Google igniteCS

An Introduction to Python

Part II: Learning the Basics

Introduction

- **Mentors:** Mike & Andrew
- **Goals**
    - Review Lecture 1
    - User-Defined Functions
    - Useful Packages
    - Lists
    - For Loops
    - Post-Lecture Activities
- **Any Questions Before We Start?**

Review Lecture 1 Activities

- **`palindrome.py`**
- Pull up your own code to compare!
- Want to look over another activity?

`palindrome.py`

1. **Run code from the Terminal**
   - Confirm functionality
2. **Open up IDLE and inspect the code**
   - Taking input from the user
   - Storing and manipulating variables
   - While loops
   - Built-in String functions
   - Control Flow (if...else statements)

Any Questions Before We Step Into New Territory?

User-Defined Functions (UDF’s)

Premise

- We want to repeat the same operations
- How should we do this?

```
# squarePlus10 n1
n1 = (n1 * n1) + 10

# squarePlus10 n2
???

# squarePlus10 n3
???
```

First Approach: Copy & Paste

2 Major Flaws:

1. Tedious to read
2. Inefficient editing requirements

```
# square_plus_10 n1
n1 = (n1 * n1) + 10

# square_plus_10 n2
n2 = (n2 * n2) + 10

# square_plus_10 n3
n3 = (n3 * n3) + 10
```

Second Approach: Generalizing

This is the first step to creating User-Defined Functions!

```
# square_plus_10 n1
n1 = (n1 * n1) + 10

# square_plus_10 n2
n2 = (n2 * n2) + 10

# square_plus_10 n3
n3 = (n3 * n3) + 10
```

Let’s review what a function is before getting ahead of ourselves.

Functions Review

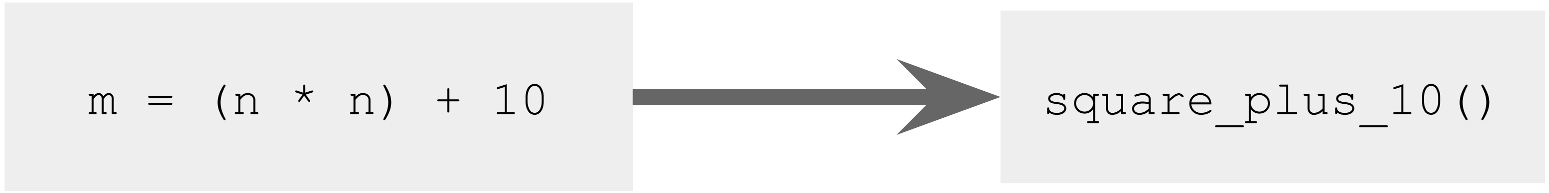
A function takes in input and then returns an output after performing operations on the original input.

$$\begin{tikzcd}\mathbf{4} \arrow[r] & \boxed{\texttt{square}} \arrow[r] & \mathbf{16}\end{tikzcd}$$

***Note:** not all functions in programming return an output, sometimes they just perform a procedure, like the **`print()`** function.

Functions Review

- We’ve used Built-in Functions
    - **`len()`**, **`print()`**, **`input()`**
- Now we can define our own functions!

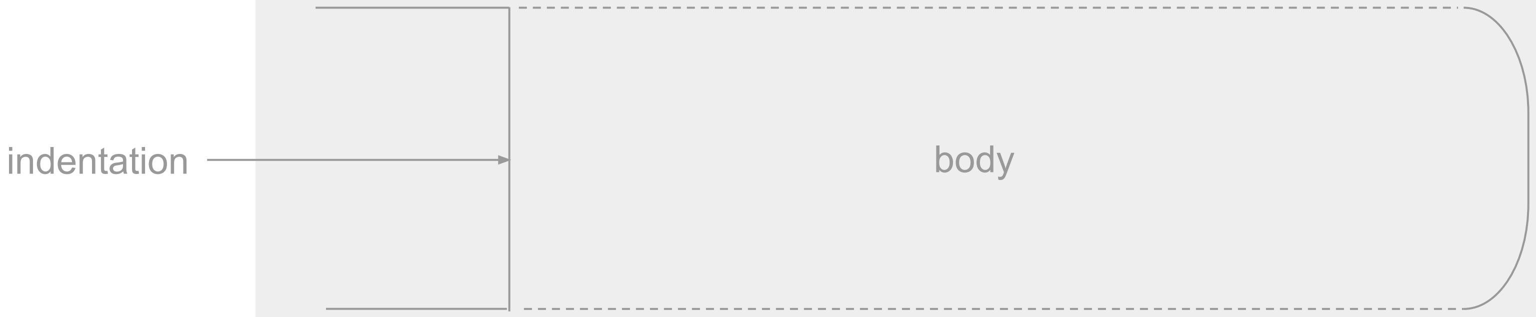


```
m = (n * n) + 10
```

```
square_plus_10()
```

User-Defined Functions: An Outline

```
# comment describing this function
def function(param1, param2, ...) :
```



So Let’s Make A Function!

```
# square_plus_10 n1
n1 = (n1 * n1) + 10

# square_plus_10 n2
n2 = (n2 * n2) + 10

# square_plus_10 n3
n3 = (n3 * n3) + 10
```

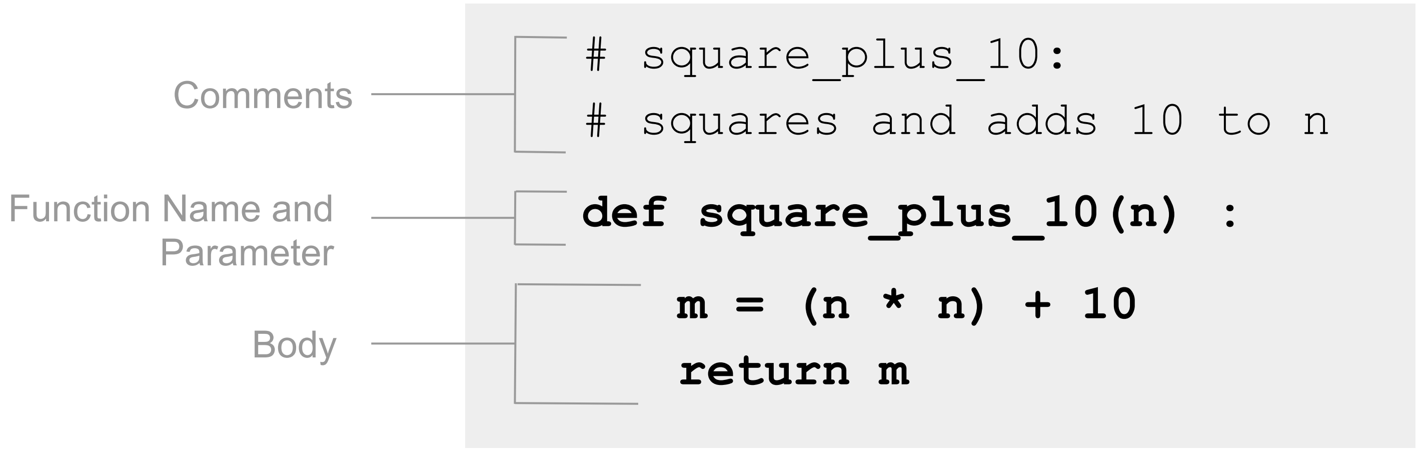
```
# square_plus_10:
# squares and adds 10 to n

def square_plus_10(n) :

    m = (n * n) + 10
    return m
```

*We’ve generalized these operations into a function

Our Function’s Outline



```
# square_plus_10:
# squares and adds 10 to n
def square_plus_10(n) :
    m = (n * n) + 10
    return m
```

***Parameter:** variable(s) that we pass into a function

Calling Our User-Defined Function

```
>> square_plus_10(3)
19
```

```
>> square_plus_10(2)
14
```

```
>> square_plus_10(1)
11
```

Try it yourself!

Function Exercise

1. Create a new file called **`repeatString.py`** by typing:

   ```
   idle3 repeatString.py
   ```

2. Complete the following exercise in this file:

   a. `Define a function that:`
      - `Takes in a string` **`s`** `and a number` **`n`**
      - `Returns string` **`s`** `repeated` **`n`** `times`

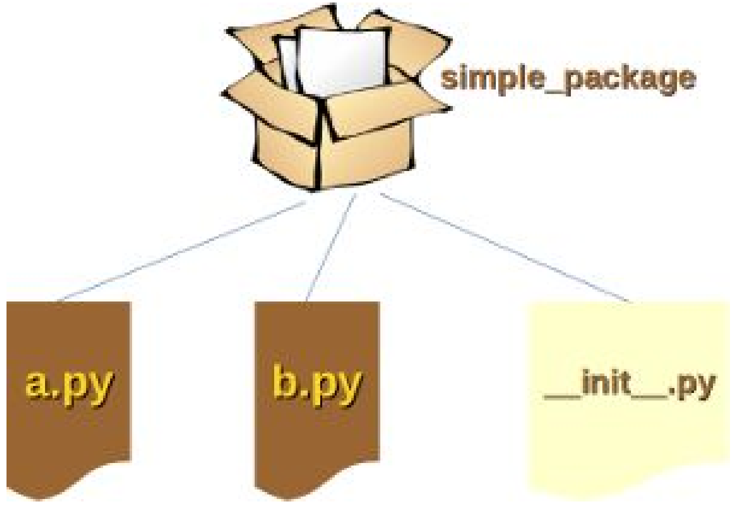
   b. `Calls the function as many times as you want`

3. Run the file in terminal: **`python3 repeatString.py`**

Math and Random Packages

What are Packages?

- A collection of Python files
- Each file contains additional functions and variables that altogether can be useful for a certain task
- We don’t have to re-invent the wheel thanks to packages!



General Guidelines for Using Packages

To import a package:

```
import package
```

To access a pre-defined value:

```
package.value
```

To access a function:

```
package.function(n)
```

The Math Package

- **`fabs()`** is a function in the Math Package
- Returns the absolute value

```
# import package
import math

# print |n|
n = -3.1415
print(math.fabs(n))
```

```
Output:
3.1415
```

Some Useful Math Functions

- **Documentation: HERE**
- **Numerical Functions:**
  - fabs(n)
  - factorial(x)
  - sqrt(x)
- **Trig Functions:**
  - sin(x), cos(x), tan(x)
  - asin(x), acos(x), atan(x)
- **Angle Conversion Functions:**
  - degrees(input_radians)
  - radians(input_degrees)

```
>>> math.fabs(-6.66)
6.66
>>> math.factorial(4)
24
>>> math.sqrt(25)
5.0
>>> math.sin(math.pi/2)
1.0
>>> math.radians(180)
3.141592…
>>>
math.degrees(3.141592)
179.999…
```

A Random Package Example

- **Random number generator in the Random Package**
- **Documentation HERE**

```
import random
max = 10
print(random.randint(1, max))
```

```
Sample Output:
8 # run 1
1 # run 2
7 # run 3
```

Try it yourself!

Math & Random Package Exercise

1. Create a new file called **`randomPrint.py`** by typing:

   `idle3 randomPrint.py`

2. Complete the following exercise in this file:
   a. `Import the random package`
   b. `Get a string from the user`
   c. `Print out that string a random number of times`
   d. `Set the min and max number of repeats`
3. Run the file in terminal: **`python3 randomPrint.py`**

Summary

- **User-Defined Functions**
  - How to define your own functions
- **Packages**
  - We can import more useful functions and variables to use
  - The Math and Random Packages
  - We don’t always have to reinvent the wheel!

Any Questions?