PYTHON PROGRAMMING

Lecture 1 Unit 2

FUNCTIONS

Name: Mr. Dheeraj Sundaragiri
Assistant Professor
Department of Computer Science and Engineering
SNIST

UNIT 2 - CONTENTS

FUNCTIONS

- Defining a function
- Calling a function
- Types of functions
- Function Arguments
- Anonymous functions
- Global and local variables

DEFINING A FUNCTION

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

Creating a Function

DEFINING A FUNCTION

In Python a function is defined using the `def` keyword.
Syntax:

```
def function_name(parameters):
        """docstring"""
        statement(s)
```

CALLING A FUNCTION

To call a function, use the function name followed by parenthesis:

Example

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

my_function()
```

$\longleftarrow$ Calling Function

TYPES OF FUNCTIONS

Functions are of two types:

1. **Built-in functions** - Functions that are built into Python.
2. **User-defined functions** - Functions defined by the users themselves.

FUNCTION ARGUMENTS

In Python, user-defined functions can take four different types of arguments.

1. **Default arguments**
2. **Required arguments**
3. **Keyword arguments**
4. **Arbitrary arguments**

FUNCTION ARGUMENTS

1. Default arguments

Function definition

```
    def defaultArg( name, msg = "Hello!"):
```

Function call

```
defaultArg( name)
```

FUNCTION ARGUMENTS

1. Default arguments - Example

```
def defaultArg(name, msg = "Hello!"):
    print("Hello",name + ', ' + msg)
defaultArg("Alice")
defaultArg("Bob","Good Morning!")
```

OUTPUT:

```
Hello Alice, Hello!
Hello Bob, Good Morning!
```

FUNCTION ARGUMENTS

2. Required arguments

Function definition

```
    def requiredArg (str,num):
```

Function call

```
requiredArg ("Hello",12)
```

FUNCTION ARGUMENTS

2. Required arguments - Example

```
def requiredArg(name, msg):
    print("Hello",name + ', ' + msg)
requiredArg("Bob","Good Morning!")
requiredArg("Alice")
```

OUTPUT:
```
Hello Bob, Good Morning!
TypeError
```

FUNCTION ARGUMENTS

3. Keyword arguments

Function definition

```
def keywordArg( name, msg ):
```

Function call

```
keywordArg( name = "Alice", msg = "hi")
keywordArg( msg = "hello", name = "Bob")
```

FUNCTION ARGUMENTS

3. Keyword arguments - Example

```
def keywordArg(name, msg):
    print("Hello",name + ', ' + msg)
keywordArg( name = "Alice", msg = "hi")
keywordArg( msg = "hello", name = "Bob")
```

OUTPUT:
```
Hello Alice, hi
Hello Bob, hello
```

FUNCTION ARGUMENTS

4. Arbitrary arguments

Function definition

```
def varlengthArgs(*vArgs):
```

Function call

```
varlengthArgs(10,20,30,40)
```

FUNCTION ARGUMENTS

4. Arbitrary arguments - Example

```
def varlengthArgs(*vArgs):
    for i in vArgs:
        print("Hello, My age is ", i)
varlengthArgs(10,20,30,40)
```

OUTPUT:

```
Hello, My age is  10
Hello, My age is  20
Hello, My age is  30
Hello, My age is  40
```

ANONYMOUS FUNCTION

- In Python, anonymous function is a function that is defined without a name.
- While normal functions are defined using the def keyword, in Python anonymous functions are defined using the lambda keyword.
- Hence, anonymous functions are also called **lambda functions.**

ANONYMOUS FUNCTION

SYNTAX

```
lambda arguments: expression
```

EXAMPLE

```
double = lambda x: x * 2

print(double(5))
```

OUTPUT: 10

GLOBAL AND LOCAL VARIABLES

Example 1: Create a Global Variable

```
x = "global"
def foo():
    print("x inside :", x)
foo()
print("x outside:", x)
```

OUTPUT:

```
x inside : global
x outside: global
```

GLOBAL AND LOCAL VARIABLES

Example 2: UnboundLocalError

```
x = "global"
def foo():
    x = x * 2
    print(x)
foo()
```

OUTPUT:
```
UnboundLocalError: local variable 'x' referenced before assignment
```

GLOBAL AND LOCAL VARIABLES

Example 3: global keyword

```
x = "global"
def foo():
    global x
    x = x * 2
    print(x)
foo()
```

OUTPUT: globalglobal

GLOBAL AND LOCAL VARIABLES

Example 4: Accessing local variable outside the scope.

```
def foo():
    y = "local"
foo()
print(y)
```

OUTPUT:

```
NameError: name 'y' is not defined
```

GLOBAL AND LOCAL VARIABLES

Example 5: Create a Local Variable

```
def foo():
    y = "local"
    print(y)
foo()
```

OUTPUT:

```
local
```

GLOBAL AND LOCAL VARIABLES

Example 6: Global variable and Local variable with same name

```
x = 5
def foo():
    x = 10
    print("local x:", x)
foo()
print("global x:", x)
```

OUTPUT:

```
local x: 10
global x: 5
```

GLOBAL AND LOCAL VARIABLES

Example 7: Create a nonlocal variable

```
def outer():
    x = "local"
    def inner():
        nonlocal x
        x = "nonlocal"
        print("inner:", x)
    inner()
    print("outer:", x)
outer()
```

OUTPUT:

```
inner: nonlocal
outer: nonlocal
```