CHITKARA UNIVERSITY

# Python Programming

## Recursive Functions

**Dr. Raj Gaurang Tiwari**

Department of Computer Science and Engineering
Chitkara University, Punjab

# Contents

CHITKARA UNIVERSITY

- Lembda Functions
- Recursive Function
- Working
- Examples

# The *Anonymous* Functions

CHITKARA UNIVERSITY

- You can use the *lambda* keyword to create small anonymous functions.
- These functions are called anonymous because they are not declared in the standard manner by using the *def* keyword.
- Python Lambda Functions are anonymous function means that the function is without a name.
- Lambda forms can take any number of arguments but return just one value in the form of an expression. They cannot contain commands or multiple expressions.
- lambda's are a one-line version of a function,
- **Syntax:**

```
lambda arguments : expression
```

# Example

CHITKARA UNIVERSITY

- Following is the example to show how *lembda* form of function works:

```
sum = lambda arg1, arg2: arg1 + arg2;

print "Value of total : ", sum( 10, 20 )
print "Value of total : ", sum( 20, 20 )
```

- This would produce following result:

```
Value of total : 30
Value of total : 40
```

# Example

CHITKARA UNIVERSITY

```
double = lambda x: x * 2
print(double(5))

x = lambda : "hello world"
print(x())
```

## NEED

Lambda functions reduce the number of lines of code when compared to normal python function defined using def keyword.

# Example

CHITKARA UNIVERSITY

- **Lambda functions can be Immediately Invoked**

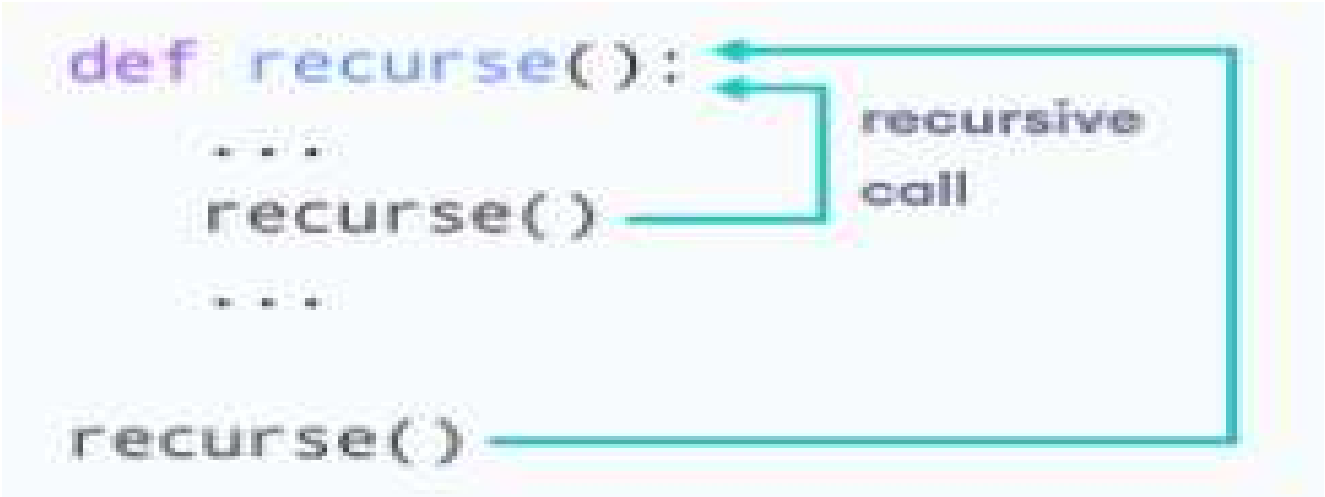
```
(lambda x, y : x*y)(5,7)
```

```
(lambda x, y=3, z=5: x*y*z)(x=7)
```

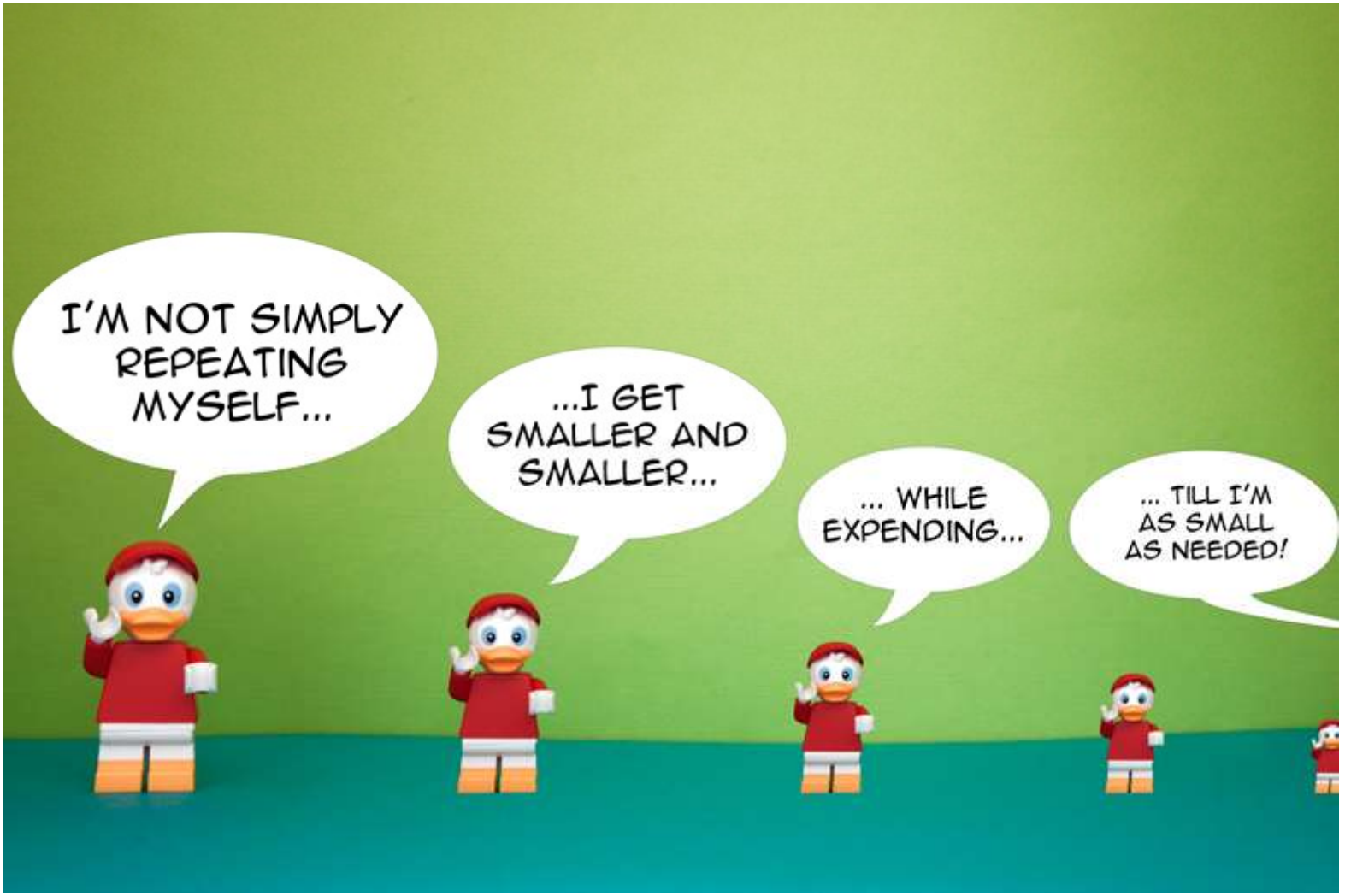
# Recursion

CHITKARA UNIVERSITY

- Recursion is the process of defining something in terms of itself.
- In Python, we know that a function can call other functions. It is even possible for the function to call itself. These types of construct are termed as recursive functions.



# Concept


I'M NOT SIMPLY REPEATING MYSELF...
...I GET SMALLER AND SMALLER...
... WHILE EXPENDING...
... TILL I'M AS SMALL AS NEEDED!

# Advantages of using recursion

CHITKARA UNIVERSITY

- A complicated function can be split down into smaller sub-problems utilizing recursion.
- Sequence creation is simpler through recursion than utilizing any nested iteration.
- Recursive functions render the code look simple and effective.

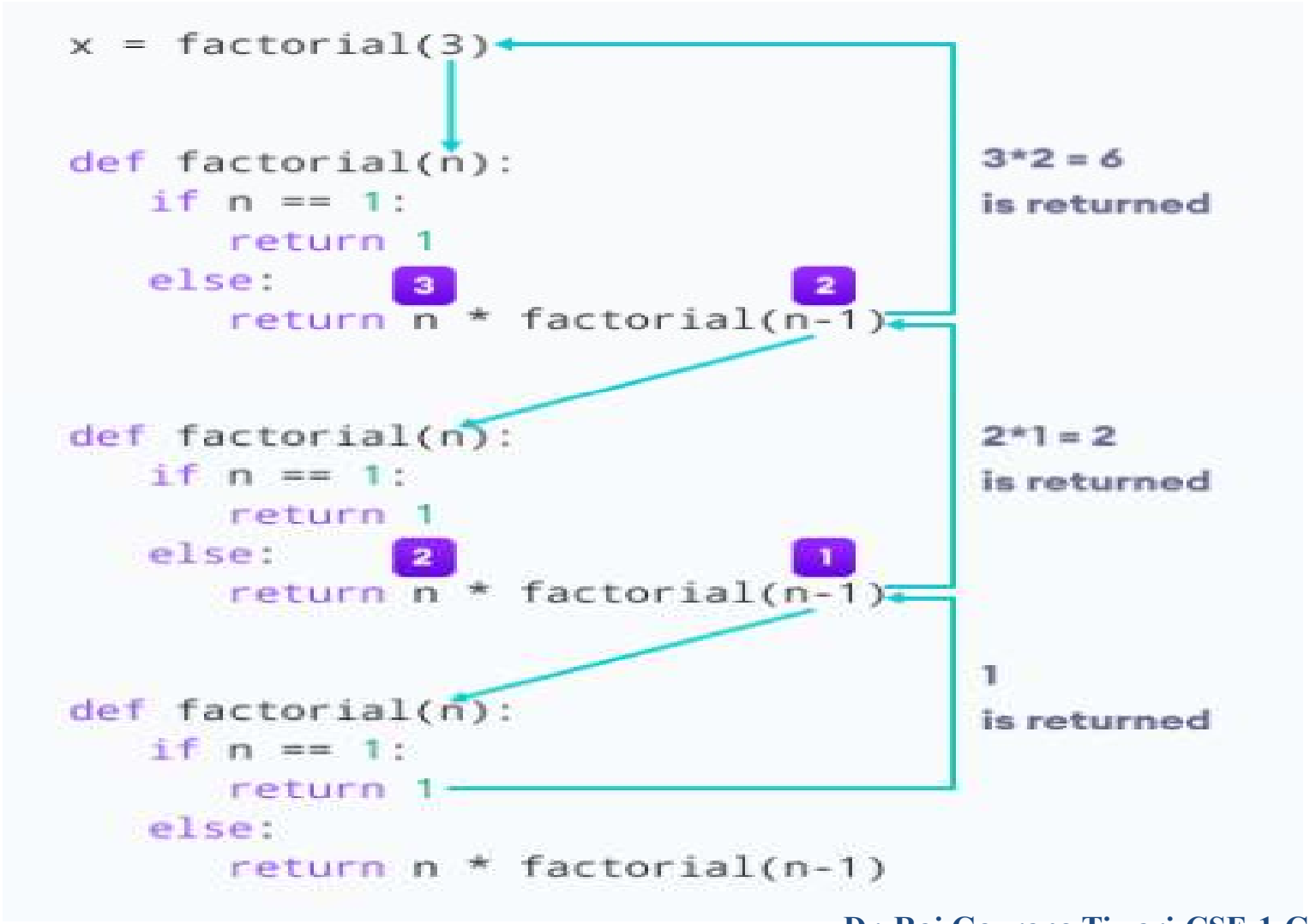
# Example

CHITKARA UNIVERSITY

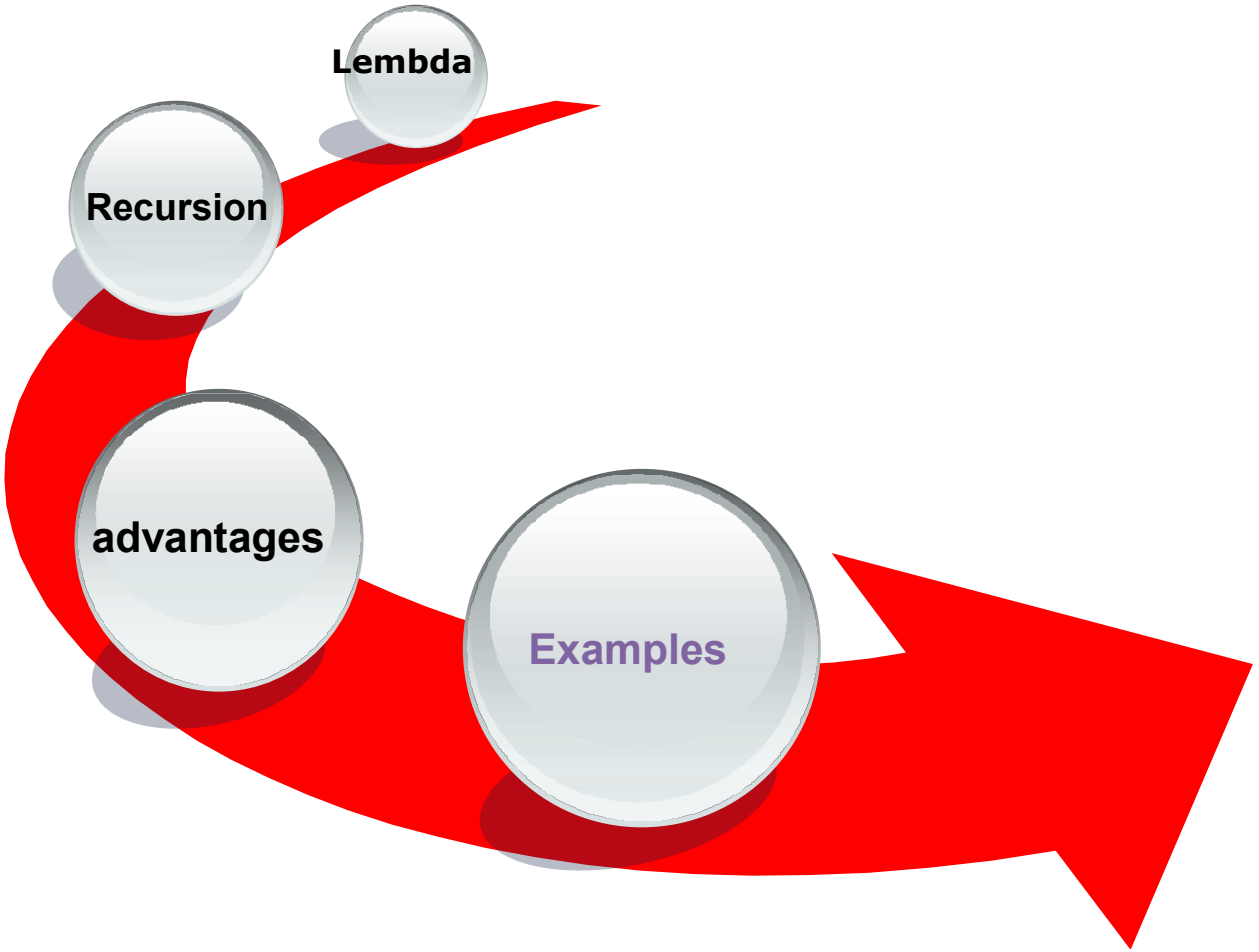
```
def factorial(x):
    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))


print("The factorial of", num, "is", factorial(5))
```

```
factorial(5)
= 5 * factorial(4)
= 5 * 4 * factorial(3)
= 5 * 4 * 3 * factorial(2)
= 5 * 4 * 3 * 2 * factorial(1)
= 5 * 4 * 3 * 2 * 1
= 120
```

# Recursion(Working)



# Summery



CHITKARA UNIVERSITY

# Thank You !