COMP1021
Introduction to Computer Science
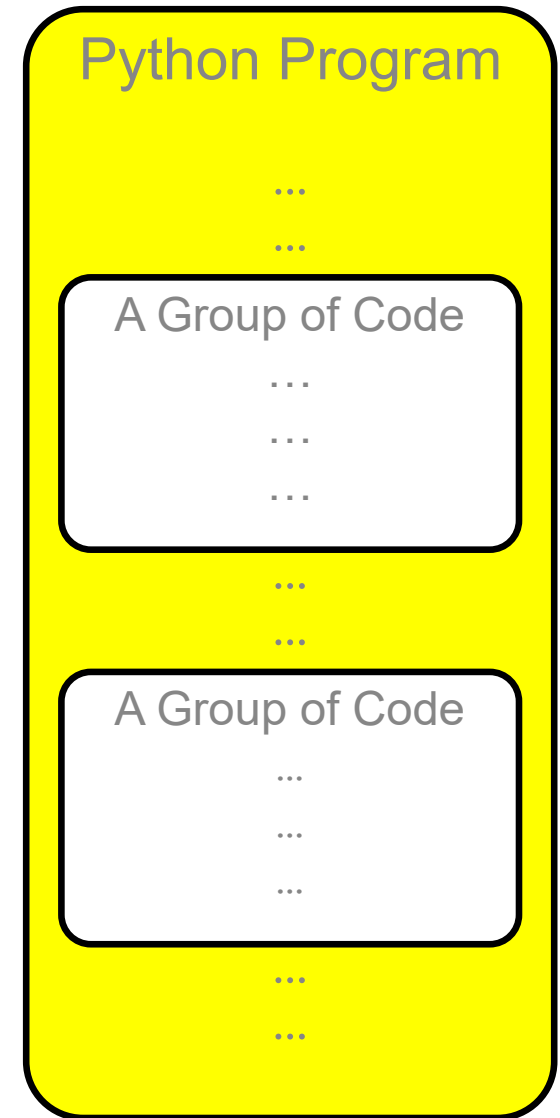
Functions

David Rossiter and Gibson Lam

Outcomes

- After completing this presentation, you are expected to be able to:

    1. Define and use a function in Python

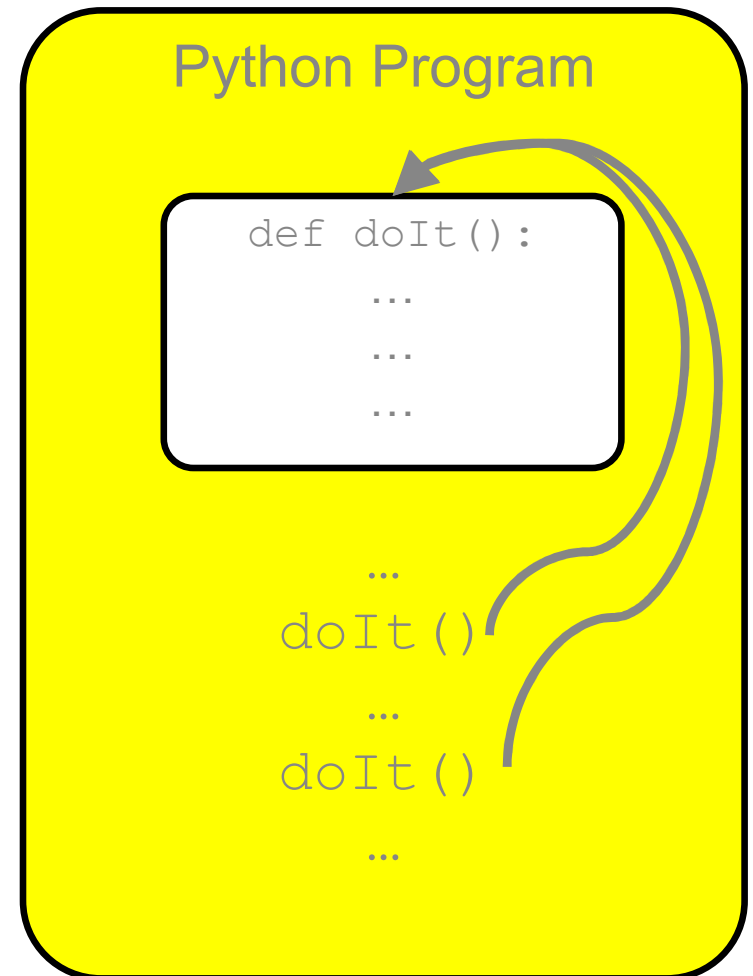   2. Pass a value to a function, and use the value appropriately

Running a Group of Code

- Sometimes you may want to put the same group of code in different places in your program

- To do that, one way is to copy and paste the same code into those places inside the program

- However, the program will become very long and contain a lot of duplicated code

Python Program

...
...

A Group of Code

...

...

...

...
...

A Group of Code

...

...

...

...
...

Functions

- Instead of copying and pasting the group of code everywhere, the group of code is first put inside a *function*

- You can then use the function as many times as you like in appropriate places inside the program

Python Program

```
def doIt():
    ...
    ...
    ...
```

```
...
doIt()
...
doIt()
...
```

Functions You Have Already Used

- We have already used a lot of different functions in the course

- For example, `print()`, `input()` and `turtle.forward()` are all functions that we have used before

- These are functions made by others, i.e. the people who made the Python language

- In this presentation, we will look at making our own functions and then using them

Defining a Function

- To make a function in Python, we use the def command (**def**ine a function)

- Here is an example:

This is the name of the function (you need to put parentheses after the name)

This is the code of the function

```
def greeting():
    name = input("What is your name? ")
    print("Welcome " + name + "!")
```

- When we define a function, we need to give it a name

- We will refer to this name when we want to use the function later

Using a Function

- To use the function we have defined in the previous slide, we simply run it using its name, like this:

```
def greeting():
    name = input("What is your name? ")
    print("Welcome " + name + "!")


print ("I am going to ask you a question...")
greeting()
```

The function we defined before

The function is used here (again, you need to put parentheses after the name)

```
I am going to ask you a question...
What is your name? Dave
Welcome Dave!
```

Defining and then Using Functions

- When you make functions you have to make sure that you define them before you use them

- If you don't, Python will give you an error, e.g.:

```
print ("I am going to ask you a question...")
greeting()


def greeting():
    name = input("What is your name? ")
    print("Welcome " + name + "!")
```

*Here the program tries to use the function **before** it is defined, which is not OK!*

```
I am going to ask you a question...
Traceback (most recent call last):
  File "C:\greeting.py", line 2, in <module>
    greeting()
NameError: name 'greeting' is not defined
```

Using a Function Multiple Times

- You can run a function as many times as you like

- For example, we can run a function three times in different places:

```
def response():
    print("Very good!")
```

```
Is it a good course?
Very good!
Is the instructor good?
Very good!
Do I look good?
Very good!
```

```
print("Is it a good course?")
response()
print("Is the instructor good?")
response()
print("Do I look good?")
response()
```

Passing a Value to a Function

- Sometimes it is useful to give a value to a function so that it can do different things

- We call that 'passing values to a function' in computer science terms

- Here is an example:

In this example, the function is expected to receive a value, stored in a variable called 'name'

```
def show_response( name ):
    if name == "Dave":
        print("What a good name!")
    else:
        print("How are you?")
```

Using the Function

- You can pass a value directly to the function

```
show_response("Estelle")
How are you?
```

```
show_response("Dave")
What a good name!
```

- Sometimes the value that you pass to the function is first stored in a variable, like this:

```
name = input("What is your name? ")
show_response(name)
```

- Both approaches are common