# `def`-ining a function

- A function as an execution control structure
- `def`-ining a new function
- Passing and using function parameters
- `return`-ing a value from a function

# Function calls and execution control

- A function call is another kind of statement that alters the order in which statements in a program are executed
- A function call creates a temporary ‘excursion,’ causing the sequence of execution to jump to the named function. When the function exits, execution returns to the point where the function call occurred.
- We've already seen these built-in function calls:
  - len(), type(), min(), max(), str(), bool(), …
- Today we are going to learn
  - how to write our own functions
  - how to specify what parameters are passed to our functions
  - how to return something from a function

# Defining a new function

We have seen these calls to built-in functions:

- `abs(),max(),len(),sum(),print()`

```
>>> abs(-9)
9
>>> max(2, 4)
4
>>> lst = [2,3,4,5]
>>> len(lst)
4
>>> sum(lst)
14
>>> print()

>>> def iSquaredPlus10(x):
        result = x**2 + 10
        return result

>>> iSquaredPlus10(1)
11
>>> iSquaredPlus10(3)
19
>>> iSquaredPlus10(0)
10
```

You can define a new function using `def`

`def`: function definition keyword

`iSquaredPlus10`: name of function

`x`: variable name for input argument

```
def iSquaredPlus10(x):
    result = x**2 + 10
    return result
```

`return`: specifies the value that the function returns (default: None)

# print() versus return

**return** returns execution to the point where the function is called. The returned value becomes the value of the function call.

```
def iSquaredPlus10(x):
    result = x**2 + 10
    return result

squarePlusTen = iSquaredPlus10(5)
print(squarePlusTen)
```

The value calculated by the function is returned, assigned to a variable, and then printed to the screen.

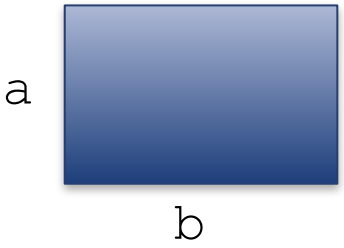
```
def iSquaredPlus10 (x):
    result = x**2 + 10
    print(result)
```

The function `print`s the value of result to the screen. Because the function does not explicitly return a value, the default value None is returned.

# Defining a new function

The format of a function `def`inition is

```
def <function name> (0 or more parameter names>):
    <indented body of the function>
```




Let's write a function `areaOfRectangle()` that:

- Takes two numbers as parameters (side lengths a and b of a rectangle )
- Returns the area of the rectangle

```
def areaOfRectangle(a, b):
    area = a*b
    return area
```

```
>>> areaOfRectangle(3,4)
12
>>>
```

# Exercise

Write function `hello` that:

- takes a name (i.e., a string) as input
- prints a personalized welcome message

(Note: because `hello` does not explicitly return anything, the default value None is returned

```
>>> hello('Julie')
Welcome, Julie, to the world of Python.

>>> print(hello('Julie'))
Welcome, Julie, to the world of Python.
None
>>>
```

```
def hello(name):
    line = 'Welcome, ' + name + ', to the world of Python.'
    print(line)
```

# Exercise

Write function `oddCount()` that:

- takes a list of numbers as input
- returns the number of odd numbers in the list
- here is an example of input and output for `oddCount`
- Hint: you are going to use both a `for` statement and an `if` statment

```
>>> numList = [4, 0, 1, -2]
>>> oddCount(numList)
1
>>>
```

# Exercise

Write function `oddCount()` that:

- takes a list of numbers as input
- returns the number of odd numbers in the list

```
def oddCount(aNumList):
    result = 0
    for i in aNumList:
        if i%2 == 1:
            result += 1
    return result
```

```
>>> numList = [4, 0, 1, -2]
>>> oddCount(numList)
1
>>>
```

# Commenting a function

A program should be documented so that:

- the program's 'mission' is explained
- the developer who writes/maintains the code understands it

Python has built-in support for documenting each function with a docstring. The docstring should

- appear immediately after the `def` line
- have an imperative style ('Do this!')
- make clear the function input and output

Example docstring

```
def iSquaredPlus10(i):
    '''return i squared plus 10'''
    result = i**2 + 10
    return result
```