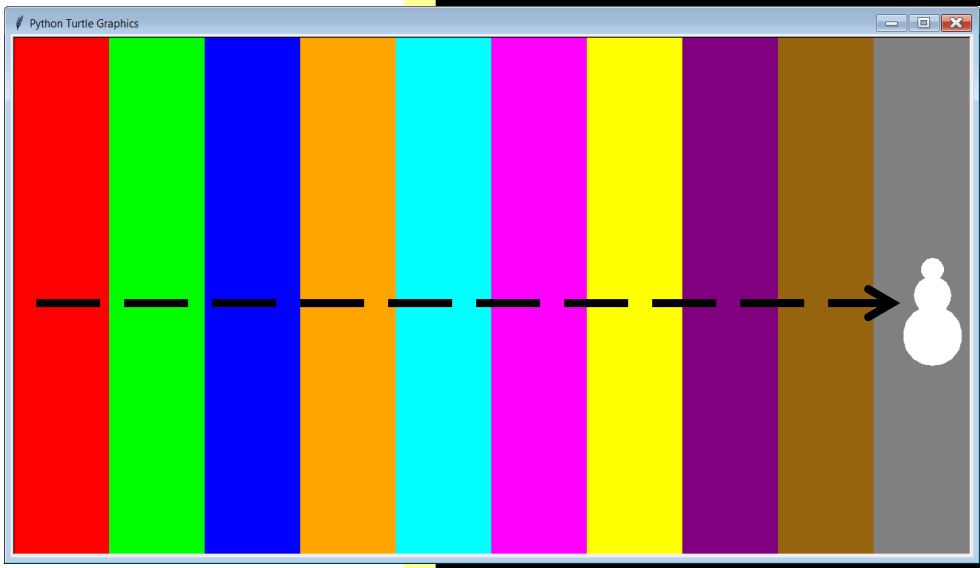
```
# Animation16.py
# This program shows another major problem with
# "Draw-And-Erase Animation."  It falls flat on
# its face when you have a multi-colored background.


from Graphics import *


def drawColorfulBackground():
    for c in range(10):
        setColor(c+1)
        fillRectangle(c*130, 0, (c+1)*130, 700)



##########
#  MAIN  #
##########

beginGrfx(1300,700)

drawColorfulBackground()

for x in range(50,1251,2):
    setColor("white")
    fillCircle(x,315,15)
    fillCircle(x,350,25)
    fillCircle(x,405,40)
    delay(10)
    setColor("red")
    fillCircle(x,315,17)
    fillCircle(x,350,27)
    fillCircle(x,405,42)

endGrfx()
```

```
1  # Animation16.py
2  # This program shows another major problem with
3  # "Draw-And-Erase Animation."  It falls flat on
4  # its face when you have a multi-colored background.
5
6
7  from Graphics import *
8
9
10 def drawColorfulBackground():
11     for c in range(10):
12         setColor(c+1)
13         fillRectangle(c*130, 0, (c+1)*130, 700)
14
15
16
17 ###########
18 #  MAIN   #
19 ###########
20
21 beginGrfx(1300,700)
22
23 drawColorfulBackground()
24
25 for x in range(50,1251,2):
26     setColor("white")
27     fillCircle(x,315,15)
28     fillCircle(x,350,25)
29     fillCircle(x,405,40)
30     delay(10)
31     setColor("red")
32     fillCircle(x,315,17)
33     fillCircle(x,350,27)
34     fillCircle(x,405,42)
35
36 endGrfx()
```



```
1  # Animation17.py
2  # This program shows how to handle a multi-colored
3  # background. By using "Draw-And-Redraw Animation"
4  # instead of "Draw-And-Erase" the entire background
5  # is redrawn every time the object moves.
6
7
8  from Graphics import *
9
10
11 def drawColorfulBackground():
12     for c in range(10):
13         setColor(c+1)
14         fillRectangle(c*130, 0, (c+1)*130, 700)
15
16
17
18 ##########
19 #  MAIN  #
20 ##########
21
22 beginGrfx(1300,700)
23
24 for x in range(50,1251,2):
25     clear()
26     drawColorfulBackground()
27     setColor("white")
28     fillCircle(x,315,15)
29     fillCircle(x,350,25)
30     fillCircle(x,405,40)
31     delay(10)
32
33 endGrfx()
```

```
# Animation17.py
# This program shows how to handle a multi-colored
# background. By using "Draw-And-Redraw Animation"
# instead of "Draw-And-Erase" the entire background
# is redrawn every time the object moves.


from Graphics import *


def drawColorfulBackground():
    for c in range(10):
        setColor(c+1)
        fillRectangle(c*130, 0, (c+1)*130, 700)



##########
#  MAIN  #
##########

beginGrfx(1300,700)

for x in range(50,1251,2):
    clear()
    drawColorfulBackground()
    setColor("white")
    fillCircle(x,315,15)
    fillCircle(x,350,25)
    fillCircle(x,405,40)
    delay(10)

endGrfx()
```



Draw-And-Redraw Animation Step-By-Step Review

1. Clear the screen.
2. Draw the entire background (preferably by using a procedure).
3. Draw the foreground image.
4. Delay for a fraction of a second.
5. Repeat this process over and over each time drawing the foreground image in a slightly different location.

This is the recommended approach for Computer Animation in our class because it solves BOTH of the problems of *Draw and Erase Animation*:

1. It works even if your background is not one solid color.
2. The animation will not slow down, even for complicated images.

Section 18.7

The Last Word

Programming Knowledge Never Becomes Obsolete

Some people may be concerned that with technology changing so rapidly, the programming knowledge that they have acquired in this course will soon be out of date. While new programming languages pop up from time to time, the logic of programming has never changed.

The next 6 programs all do the exact same thing:
The 1st is written in Python. The 2nd is in Java. The 3rd is in C++. The 4th is in Pascal; the 5th is in BASIC; and the 6th is written in VEXcode VR Blocks. Even though you may have never seen some of these languages before, you may be surprised how familiar these programs will look to you now that you know Python.

```
1 # LastWord01.py
2 # This program will count backwards from 20 to 0.
3 # For each number, it will display whether it has
4 # 1 or 2 digits. Compare this Python code to the
5 # code from other languages.
6
7
8 print()
9 for k in range (20, -1, -1):
10     if k >= 10:
11         print (k,"is a 2-digit number.")
12     else:
13         print (k,"is a 1-digit number.")
14
```

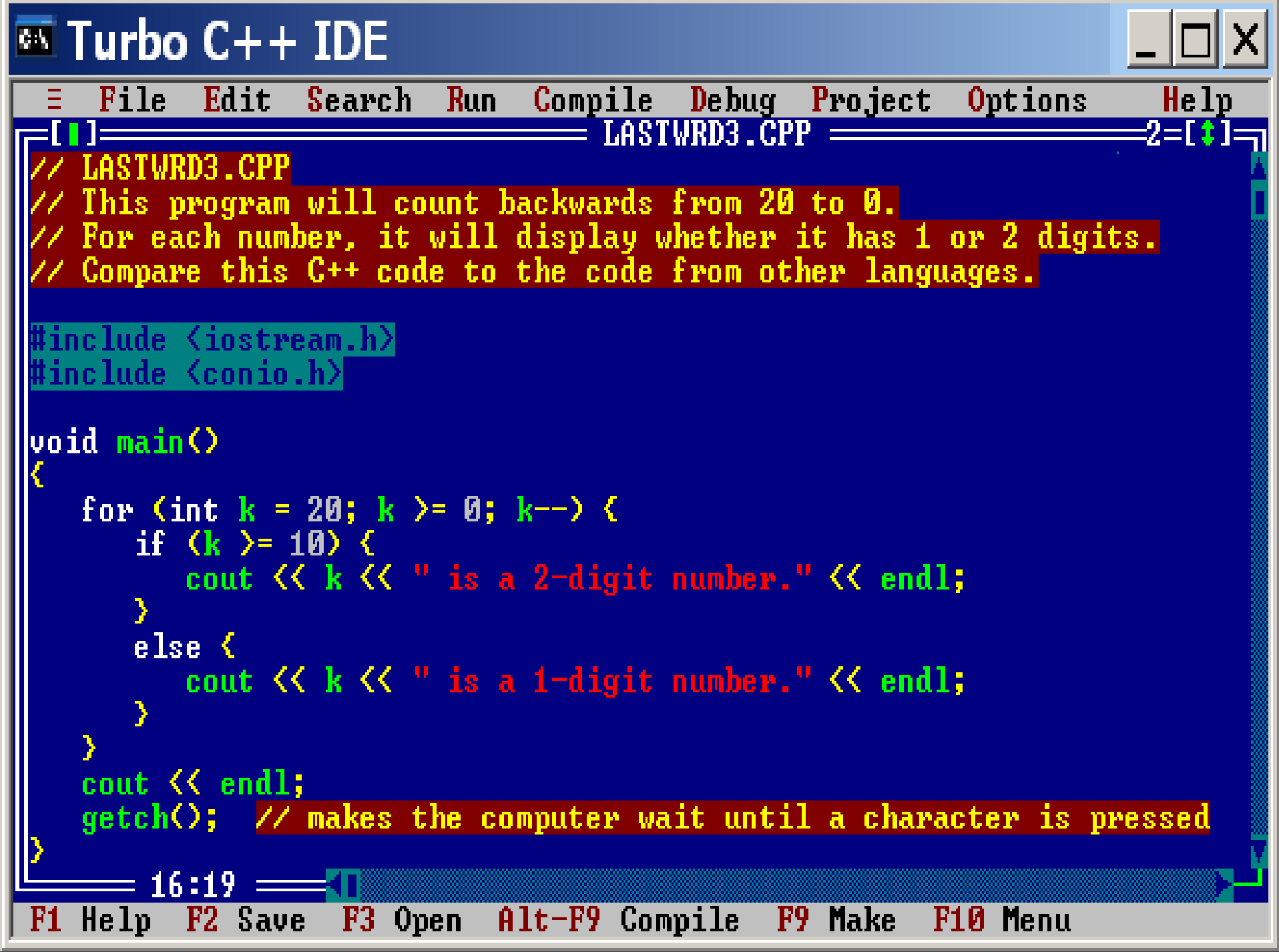
```
1  # LastWord01.py
2  # This program will count backw
3  # For each number, it will disp
4  # 1 or 2 digits. Compare this P
5  # code from other languages.
6
7
8  print()
9  for k in range (20, -1, -1):
10     if k >= 10:
11         print (k,"is a 2-digit
12     else:
13         print (k,"is a 1-digit
14
```

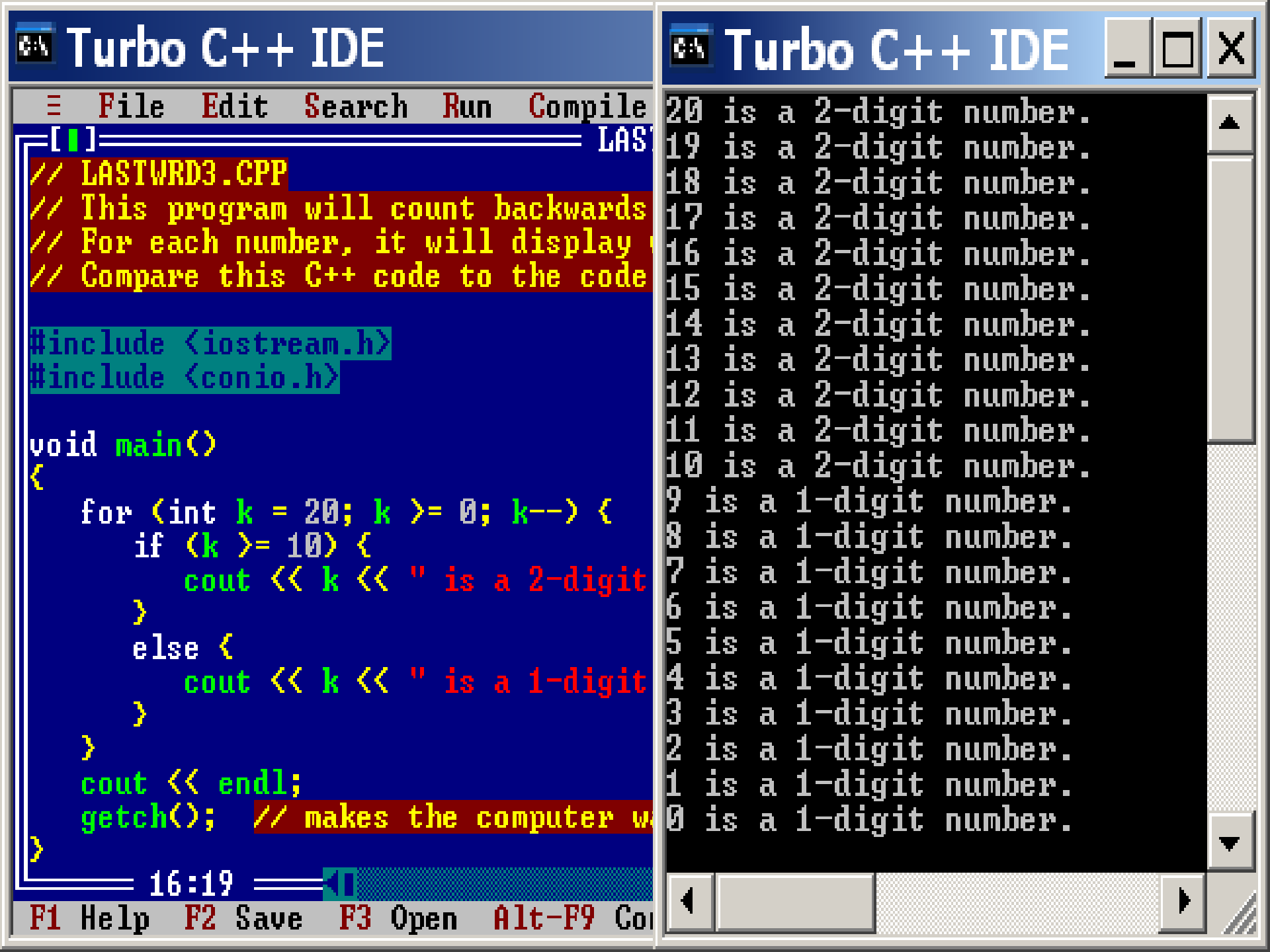
```
----jGRASP exec: python

20 is a 2-digit number.
19 is a 2-digit number.
18 is a 2-digit number.
17 is a 2-digit number.
16 is a 2-digit number.
15 is a 2-digit number.
14 is a 2-digit number.
13 is a 2-digit number.
12 is a 2-digit number.
11 is a 2-digit number.
10 is a 2-digit number.
9 is a 1-digit number.
8 is a 1-digit number.
7 is a 1-digit number.
6 is a 1-digit number.
5 is a 1-digit number.
4 is a 1-digit number.
3 is a 1-digit number.
2 is a 1-digit number.
1 is a 1-digit number.
0 is a 1-digit number.

----jGRASP: operation c
```

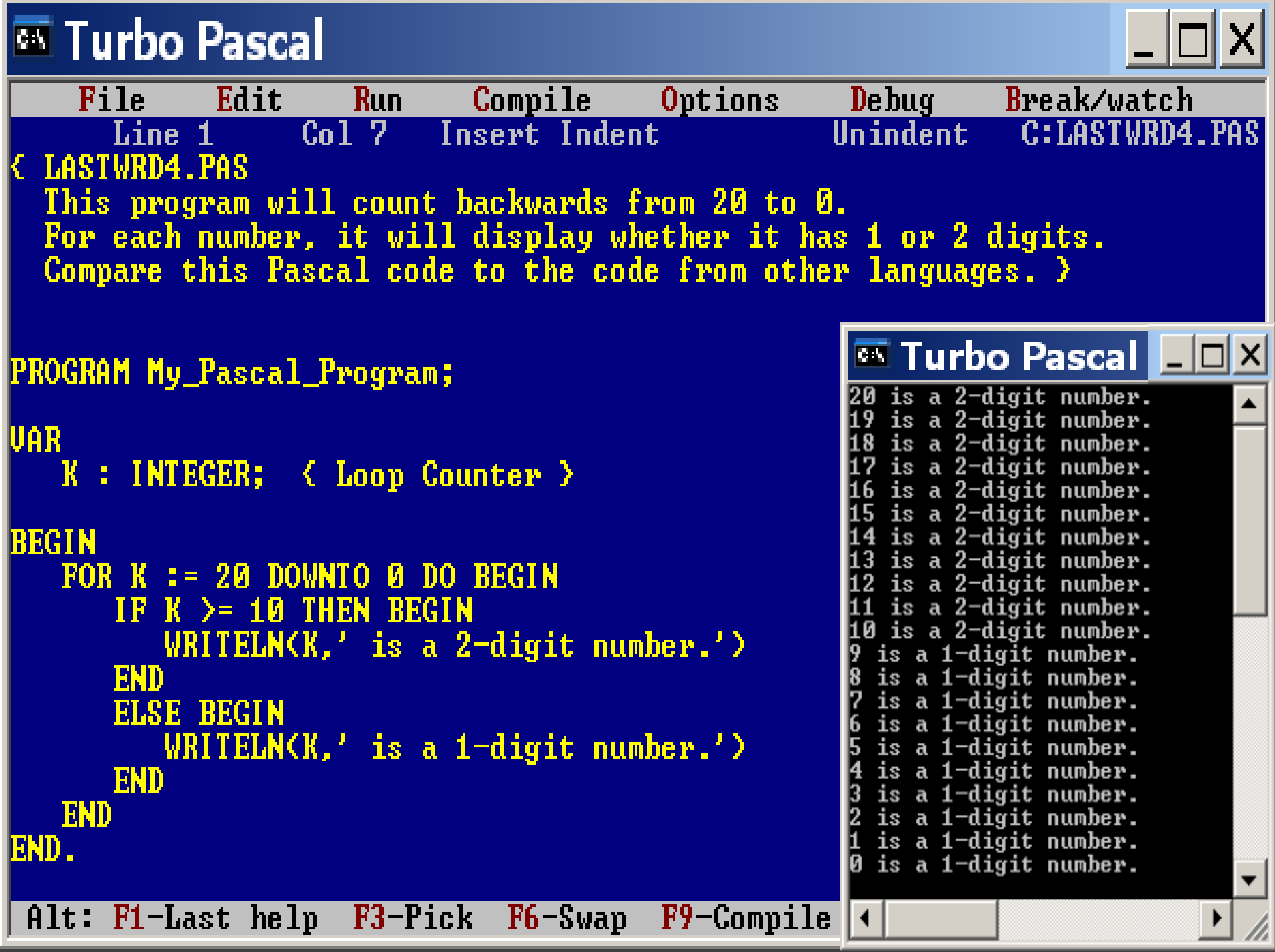
```java
// LastWord02.java
// This program will count backwards from 20 to 0.
// For each number, it will display whether it has
// 1 or 2 digits. Compare this Java code to the
// code from other languages.

public class LastWord02
{
   public static void main (String args[])
   {
      System.out.println();
      for (int k = 20; k >= 0; k--)
      {
         if (k >= 10)
         {
            System.out.println(k + " is a 2-digit number.");
         }
         else
         {
            System.out.println(k + " is a 1-digit number.");
         }
      }
   }
}
```

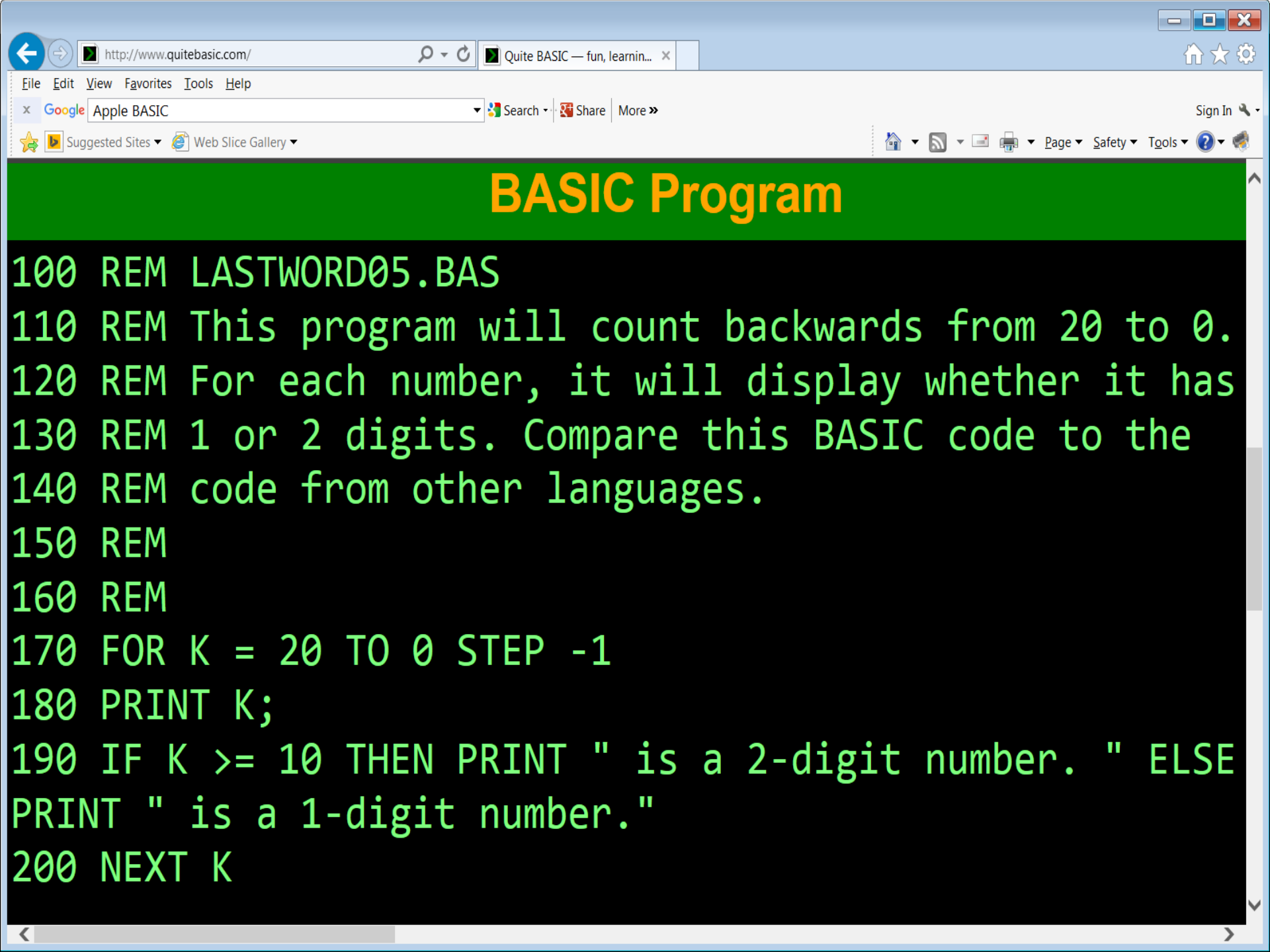
```java
// LastWord02.java
// This program will count backwards from 2(
// For each number, it will display whether
// 1 or 2 digits. Compare this Java code to
// code from other languages.

public class LastWord02
{
   public static void main (String args[])
   {
      System.out.println();
      for (int k = 20; k >= 0; k--)
      {
         if (k >= 10)
         {
            System.out.println(k + " is a 2
         }
         else
         {
            System.out.println(k + " is a 1
         }
      }
   }
}
```

```
----jGRASP exec: java

20 is a 2-digit number.
19 is a 2-digit number.
18 is a 2-digit number.
17 is a 2-digit number.
16 is a 2-digit number.
15 is a 2-digit number.
14 is a 2-digit number.
13 is a 2-digit number.
12 is a 2-digit number.
11 is a 2-digit number.
10 is a 2-digit number.
9 is a 1-digit number.
8 is a 1-digit number.
7 is a 1-digit number.
6 is a 1-digit number.
5 is a 1-digit number.
4 is a 1-digit number.
3 is a 1-digit number.
2 is a 1-digit number.
1 is a 1-digit number.
0 is a 1-digit number.

----jGRASP: operation
```


Turbo C++ IDE
File Edit Search Run Compile Debug Project Options Help
LASTWRD3.CPP
// LASTWRD3.CPP
// This program will count backwards from 20 to 0.
// For each number, it will display whether it has 1 or 2 digits.
// Compare this C++ code to the code from other languages.
#include <iostream.h>
#include <conio.h>
void main()
{
for (int k = 20; k >= 0; k--) {
if (k >= 10) {
cout << k << " is a 2-digit number." << endl;
}
else {
cout << k << " is a 1-digit number." << endl;
}
}
cout << endl;
getch(); // makes the computer wait until a character is pressed
}
16:19
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu


Turbo C++ IDE
File Edit Search Run Compile
// LASTWRD3.CPP
// This program will count backwards
// For each number, it will display
// Compare this C++ code to the code
#include <iostream.h>
#include <conio.h>
void main()
{
for (int k = 20; k >= 0; k--) {
if (k >= 10) {
cout << k << " is a 2-digit
}
else {
cout << k << " is a 1-digit
}
}
cout << endl;
getch(); // makes the computer w
}
16:19
F1 Help F2 Save F3 Open Alt-F9 Co
Turbo C++ IDE
20 is a 2-digit number.
19 is a 2-digit number.
18 is a 2-digit number.
17 is a 2-digit number.
16 is a 2-digit number.
15 is a 2-digit number.
14 is a 2-digit number.
13 is a 2-digit number.
12 is a 2-digit number.
11 is a 2-digit number.
10 is a 2-digit number.
9 is a 1-digit number.
8 is a 1-digit number.
7 is a 1-digit number.
6 is a 1-digit number.
5 is a 1-digit number.
4 is a 1-digit number.
3 is a 1-digit number.
2 is a 1-digit number.
1 is a 1-digit number.
0 is a 1-digit number.


Turbo Pascal
File Edit Run Compile Options Debug Break/watch
Line 1 Col 7 Insert Indent Unindent C:LASTWRD4.PAS
{ LASTWRD4.PAS
This program will count backwards from 20 to 0.
For each number, it will display whether it has 1 or 2 digits.
Compare this Pascal code to the code from other languages. }
PROGRAM My_Pascal_Program;
VAR
K : INTEGER; { Loop Counter }
BEGIN
FOR K := 20 DOWNTO 0 DO BEGIN
IF K >= 10 THEN BEGIN
WRITELN(K,' is a 2-digit number.')
END
ELSE BEGIN
WRITELN(K,' is a 1-digit number.')
END
END
END.
Alt: F1-Last help F3-Pick F6-Swap F9-Compile
Turbo Pascal
20 is a 2-digit number.
19 is a 2-digit number.
18 is a 2-digit number.
17 is a 2-digit number.
16 is a 2-digit number.
15 is a 2-digit number.
14 is a 2-digit number.
13 is a 2-digit number.
12 is a 2-digit number.
11 is a 2-digit number.
10 is a 2-digit number.
9 is a 1-digit number.
8 is a 1-digit number.
7 is a 1-digit number.
6 is a 1-digit number.
5 is a 1-digit number.
4 is a 1-digit number.
3 is a 1-digit number.
2 is a 1-digit number.
1 is a 1-digit number.
0 is a 1-digit number.


http://www.quitebasic.com/
Quite BASIC — fun, learnin...
File Edit View Favorites Tools Help
Google Apple BASIC
Search
Share
More »
Sign In
Suggested Sites
Web Slice Gallery
Page
Safety
Tools
BASIC Program
100 REM LASTWORD05.BAS
110 REM This program will count backwards from 20 to 0.
120 REM For each number, it will display whether it has
130 REM 1 or 2 digits. Compare this BASIC code to the
140 REM code from other languages.
150 REM
160 REM
170 FOR K = 20 TO 0 STEP -1
180 PRINT K;
190 IF K >= 10 THEN PRINT " is a 2-digit number. " ELSE
PRINT " is a 1-digit number."
200 NEXT K

Output

```
20 is a 2-digit number.
19 is a 2-digit number.
18 is a 2-digit number.
17 is a 2-digit number.
16 is a 2-digit number.
15 is a 2-digit number.
14 is a 2-digit number.
13 is a 2-digit number.
12 is a 2-digit number.
11 is a 2-digit number.
10 is a 2-digit number.
9 is a 1-digit number.
8 is a 1-digit number.
7 is a 1-digit number.
6 is a 1-digit number.
5 is a 1-digit number.
4 is a 1-digit number.
3 is a 1-digit number.
2 is a 1-digit number.
1 is a 1-digit number.
0 is a 1-digit number.
```

VR | File | TUTORIALS | ACTIVITIES | LEARN | LastWord06 | PLAYGROUND | START | STEP | STOP | SHARE | FEEDBACK

Code

Drivetrain
Magnet
Looks
Control
Sensing
Operators
Variables
My Blocks
Comments

```
when started
LastWord06.vrblocks
This program will count backwards from 20 to 0.
For each number, it will display whether it has 1 or 2 digits.
Compare this VEXcode VR Blocks program to the code from other languages.
clear all rows
set k to 20
repeat 21
    print k
    if k > 9 then
        print is a 2-digit number.
    else
        print is a 1-digit number.
    set cursor to next row
    change k by -1
```

Monitor

Sensors

Variables

```
20 is a 2-digit number.
19 is a 2-digit number.
18 is a 2-digit number.
17 is a 2-digit number.
16 is a 2-digit number.
15 is a 2-digit number.
14 is a 2-digit number.
13 is a 2-digit number.
12 is a 2-digit number.
11 is a 2-digit number.
10 is a 2-digit number.
9 is a 1-digit number.
8 is a 1-digit number.
7 is a 1-digit number.
6 is a 1-digit number.
5 is a 1-digit number.
4 is a 1-digit number.
3 is a 1-digit number.
2 is a 1-digit number.
1 is a 1-digit number.
0 is a 1-digit number.
```

CLEAR | SAVE | COPY TO CLIPBOARD