BIS 420 PROGRAMMING FOR DATA SCIENCE

PRAJAKTA POHARE
CHAPTER 4 EXERCISE 4.1
ILLINOIS STATE UNIVERSITY

Download the code in this chapter from http: // thinkpython. com/ code/

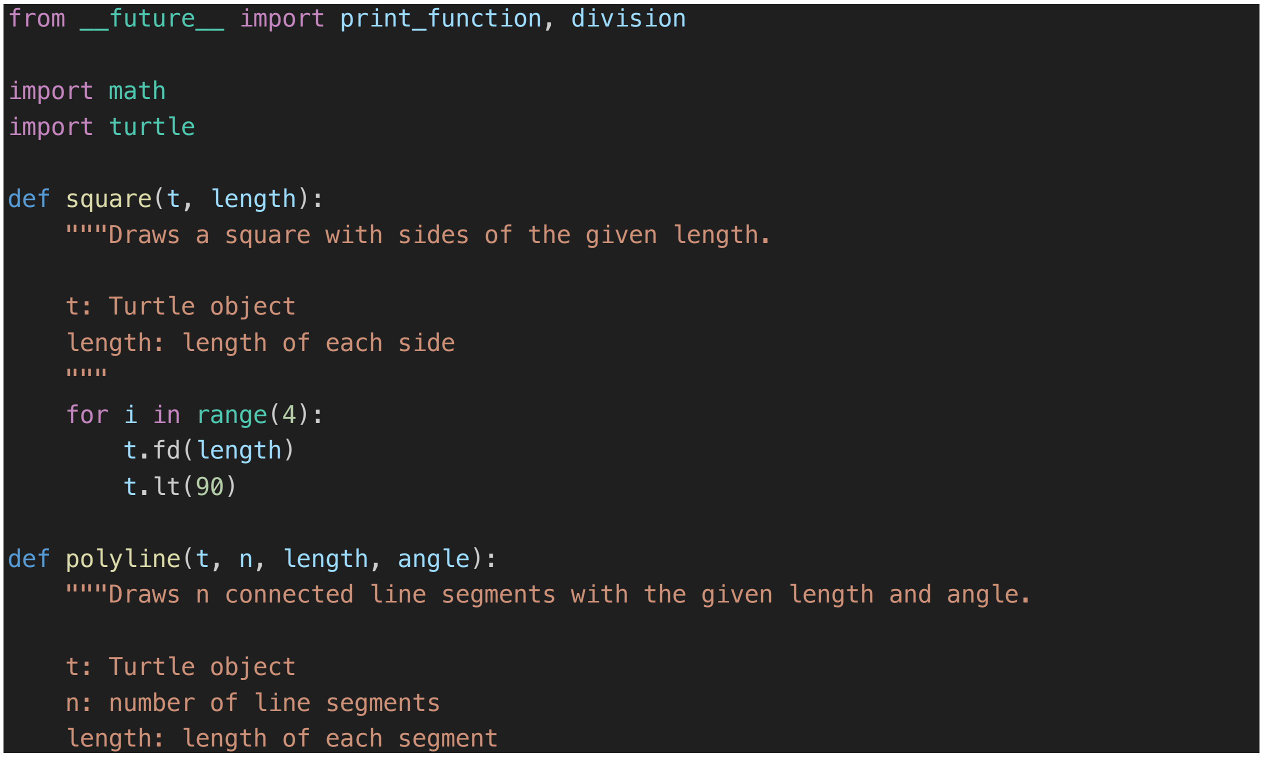
polygon. py .

1. Write appropriate docstrings for polygon, arc and circle.

2. Draw a stack diagram that shows the state of the program while executing circle(bob, radius). You can do the arithmetic by hand or add print statements to the code.

3. The version of arc in Section 4.7 is not very accurate because the linear approximation of the circle is always outside the true circle. As a result, the turtle ends up a few units away from the correct destination. My solution shows a way to reduce the effect of this error. Read the code and see if it makes sense to you. If you draw a diagram, you might see how it works.

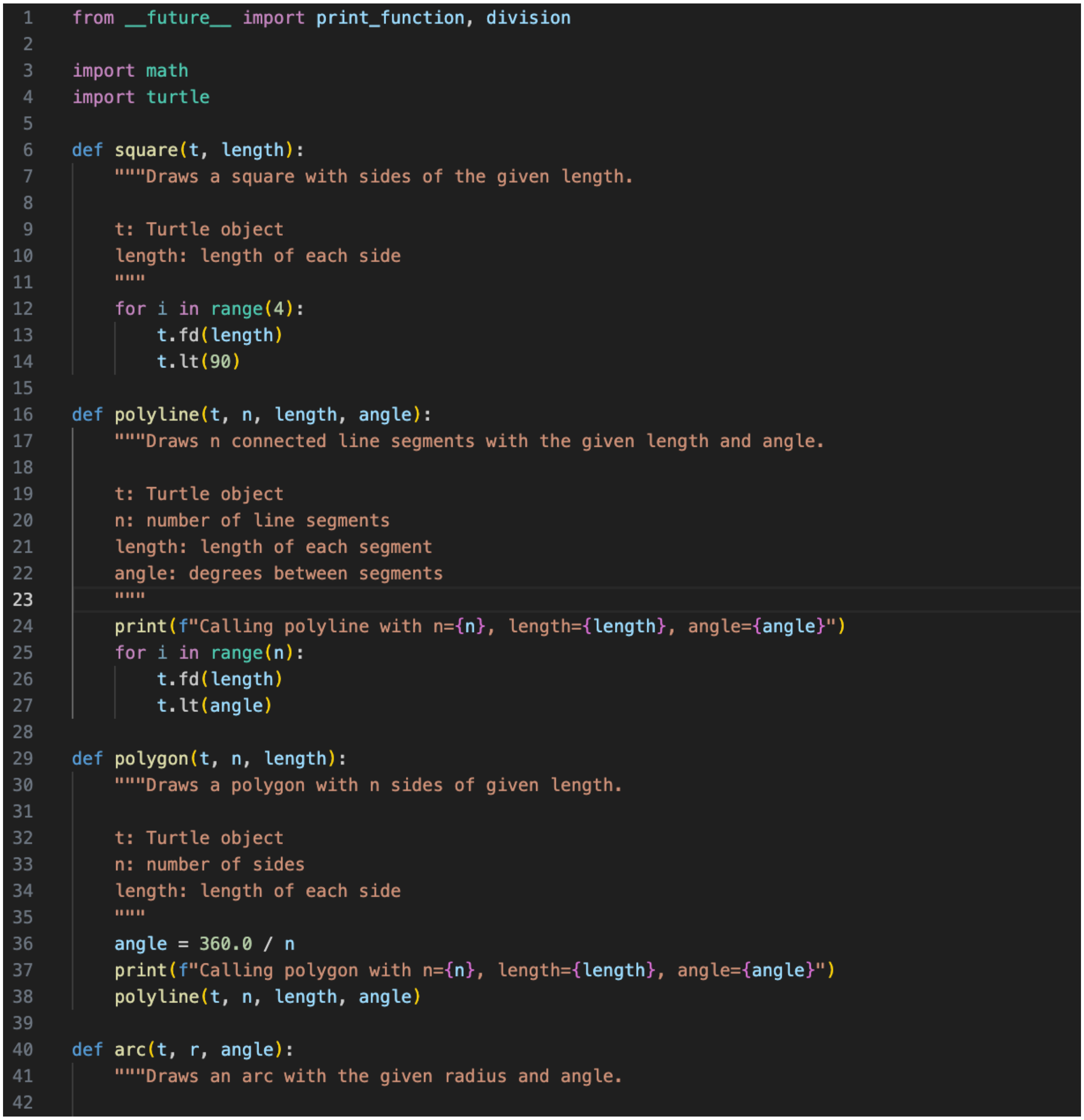
1,2)



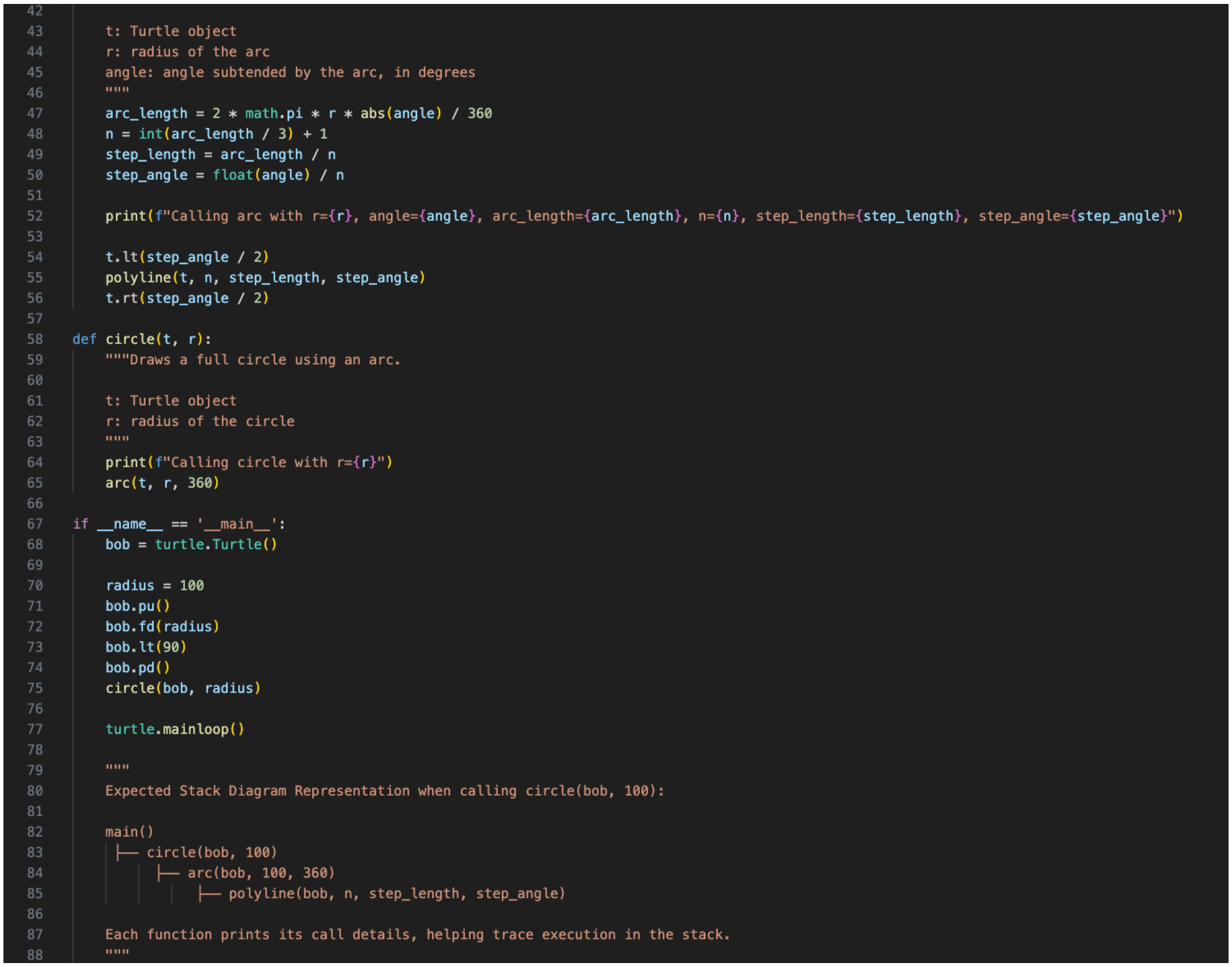
```
from __future__ import print_function, division

import math
import turtle

def square(t, length):
    """Draws a square with sides of the given length.

    t: Turtle object
    length: length of each side
    """
    for i in range(4):
        t.fd(length)
        t.lt(90)

def polyline(t, n, length, angle):
    """Draws n connected line segments with the given length and angle.

    t: Turtle object
    n: number of line segments
    length: length of each segment
```

```python
    angle: degrees between segments
    """
    print(f"Calling polyline with n={n}, length={length}, angle={angle}")
    for i in range(n):
        t.fd(length)
        t.lt(angle)

def polygon(t, n, length):
    """Draws a polygon with n sides of given length.

    t: Turtle object
    n: number of sides
    length: length of each side
    """
    angle = 360.0 / n
    print(f"Calling polygon with n={n}, length={length}, angle={angle}")
    polyline(t, n, length, angle)

def arc(t, r, angle):
    """Draws an arc with the given radius and angle.

    t: Turtle object
    r: radius of the arc
    angle: angle subtended by the arc, in degrees
    """
    arc_length = 2 * math.pi * r * abs(angle) / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n

    print(f"Calling arc with r={r}, angle={angle}, arc_length={arc_length}, n={n}, step_length={step_length}, step_angle={step_angle}")

    t.lt(step_angle / 2)
    polyline(t, n, step_length, step_angle)
    t.rt(step_angle / 2)

def circle(t, r):
    """Draws a full circle using an arc.

    t: Turtle object
    r: radius of the circle
    """
    print(f"Calling circle with r={r}")
    arc(t, r, 360)

if __name__ == '__main__':
    bob = turtle.Turtle()
```

```
radius = 100
bob.pu()
bob.fd(radius)
bob.lt(90)
bob.pd()
circle(bob, radius)

turtle.mainloop()

"""
Expected Stack Diagram Representation when calling circle(bob, 100):

main()
 ├── circle(bob, 100)
      ├── arc(bob, 100, 360)
           ├── polyline(bob, n, step_length, step_angle)

Each function prints its call details, helping trace execution in the stack.
"""
```

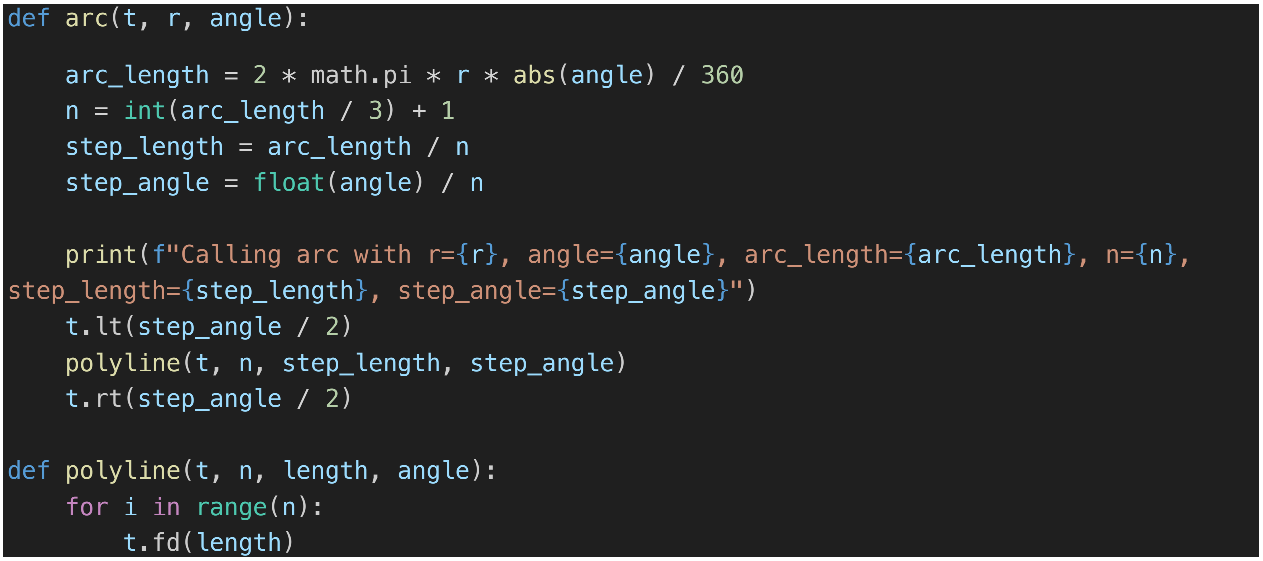


```python
from __future__ import print_function, division

import math
import turtle

def square(t, length):
    """Draws a square with sides of the given length.

    t: Turtle object
    length: length of each side
    """
    for i in range(4):
        t.fd(length)
        t.lt(90)

def polyline(t, n, length, angle):
    """Draws n connected line segments with the given length and angle.

    t: Turtle object
    n: number of line segments
    length: length of each segment
    angle: degrees between segments
    """
    print(f"Calling polyline with n={n}, length={length}, angle={angle}")
    for i in range(n):
        t.fd(length)
        t.lt(angle)

def polygon(t, n, length):
    """Draws a polygon with n sides of given length.

    t: Turtle object
    n: number of sides
    length: length of each side
    """
    angle = 360.0 / n
    print(f"Calling polygon with n={n}, length={length}, angle={angle}")
    polyline(t, n, length, angle)

def arc(t, r, angle):
    """Draws an arc with the given radius and angle.

```

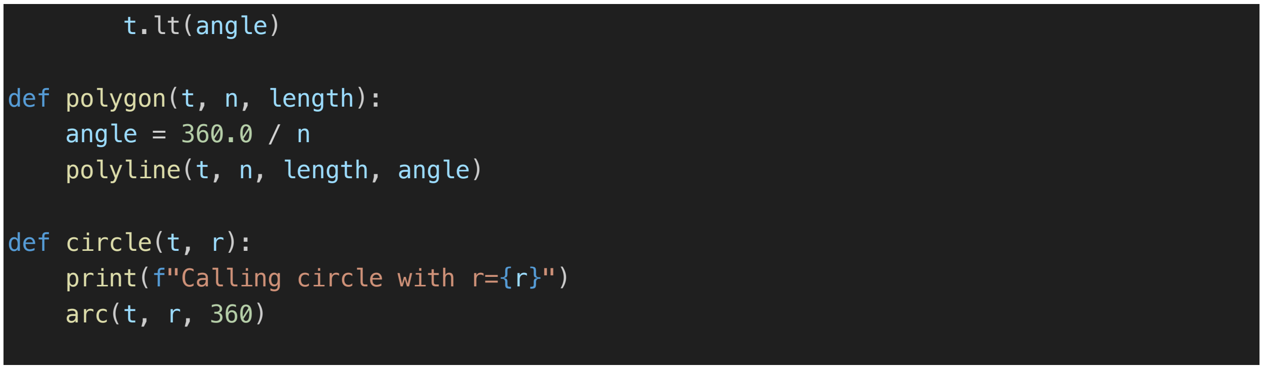


```
    t: Turtle object
    r: radius of the arc
    angle: angle subtended by the arc, in degrees
    """
    arc_length = 2 * math.pi * r * abs(angle) / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n

    print(f"Calling arc with r={r}, angle={angle}, arc_length={arc_length}, n={n}, step_length={step_length}, step_angle={step_angle}")

    t.lt(step_angle / 2)
    polyline(t, n, step_length, step_angle)
    t.rt(step_angle / 2)

def circle(t, r):
    """Draws a full circle using an arc.

    t: Turtle object
    r: radius of the circle
    """
    print(f"Calling circle with r={r}")
    arc(t, r, 360)

if __name__ == '__main__':
    bob = turtle.Turtle()

    radius = 100
    bob.pu()
    bob.fd(radius)
    bob.lt(90)
    bob.pd()
    circle(bob, radius)

    turtle.mainloop()

    """
    Expected Stack Diagram Representation when calling circle(bob, 100):

    main()
      ├── circle(bob, 100)
           ├── arc(bob, 100, 360)
                ├── polyline(bob, n, step_length, step_angle)

    Each function prints its call details, helping trace execution in the stack.
    """
```

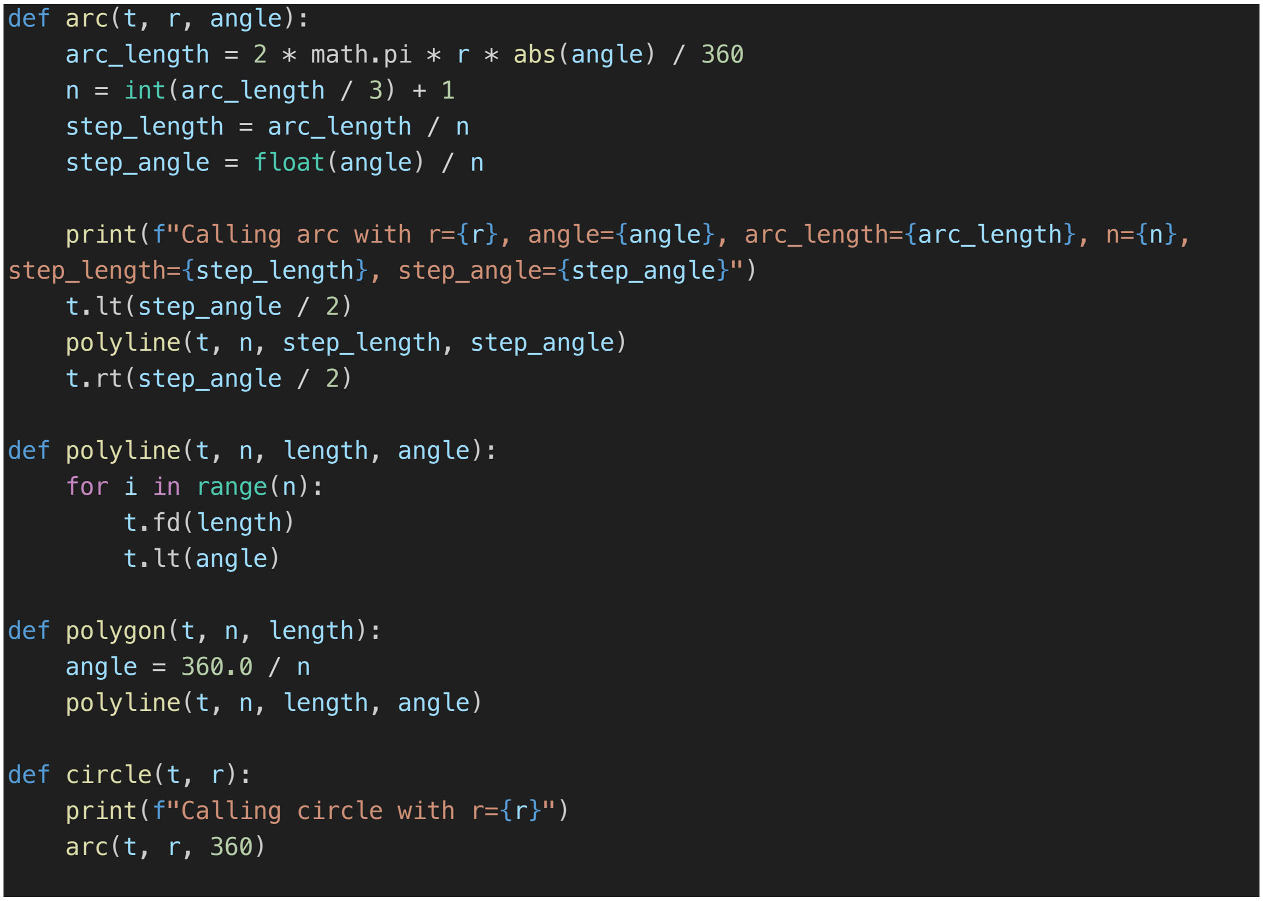
3)



```
def arc(t, r, angle):

    arc_length = 2 * math.pi * r * abs(angle) / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n

    print(f"Calling arc with r={r}, angle={angle}, arc_length={arc_length}, n={n}, 
step_length={step_length}, step_angle={step_angle}")
    t.lt(step_angle / 2)
    polyline(t, n, step_length, step_angle)
    t.rt(step_angle / 2)

def polyline(t, n, length, angle):
    for i in range(n):
        t.fd(length)
```



```python
        t.lt(angle)

def polygon(t, n, length):
    angle = 360.0 / n
    polyline(t, n, length, angle)

def circle(t, r):
    print(f"Calling circle with r={r}")
    arc(t, r, 360)
```



```python
def arc(t, r, angle):
    arc_length = 2 * math.pi * r * abs(angle) / 360
    n = int(arc_length / 3) + 1
    step_length = arc_length / n
    step_angle = float(angle) / n

    print(f"Calling arc with r={r}, angle={angle}, arc_length={arc_length}, n={n}, step_length={step_length}, step_angle={step_angle}")
    t.lt(step_angle / 2)
    polyline(t, n, step_length, step_angle)
    t.rt(step_angle / 2)

def polyline(t, n, length, angle):
    for i in range(n):
        t.fd(length)
        t.lt(angle)

def polygon(t, n, length):
    angle = 360.0 / n
    polyline(t, n, length, angle)

def circle(t, r):
    print(f"Calling circle with r={r}")
    arc(t, r, 360)
```