ISLab Python Course

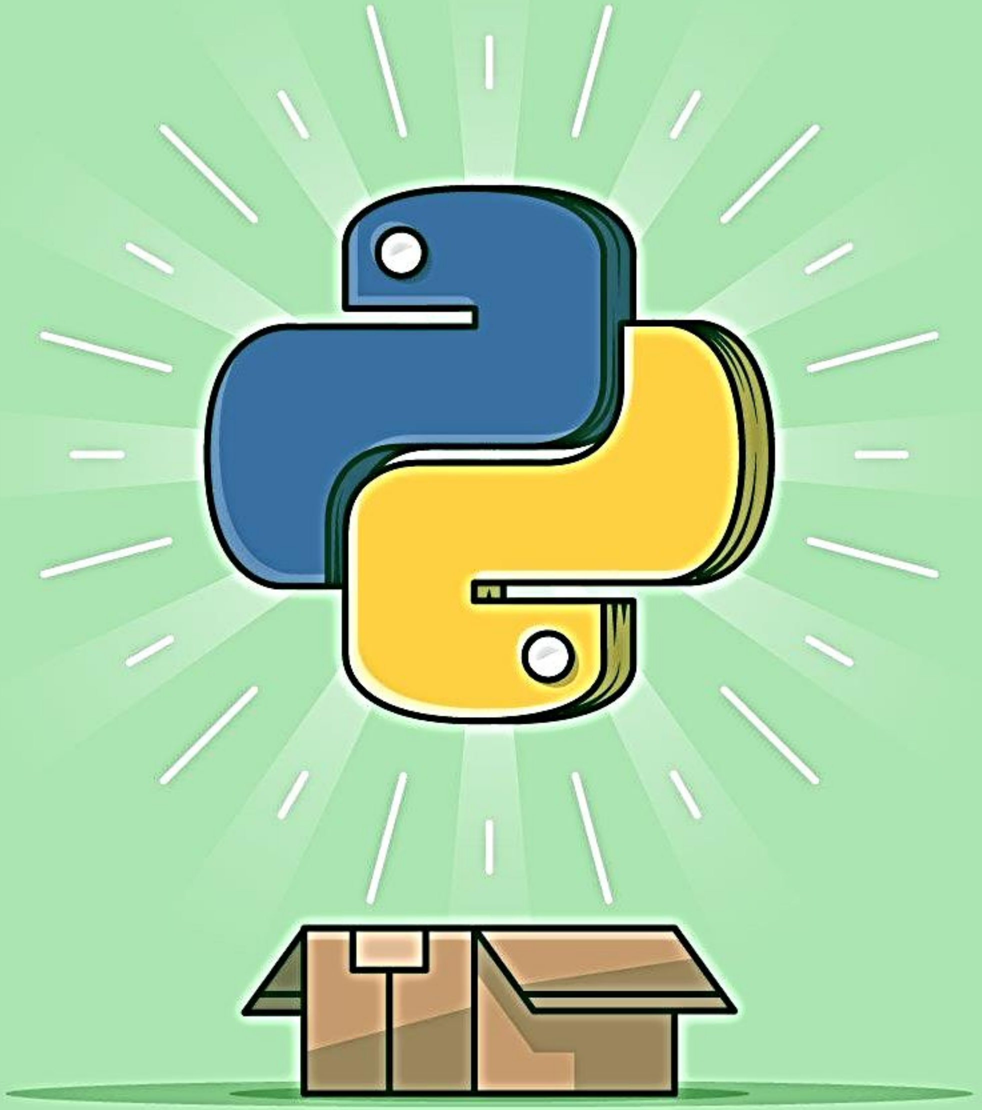
Session 5: Functions in Python

Presenters:

Shahrzad Shashaani

Hamed Homaei Rad

Saeed Samimi



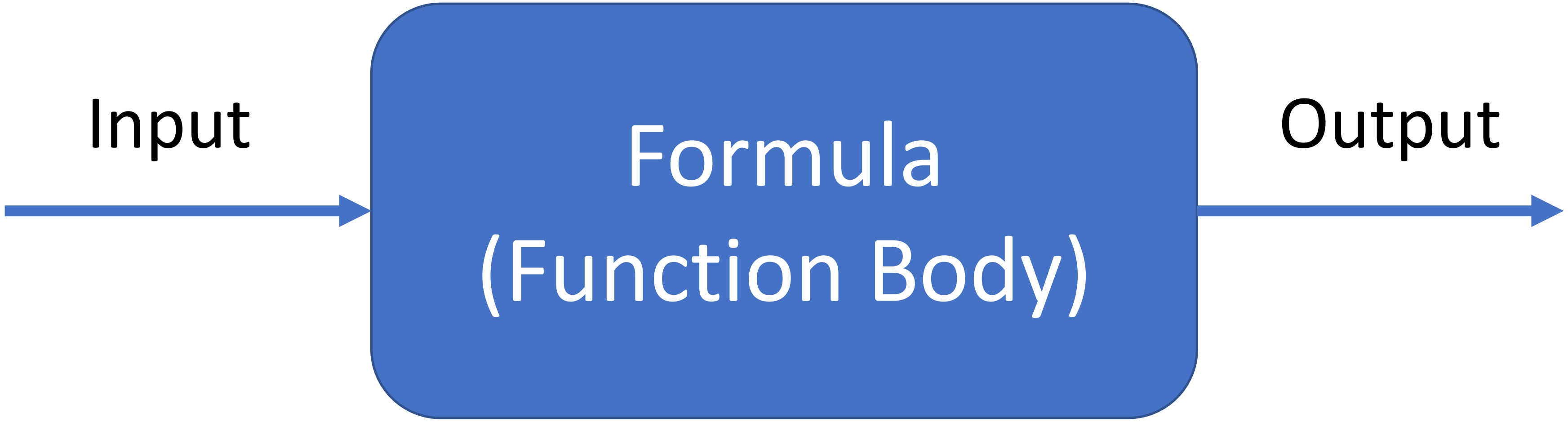
Summer 2023

K.N.Toosi University of Technology

Functions

$$\underbrace{f(\overset{\overset{input}{\uparrow}}{x})}_{Y\,,\,output} = \underbrace{x^{2} + 3}_{Formula}$$

Functions



Functions

- A block of code which only runs when called
- You can pass data, known as parameters, into a function
- A function can return data as a result
- Parameter names in function definition are optional
- We can define a default value for arguments so that if no argument is passed, the default value is used by function

```
def  fn_name(input1, input2):
    do_sth()
    do_sth_else()
    return sth
```

Parameters or Arguments?

The terms **parameter** and **argument** can be used for the same thing:

information that are passed into a function

From a function's perspective:

- A parameter is the variable listed inside the parentheses in the function definition
- An argument is the value that are sent to the function when it is called

Variable Scopes

- Scope Definition
  - Part of a program where the name binding is valid
  - In another words: where the name can be used to refer to the entity
  - When you make an assignment to a variable in a scope (e.g. the function scope), that variable becomes local to that scope

- Execution Order

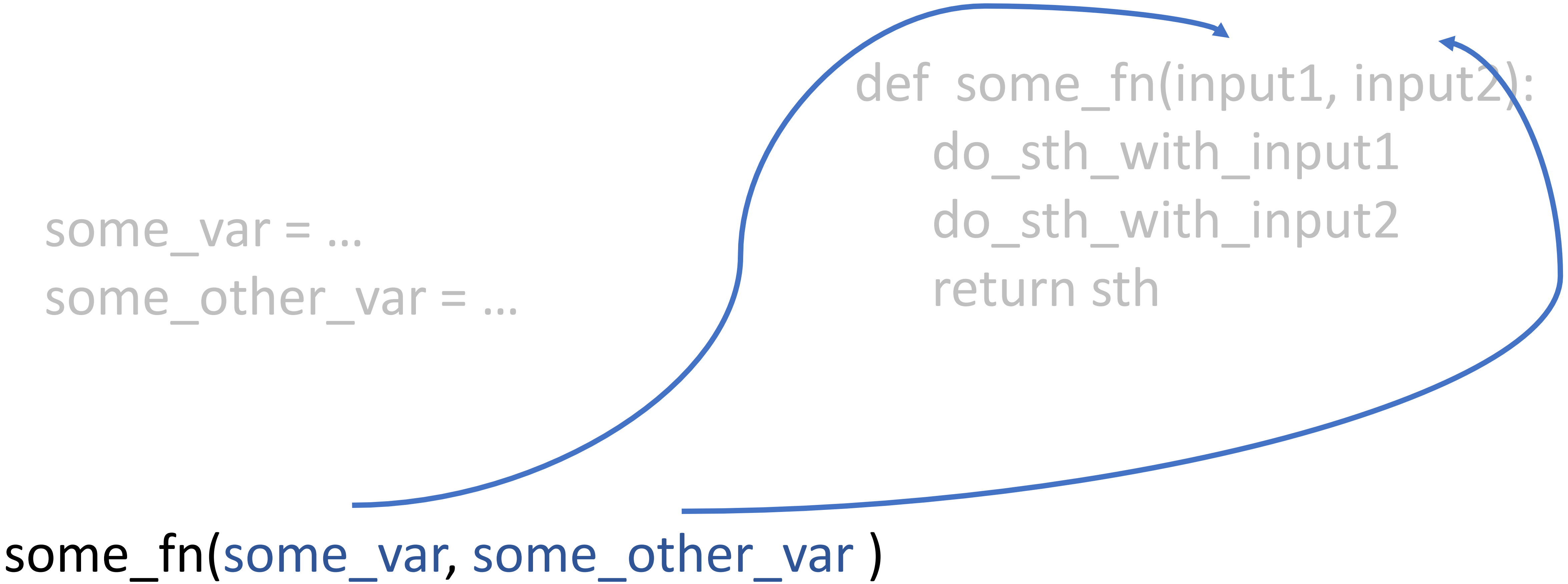
```
variable_1 = 5*a
```

Call by Value / Reference

```
some_var = …
some_other_var = …
```

```
def  some_fn(input1, input2):
     do_sth_with_input1
     do_sth_with_input2
     return sth
```

Call by Value / Reference



```
def  some_fn(input1, input2):
    do_sth_with_input1
    do_sth_with_input2
    return sth
```

```
some_var = …
some_other_var = …
```

```
some_fn(some_var, some_other_var )
```

Call by Value / Reference

- **Are Python function arguments considered call by reference or call by value?**
  - Short answer is: **both!**

- Python follows the idea of **"Call by Object Reference"** or **"Call by Assignment"**
- If you pass **immutable** objects such as Numbers, Strings or Tuples to a function, the passing is treated as **"call by value"**, since it is not possible to change the value of those objects.
- On the other hand, passing **mutable** objects such as Lists and Dictionaries can be regarded as **"call by reference"**. Changing their values inside the function results in a change on the main object, since we have a reference type of call for mutable objects in Python.

- So, **be careful** what you pass in, and what you do with passed-in values inside a function