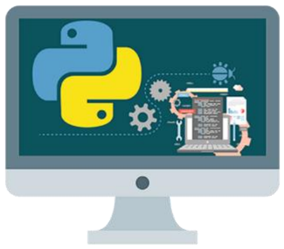
Creating Functions in Python



What are functions?

Functions are **little self-contained programs** that perform a specific task. As we've seen, python comes with many pre-made functions: `max()` `abs()` `sqr()`, **but you can also make your own**.

Example:

```
def happyBirthday(person):
    print("Happy Birthday to you!")
    print("Happy Birthday to you!")
    print("Happy Birthday, dear " + person + ".")
    print("Happy Birthday to you!")
```

We could then "**call**" our function above in a program above using ***any*** name we wanted:

```
happyBirthday('Emily')
happyBirthday('Jeff')
```

Why should we use functions (or make our own)?

You have probably noticed that you can create quite complex programs *without* dividing them up into functions, but there are several reasons why you should become familiar with how to create functions and how they work:

1. **Function are the standard way in which code is written** in most computer languages. Even if they aren't always necessary, lots of code you will see in lessons and examples on the internet will be written using functions. Most people make a habit of writing code using functions, so you have to become familiar with them.

2. ***Reusability.*** Once a function is defined, it can be ***used over and over and over again***. You can save functions or grab them from other programs and use them in new projects.

3. **Great way to divide up the work.** Complex programming projects involve lots people and many objectives. Using functions allows problems to be broken up into smaller specific tasks.

Look closer at the previous example:

Let's say you write a program that is designed to print out the "happy birthday" song to a friend:

```
print("Happy Birthday to you!")
print("Happy Birthday to you!")
print("Happy Birthday, dear Emily")
print("Happy Birthday to you!")
```

Perfect. We could also ***reuse*** this code again for *another* friend by simply cutting and pasting the code and replacing the word **`Emily`** with the **`Jeff`**

```
print("Happy Birthday to you!")
print("Happy Birthday to you!")
print("Happy Birthday, dear Jeff")
print("Happy Birthday to you!")
```

A more efficient way to do this in programming would be to create a **function**

Example:

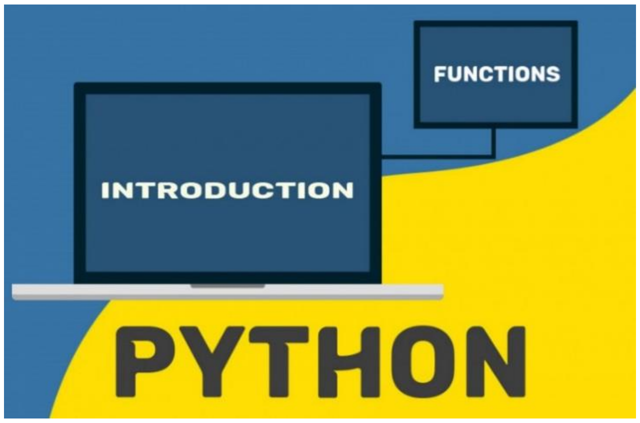
```
def happyBirthday(person):
    print("Happy Birthday to you!")
    print("Happy Birthday to you!")
    print("Happy Birthday, dear " + person + ".")
    print("Happy Birthday to you!")
```

We could then "**call**" the function in our program using **any** name we wanted:

```
happyBirthday('Emily')
happyBirthday('Jeff')
```

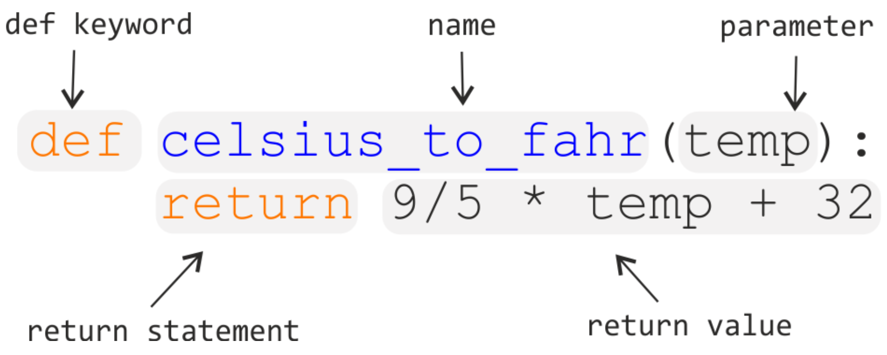
Exerciese#1

Type the function above into Trinket and then *call* the function with your name and a friend's name as the (person) **parameter.**

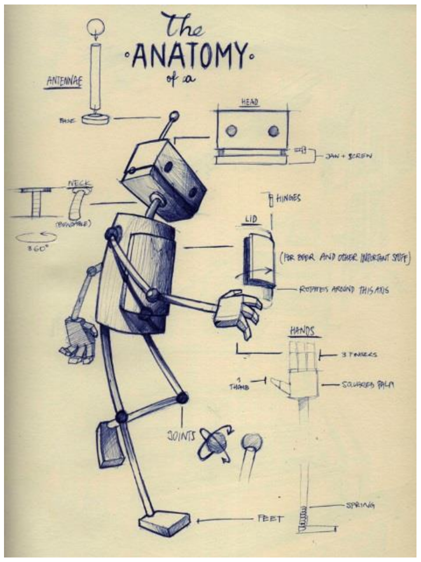


NOTE: ***parameters*** are values that can put into a function. In the example above (person) is required ***parameter.***

Anatomy of a function in python



```
def celsius_to_fahr(temp):
    return 9/5 * temp + 32
```



As you can see from the example above:

`def` – is the *keyword* that **tells python you are creating a function**.

`name` – the **name** of a function you use to *call* the function.

`Parameter` – information you need to give to the function.

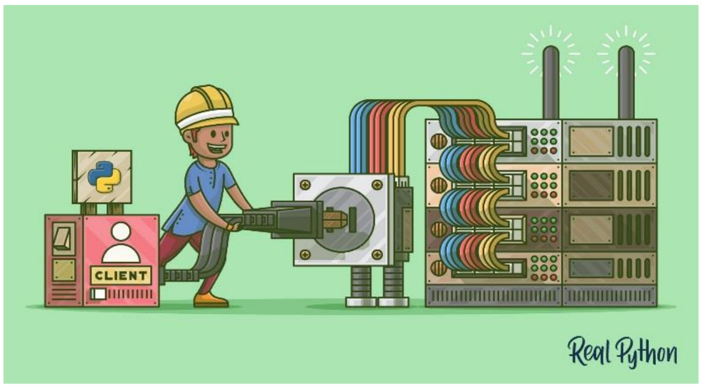
`Return statement` – code that output or *return* the desired value.

Exercise #2

Type in the following snippets of code to Trinket. Use the functions to determine the max value of numbers that you choose. Use the functions, repeatedly, by calling the functions with different **parameters.**

Function to find max of **2** numbers:

```
def max_of_two( x, y ):
    if x > y:
        return x
    return y
print(max_of_two(3,7))
```



Function to find max of **3** numbers:

```
def max_of_three( x, y, z ):
    return max_of_two( x, max_of_two( y, z ))

print(max_of_three(3, 6, -5))
```

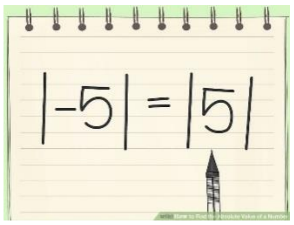
Notice here how we *"called"* the previous function *within* this new function. TRICKY!

Exercise 3

Type in the following snippets of code below to Trinket. "***Call***" the functions with appropriate ***parameters*** to see them work.

```
def absolute_value(num):

     if num >= 0:
          return num
     else:
          return -num
```



Note: this function is ***not*** necessary! Remember the built in function `abs()`

Dice throw:

```
import random

def roll_dice(sides):
     number = random.randint(1,sides)

     return(number)

sides = int(input("How many sides does the dice have?"))
throw = roll_dice(sides)

print(throw)
```

Exercise#4

Your turn….

a) Create a **function** that you can use repeatedly use that returns and prints the cube root of a number.

b) Create a **function** that converts inches to centimeters.

c) Create a **function** that can take a take someone's birthday and outputs how old they are.

Exercise#5

See if you can create a series of functions that make drawing basic shapes in python turtle easier.

create **at least 3** and get you classmates to do three **different** ones. **Share** your functions with each other and use the functions to draw some cool stuff.
Show Mr. Walzl when you are done.

Example:

Making a solid circle (disk)

```
import turtle as t

def draw_disk (col,size, x, y):


     t.penup()
     t.goto(x,y)
     t.pendown()
     t.color(col)
     t.begin_fill()
     t.circle(size)
     t.end_fill()

draw_disk('red',25,20,20)
```

Exercise #6

Write a **function** called `calculator(operation, num1, num2).`
When `calculator()` is called with an operation (add, subtract, divide, multiply) and two numbers, it will print out the answer needed.

Exercise #7

Using your brief experience using python, think of a task or section of code that you seem to use repeatedly. Write a **function** that you think *might* be useful to have on hand in the future.
You may start to build your own "library" of **functions** in a separate file that you can use on future projects.