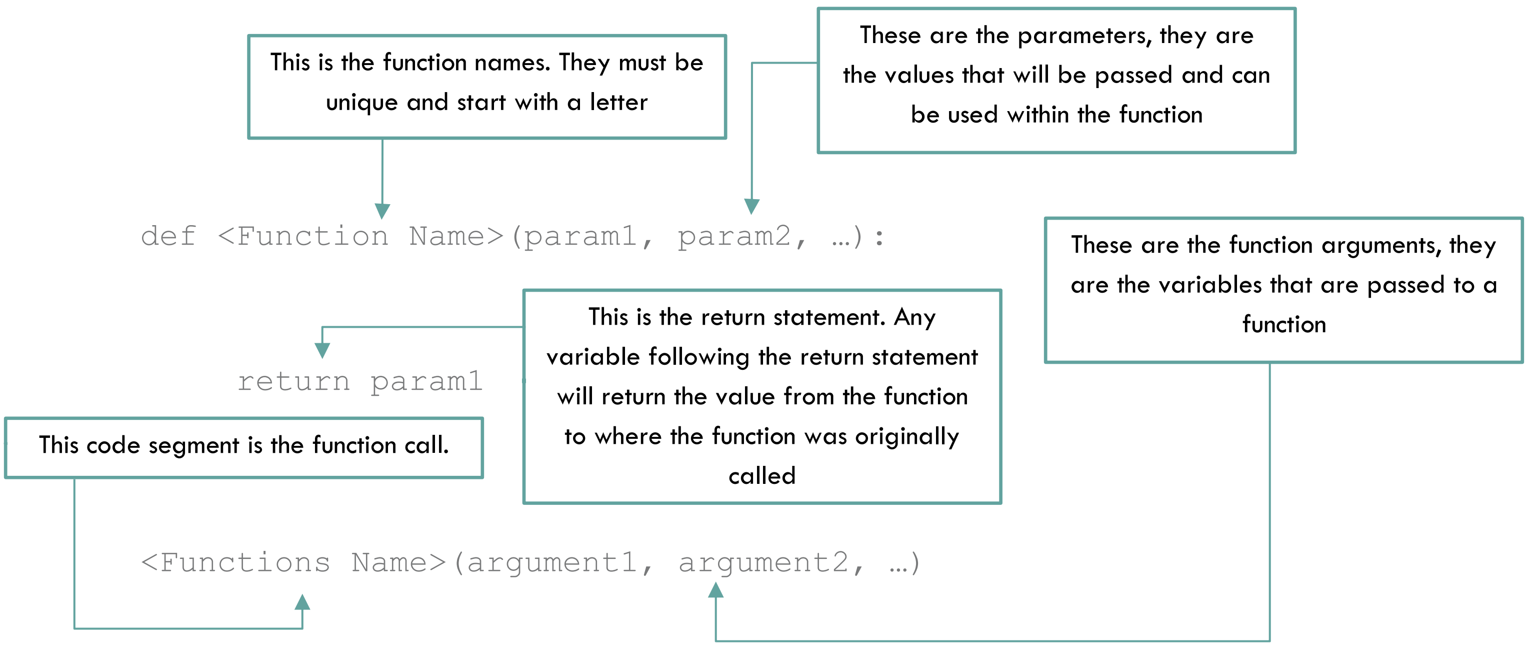
PYTHON NOTES (FUNCTIONS)

There are two fundamental reasons functions are helpful when programming. They help by dividing programs into smaller manageable pieces, also by taking advantage of code reusability. Functions take in an input value and return an output value to where the function was called.

The function syntax in python is the following.



This is the function names. They must be unique and start with a letter

These are the parameters, they are the values that will be passed and can be used within the function

```
def <Function Name>(param1, param2, …):

     return param1
```

These are the function arguments, they are the variables that are passed to a function

This is the return statement. Any variable following the return statement will return the value from the function to where the function was originally called

This code segment is the function call.

```
<Functions Name>(argument1, argument2, …)
```

Only variables declared globally, declared within the function, or passed as a parameter may be used. Variables declared. The user will receive a "UnboundLocalError" when trying to use variables that have not been created or passed within the function.

Functions do not necessarily need parameters and arguments when performing anything neither do they need to return a value. With simply a `return` that is not followed by a variable will return nothing when executed.

Reusability allows a simple way to execute the same sequence of code multiple times compared to copying the same lines of code into multiple sections of the program. Using the code above if we were to get the value for 2^3 and 3^2.

Example of solving problem from above without a function:

```
#without function example program
val = 1
base = 2
exp = 3

while(exp > 0):
  val = val * base
  exp-=1

print(val)

val=1
base = 3
exp = 2

while(exp > 0):
  val = val * base
  exp-=1

#prints 9
Print(val)
```

Example of solving problem from above with function:

```
#function example program
def pow(base, exp):
    if(exp == 0):
        return 1

    while(exp > 0):
        val = val * base
        exp-=1

    return val

val = 1

#prints 8
print(pow(2, 3))

#prints 9
Print(pow(3, 2))
```

Note that above it was a lot easier to make two functions calls of `pow` compared to the multiple `while` loops and reassigning variables. The code on the right has readability compared to the example on the left. The program stars2017.py (below) it is apparent the usefulness of functions shortening and simplifying the code if functions were not used.

```
# Arup Guha
# 6/19/2017
# Stars program to practice nested loops.

QUIT = 7

def main():

    menu()
    choice = int(input("What is your choice?\n"))

    # Continue printing until the user quits.
    while choice != QUIT:

        # All designs take in a character, so do this first.
        ch = input("What character do you want for your design?\n")
        # First choice - parallelogram
        if choice == 1:
            rows = int(input("How many rows in your parallelogram?\n"))
            cols = int(input("How many columns in your parallelogram?\n"))
            parallelogram(rows, cols, ch)

        # Second choice - right triangle, left justified
        elif choice == 2:
            n = int(input("How many rows for your left justified right
triangle?\n"))
            rightTri(n, ch)

        # Third choice - right triangle, right justified
        elif choice == 3:
            n = int(input("How many rows for your right justified right
triangle?\n"))
            rightJustifiedTri(n, ch)

        # Error message.
        elif choice != QUIT:
            print("Sorry, that was an invalid choice.")

        # Get next choice.
        menu()
        choice = int(input("What is your choice?\n"))

def menu():

    print("Please choose one of the following options:")
    print("1. Print a parallelogram.")
    print("2. Print a left justified right triangle.")
    print("3. Print a right justified right triangle.")
    #print("4. Print a sideways pyramid.")
    #print("5. Print a regular pyramid.")
    #print("6. Print a diamond.")
    print("7. Quit")

# Prints out a left justified triangle of n rows using character c.
def rightTri(n, c):

    # Print out row i.
    for i in range(1,n+1):

        # We need i copies of character c.
        for j in range(i):
            print(c,end="")
        print()

# Prints out a right justifed triangle of n rows using character c.
def rightJustifiedTri(n, c):

    # i is the row number.
    for i in range(1, n+1):

        # Print n-i spaces
        for j in range(n-i):
            print(" ", end="")
        # Print i copies of character c.
        for j in range(i):
            print(c, end="")

        # Go to the new line.
        print()

# Prints a parallelogram with rows # of rows, cols # of cols using
# character c.
def parallelogram(rows, cols, c):

    # Go through each row.
    for i in range(rows):

        # Print i spaces.
        for j in range(i):
            print(" ", end="")

        # Print cols copies of character c.
        for j in range(cols):
            print(c, end ="")

        # Go to the next line.
        print()

main()
```

When using functions in python it is important to note that changes to variables passed to the function will remain the same value it was prior to the function after returning from the function. To combat this issue Global variables may be used or returning the needed value from the function.

Example of global variable:

```
#function example program

val = 1
def main():
    #prints 1
    print(val)

    pow(2, 3)

    #prints 8
    print(val)

def pow(base, exp):
    global val

    if(exp == 0):
        return 1

    while(exp > 0):
        val = val * base
```

Example of returning value:

```
#function example program

def main():
    val = 1
    #prints 1
    print(val)

    val = pow(2, 3)

    #prints 8
    print(val)

def pow(base, exp):
    if(exp == 0):
        return 1

    while(exp > 0):
        val = val * base
        exp-=1
        exp-=1

    #prints 8
    print(val)

main()
```

```
    #prints 8
    print(val)

    return val

main()
```

Functions may be called within the functions or functions may also be called within function calls in fueleff_func.py we can see the function `main` calls the functions `calcDistance`, `calcMpg`, `calcTotalCost`, and `printCarInfo`.

```
# Conner Brooks
# 7/6/2012
# An example that calculates fuel efficiency using functions.

#!/usr/bin/python

def main():
        #get input
        print("Space trip logging program.")
        makeCar = input('Please enter make and model of car ')
        startMiles = int( input('enter initial reading on the odometer '))
        endMiles = int( input('enter final reading on the odometer '))
        gallonsUsed = int( input('enter gallons of gas used '))
        pricePerGallon = float( input('enter price of one gallon of gas '))

        #calculate values
        distance = calcDistance(startMiles, endMiles)
        milesPerGallon = calcMpg(distance, gallonsUsed)
        totalPrice = calcTotalCost(pricePerGallon, gallonsUsed)


        #print values
        printCarInfo(makeCar, startMiles, endMiles, distance, milesPerGallon,
totalPrice)



def calcDistance(sMiles, eMiles):
        return eMiles - sMiles

def calcMpg(dist, gallons):
        return dist / gallons

def calcTotalCost(price, used):
        return price * used

def printCarInfo(make, start, end, dist, mpg, total):
        print('Make & Model: ' + make)
        print('Initial odometer reading: ' + str(start))
        print('Final odometer reading: ' + str(end))
        print('Distance traveled: ' + str(dist))
        print('MPG: ' + str(mpg))
        print('Total cost of trip: ' + str(total))
```

When using Pygame functions play a big role when it comes to taking a large problem and making it simpler to code. For example using chessboard4.py there are multiple pieces to solve the problem of building a chessboard instead of tackling the problem as a whole we can separate each piece and tackle the problem as pieces opposed to a whole. In chessboard4.py the problem of getting the starting point of each individual square is handling by the functions `mapX` and `mapY`, then the problem of actually drawing the square is handled the function `drawSquare`.

```
# Returns the x pixel value for square on row index.
def mapX(index):
    return 300 + 50*index

# Returns the y pixel value for square on col index.
def mapY(index):
    return 100 + 50*index

# Draws one square with top left corner (x,y) side length side and fills it
# based on the boolean fill.
def drawSquare(x,y,side,fill):

    # Set the border based on the boolean variable fill.
    border = 2
    if fill:
        border = 0

    pygame.draw.rect(DISPLAYSURF, red, (x, y, side, side), border)
```