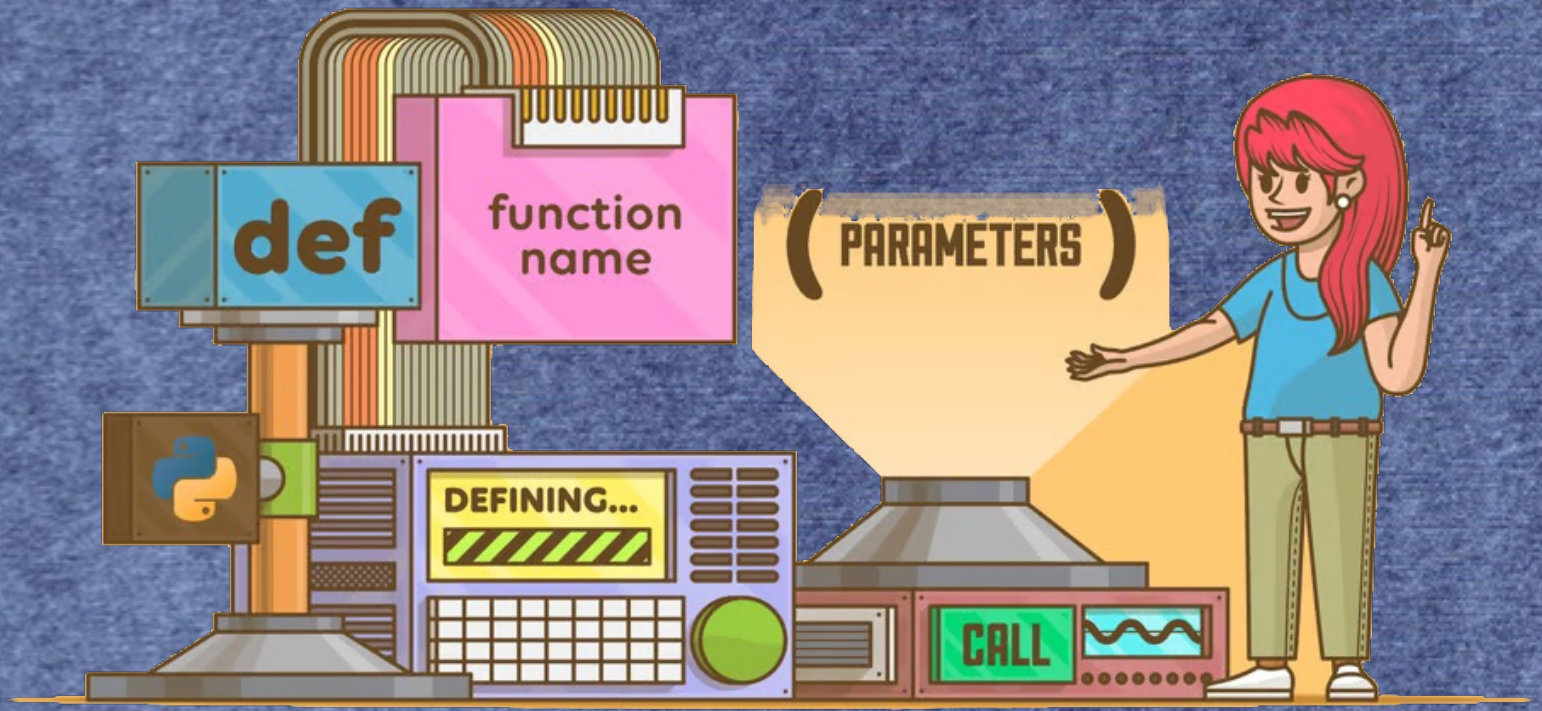
Functions



Functions

- What are functions?
- Syntax
- Calling
- Parameters
- Return values
- Variable Scope

Functions

- Named sequence of statements that belong together
- Primary purpose
  - organize programs into chunks
  - that match how we think about problem solution

Functions

- Compound Statement
  - –consists of one or several statements
  - –: used to indicate beginning
  - –indentation

```
def name( parameters ):
    statements
```

Function Syntax

```
def name( parameters ):
    statements
```

Function Example

```
def drawSquare():                    #function definition
    tess.forward(120)
    tess.left(90)
    tess.forward(120)
    tess.left(90)
    tess.forward(120)
    tess.left(90)
    tess.forward(120)
    tess.left(90)


drawSquare()                 # function call
```

Function Example

```
def drawSquare():                    #function definition
    for side in range(4):
                tess.forward(120)
                tess.left(90)


drawSquare()                         # function call
```

You Try It

- modify rotateSquare.py to use a function to draw one of the squares

```
def drawSquare():                    #function definition
    for side in range(4):
            tess.forward(120)
            tess.left(90)


drawSquare()                         # function call
```

Function Parameter Example

```
def drawSquare(sideLength):       #function definition
    for side in range(4):
            tess.forward(sideLength)
            tess.left(90)

tess.speed(0)

for square in range(36):
        drawSquare(150)           # function call
        tess.color(?)
        tess.left(5)
```

Function Parameter Example

```
import turtle
wn=turtle.Screen()

alex = turtle.Turtle()
tess = turtle.Turtle()

def drawSquare(turtle, sideLength):          # function definition
    for side in range(4):
                    turtle.forward(sideLength)
                    turtle.left(90)


drawSquare(alex, 150)                        # function call
drawSquare(tess, 300)
wn.exitonclick()
```

Function Return Example

```
def square(x):
        y = x * x
        return y

toSquare = 10

squareResult = square(toSquare)

print("The result of ", toSquare, " squared is ", squareResult)
```

You Try It

- write and test a function that calculates the area of a circle (πr^2) and returns the result

write and test a function that calculates the area of a circle (πr^2)

```
def circleArea(radius):
        area = 3.14 * radius * radius
        return area

radius = int(input('the radius is: '))

area = circleArea(radius)

print("The area of a circle of radius", radius, " is ", area)
```

Function Variable Scope

- variables and parameters are **local**
- functions can access **global** variables
  - but it is generally not good form

Function Variable Scope

```
def square():
        y = x * x
        return y

x = 10
z = square()

print(z)
print(x)
```

Function Variable Scope

```
def square(x):
        y = x * x
        x = 5
        return y

x = 10
z = square(x)

print(z)
print(x)
```

Function Variable Scope

- variables and parameters are **local**
- functions can access **global** variables
  - but it is generally not good form

Function Review

- What are functions?
- Syntax

```
def name( parameters ):
    statements
    return
```

- Calling
- Parameters
- Return values
- Variable Scope

Accumulator Pattern

```
def square(x):
    runningtotal = 0
    for counter in range(x):
        runningtotal = runningtotal + x

    return runningtotal

toSquare = 10
squareResult = square(toSquare)
print("The result of", toSquare, "squared is", squareResult)
```

Flow of Execution

```
def square(x):
    runningtotal = 0
    for counter in range(x):
        runningtotal = runningtotal + x

    return runningtotal

toSquare = 10
squareResult = square(toSquare)
print("The result of", toSquare, "squared is", squareResult)
```

Flow of Execution

```
 1 def pow(b, p):
 2       y = b ** p
 3       return y
 4
 5 def square(x):
 6       a = pow(x, 2)
 7       return a
 8
 9 n = 5
10 result = square(n)
11 print(result)
```

Which is correct?

A	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
B	1, 2, 3, 5, 6, 7, 9, 10, 11
C	9, 10, 11, 1, 2, 3, 5, 6, 7
D	9, 10, 5, 6, 7, 1, 2, 3, 11
E	1, 5, 9, 10, 6, 2, 3, 7, 11

Advanced Topic

```
def squareit(n):
    return n * n

def cubeit(n):
    return n*n*n

def main():
    anum = int(input("Please enter a number"))
    print(squareit(anum))
    print(cubeit(anum))

if __name__ == "__main__":
    main()
```

Program Development

- Incremental development
  - add and test small amounts of code
- Use temporary variables
- Consolidate once working

Distance = sqrt($(x1 - x2)^2 + (y1 - y2)^2$)

- create empty function

```
def distance(x1, x2, y1, y2)
    return 0.0
```

Distance = sqrt($(x1 - x2)^2 + (y1 - y2)^2$)

- create empty function
- choose values with known result
  - print(distance(1, 2, 4, 6))
- values chosen for 3, 4, 5 right triangle

```
def distance(x1, y1, x2, y2)
    return 0.0
```

Distance = sqrt($(x1 - x2)^2 + (y1 - y2)^2$)

- create empty function
- choose values with known result
- create intermediate values
  - so results can be checked

```
def distance(x1, y1, x2, y2)
        dx = x2 – x1
        dy = y2 – y1
        print(“dx=”, dx, “ dy=”, dy)
        return 0.0
```

Distance = sqrt($(x1 - x2)^2 + (y1 - y2)^2$)

- create empty function
- choose values with known result
- create intermediate values
  - so results can be checked

```
def distance(x1, y1, x2, y2)
        dx = x2 – x1
        dy = y2 – y1
        dsquared = dx**2 + dy**2
        print('dsquared=', dsquared)
        return 0.0
```

Distance = sqrt($(x1 - x2)^2 + (y1 - y2)^2$)

- create empty function
- choose values with known result
- create intermediate values
  - so results can be checked

```
def distance(x1, y1, x2, y2)
        dx = x2 – x1
        dy = y2 – y1
        dsquared = dx**2 + dy**2
        result = dsquared**0.5
        print('result=', result)
        return 0.0
```

Distance = sqrt($(x1 - x2)^2 + (y1 - y2)^2$)

- create empty function
- choose values with known result
- create intermediate values
- consolidate statements
  - only if readability doesn't suffer

```
def distance(x1, y1, x2, y2)
        dx = x2 – x1
        dy = y2 – y1
        dsquared = dx**2 + dy**2
        result = dsquared**0.5
        print('result=', result)
        return result
                        def distance(x1, y1, x2, y2)
                                return ( (x2 – x1)**2 + (y2 – y1)**2 )**0.5
```

Function Docstring Example

```
def drawSquare(sideLength):       #function definition
        """make turtle t draw a square with sides of
                        length sideLength"""
    for side in range(4):
                tess.forward(sideLength)
                tess.left(90)


drawSquare(150)                          # function call
```

Function Syntax

```
def drawSquare():
    for side in range(4):
        tess.forward(100)
        tess.left(90)

drawSquare()
```

```
def drawSquare(sideLength):
    for side in range(4):
        tess.forward(sideLength)
        tess.left(90)

drawSquare(150)
```

```
def calcRectArea(sideLength, topLength):
    area = sideLength * topLength
    return area

rectArea = calcRectArea(50, 80)
```