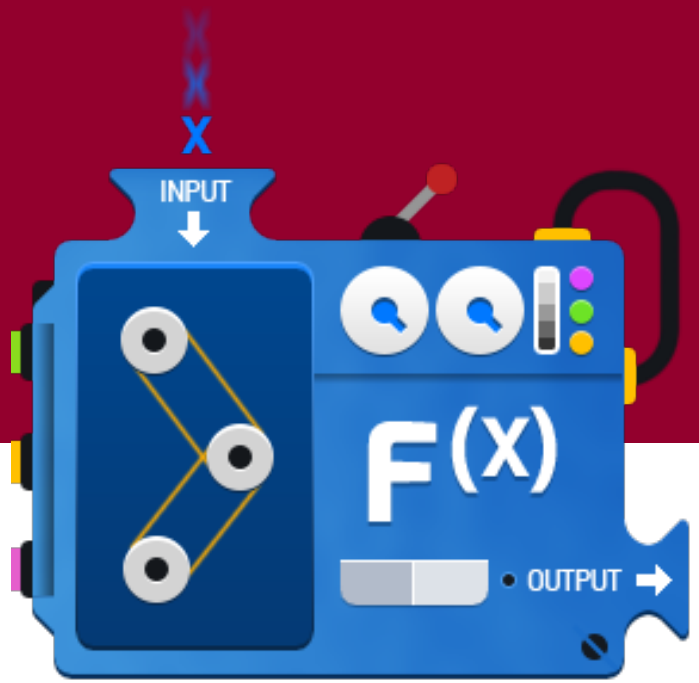
CEBD 1100 Introduction to Data Analysis and Python

Functions



Functions

Functions in Python

Function Definition

- Functions are defined with the "def" prefix.

```
def my_function1():
  print("Hello from a function")

def my_function2():
  return "Hello from a function"
```

A function that prints something from within the function.

A function that returns a result.

Calling a Function

- Call it by using its name, with empty brackets.
- The brackets can be used to provide information to the function, later we will show how this is done.

```
def my_function():
  print("Hello from a function")

my_function()
```

Passing Parameters to a function

- To pass in parameters, just supply the value between the brackets.

```
def my_function(fname):
  print("Hello" + fname)

my_function("Brendan")
```

Passing Parameters to a function

- To pass in parameters, just supply the value between the brackets.
- Don't forget, order matters in the positioning of arguments.

```
def my_function(fname, lname):
  print("Hello" + fname + " " + lname)

my_function("Brendan", "Wood")
```

Passing Parameters to a function

- To pass in parameters, just supply the value between the brackets.

```
def my_function(fname):
  print("Hello" + fname)

a = "Mary"
my_function(a)
```

Exercise 1
(10 minutes)

- Write a method to find a maximum between 3 integers with the following signature:

```
def get_max(n1, n2, n3):
```

- The function should return the largest number back to the user.
- This can be done in two ways, by using a “list” to help you, or by manually comparing values.

Default Value

```
def my_function(fname = "Unknown"):
  print("Hello" + fname)
```

Try the following function calls;

- my_function("Joe")
- my_function()

Returning a Value

```
def isnumbernegative(n):
    if n < 0:
        return True
    return False
```

- Using the return statement we can return a value to the caller.

```
if isnumbernegative(-100):
    print("Error:  Age can't be negative")
```

Exercise 2
(5 minutes)

- Write a function that gets a number as its parameter and then returns True if the number is divisible by 3 otherwise it will return False.
- Use your function to prompt a number from the user and print if the number is divisible by 3 or not.
- *Pro Tip*: As a developer, you want to make your function as flexible as possible. Try to find a way NOT to hard code the number “3” in the function.

provide a number: **15**

15 is divisible by 3

Supplying Less Parameters than Required

- Given:

```
def describe_pet(pet_name='spot', animal_type='dog'):
```

- What if I only want to call this function with animal_type?

```
describe_pet(animal_type='dolphin')
```

Discussion

- Given:

```
def describe_pet(pet_name, animal_type='dog'):

describe_pet(animal_type='dolphin')
```

- What happens?

Exercise 3
(5 minutes)

Create a function that adds two numbers and returns the value.

Use your function in your code, in some scenario.

Exercise 4
(10 minutes)

1. Create a function that received a list as a parameter. The function should print the list items one over the other.
2. Modify the previous function to get a title, and use the supplied title as the title of your list.

Exercise 5
(10 minutes)

Make a function that takes up to 3 arguments and returns a list of 1, 2 or 3 items (an array).

Strongly typing a function signature.

- To force a function to accept parameters (arguments) which are of a specific type (to ensure you choose the type yourself), put a hint in the function signature like this:

def addintegers(n : int, m : int)

def sayhello (first : str, last : str = “Unknown”)

Also, when you apply a default, Python assumes that you are specifying a type. For example, this function accepts integers only:

def addnumber(num = 0)

Making function packages and importing them

- Your functions can be put into another file and accessed from your main program.
- We use the “import .. as” or “import” to retrieve the file when necessary.
- There are no limits to how many times you can import the file.
- What are the benefits?
  - Re-use: We can reuse the same functions between ALL your programs in your project (some Python projects can be composed of many files).
  - Clean code: With the functions removed, you have much less code on the screen to work with, and it’s easier to manage.
  - Testability: We didn’t cover this yet, but you can test your functions much easier when they are in their own files (independent).

Example of Packages

- Filename : “mathfunctions.py”.
- Content:

```
def myadder(a = 0, b = 0)
    return a + b
```

- Filename “myprogram”.
- Content:

```
import mathfunctions as mf
Print(“Sum of 4 and 5 is : “ + mf.myadder(4,5)
```

Quick Review of Functions

1. If we want a function to determine if a number is even or odd, do we print “The number is even/odd” from the function itself? Explain.
2. Should we look for areas in the function where a “return” CANNOT be reached? Why?
3. If a function that generates an aggregate calculation like the average of some numbers, maybe 2, maybe 3… maybe 99, how do we input these into the function?
4. If a function ultimately returns nothing (nothing found), what can you return? If expecting and integer? If expecting a string? A list? An object of any type?
5. How to define a default value for an argument?
6. How to specify which argument implicitly when you call the function itself?
7. Why does this not work: `def describe_pet(pet_name, animal_type='dog'):`

Homework

- Make a function that will print out a row of asterisks depending on the value given to it.
- Use this in a loop – so we can print a triangle of stars.

```
*
**
***
```

UNIVERSITÉ Concordia UNIVERSITY

CONCORDIA.CA