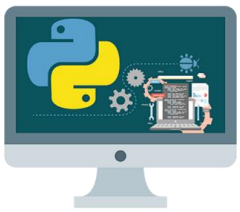
Creating Functions in Python



What are functions?

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

Example (enter both blocks of code below to see how a function works):

```
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 above in a program above using ***any person's*** name we wanted:

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

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. Using functions is the **standard** way in which code is written in most coding languages. Even if functions aren't always necessary, much 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 must 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.

More examples of functions:

Let's say you write a program that is designed to find out if a number is odd or even:

```
x=int(input("give me a number"))
if x%2 == 0:
        print("even")
    else:
        print("odd")
```

Perfect. If we want to ***reuse*** this code again and again we could create a **function**

```
def evenOdd( x ):
    if x%2 == 0:
        print("even")
    else:
        print("odd")
```

Once a function is defined, we can "**call**" the function in any time in our program and use if over and over again without repeating the original code:

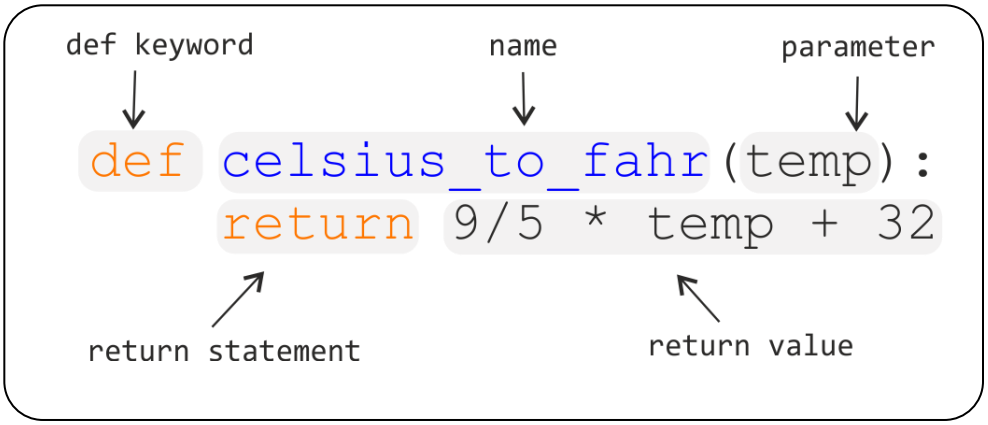
```
evenOdd(2)
evenOdd(3)
evenOdd(9)
evenOdd(8)
evenOdd(14)
evenOdd(23)
```

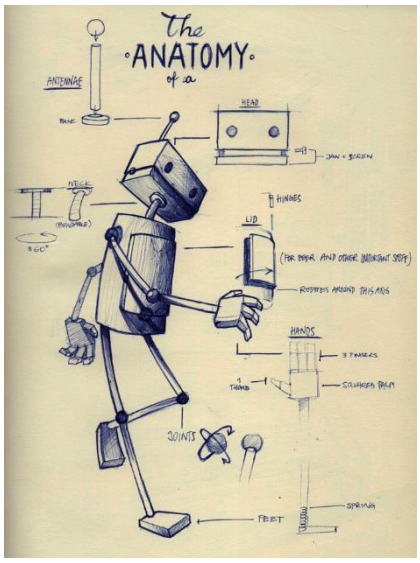
Exerciese#1

Type the function above into Trinket and then *call* the function with **any number** you wish name as the function's **parameter.**

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

Anatomy of a function in python





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 outputs or *returns* the desired value.

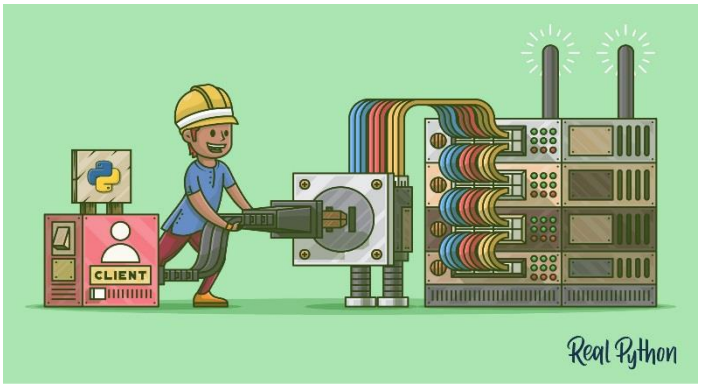
Exercise #2

Type in the following snippets of code to Trinket. Use the functions to determine the max value of numbers that you choose. Use the function, 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))
print(max_of_two(2,10))
print(max_of_two(11,7))
print(max_of_two(3,7))
```

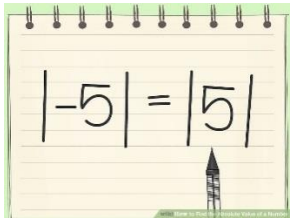


Exercise 3

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: there is actually a built in function in python that calculates absolute value…do you 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 converts inches to centimeters.

b) Create a function that can convert Miles/hour into Km/hour.

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

d) Create a **function** that you can use to return and print the **cube root** of a number.

The cube root of 64 equals 4,

$\sqrt[3]{64} = 4$ $\quad y = \sqrt[3]{x} \quad y = x^{\frac{1}{3}}$

Exercise#5

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 circular (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.

```
calculator(multiply,8 ,2).
16
```

Exercise #7

Create a **function** that asks the user for a number and then appends that number to a list. Once you have built your function put it in a while loop that repeatedly **calls function.** Your loop should end when the person types 0 as an input value and then print out the list
No parameter will be necessary for your function. `def ask_num ():`

Exercise #7

Imagine you are tasked with creating a program to simulate the duties of a robotic car sales person. The robot has various functions and features that need to be implemented. Your goal is to create a Python program that incorporates user-defined functions for different aspects of the robot. Your program should have the following functions:

1. A function called **greet_customer** that takes the customer's name as input and prints a personalized message welcoming them to the auto dealership.

2. A function called **car_preference** that asks the customer what kind of car they are looking for and then has a series of replies based on their answer.

3. A function called **budget** that asks the customer their budget and then can list cars that fit are under the number of dollars they are willing to spend.

Create a program that can call and repeat the above functions in any order or sequence that you think makes most sense for a believable simulated dialog.

Exercise#8 School Grades

Create a program that helps students in grade 12 get ready for graduation. First create several predefined **lists** of grades:

```
Kevin = [76, 88,90,91,89]
Julie = [70, 98,71,81,77]
Carrol = [60, 72,95,64,57]
```

Then create 3 user-defined functions that do the following:

1. The **calculate_average** function calculates the average score for each student.
2. The **calculate_grade** function assigns a letter grade based on the average score.
3. The **Scolarship** function that determines if their average is above a certain grade to get a scholarship.

Each function should have a list as an input

Create a program that incorporates the user-defined lists above that guides them through selecting the functions above.

Example:

```
Hi! Welcome to Graduation Ready WSS App! Enter your name: kevin.

Hi “Kevin” how can I help you today?
press 1 calculate your average.
press 2 to find out your letter grade average.
press 3 to find out if you qualify for a scholarship.
```

Important Note:

It may not seem obvious or terribly useful to user function at this stage in programming. But using user-defined functions, you can break up code, making it more readable, reusable, and easier to maintain. Creating functions that have specific responsibilities, help create a clean and organized structure for the program. This approach is especially valuable when dealing with more complex programs or when multiple aspects of the program need to be reused or modified independently.

Functions…you gotta know em!