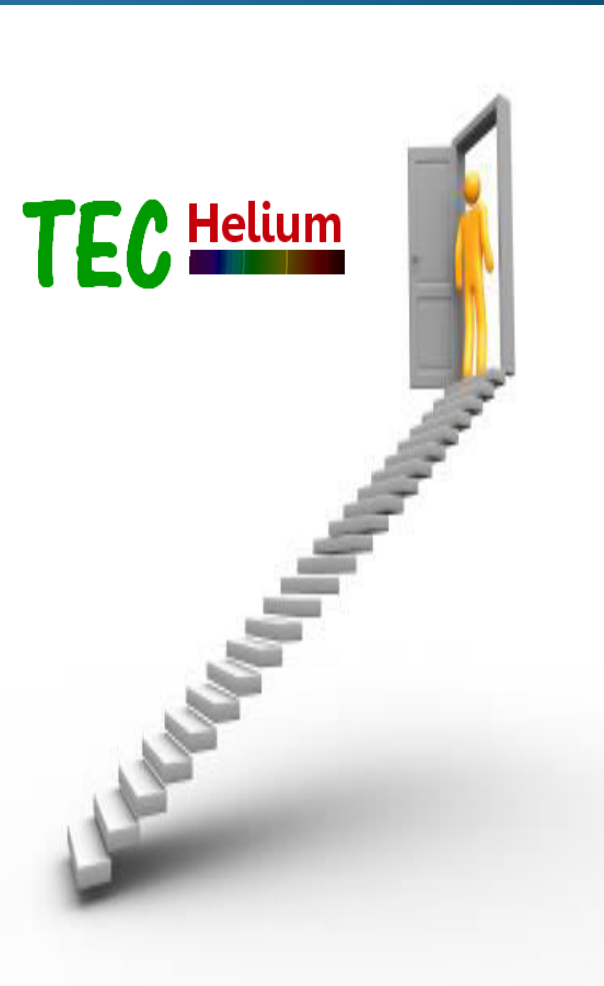


Lets Face it, now

# Project Oriented Python

## Functions & Programs



# Defining a function

```
1. def sum(numbers):  
2.     """Finds the sum of the numbers in a list."""  
3.     total = 0  
4.     for number in numbers:  
5.         total = total + number  
6.     return total
```

1. **def** defines a function  
**numbers** is a **parameter**
2. This **doc string** tells what the function does
6. A function that computes a value must **return** it  
**sum(range(1, 101))** will return **5050**

# Example Function

```
def gcd(a, b):  
    "greatest common divisor"  
    while a != 0:  
        a, b = b%a, a    # parallel assignment  
    return b
```

```
>>> gcd.__doc__  
'greatest common divisor'  
>>> gcd(12, 20)  
4
```

# Functions from other files

```
def double_num( a) :  
    return a*a
```

double.py

```
from double import *  
print double_num(3)  
  
##Will print 9
```

Sample function.py

# Functions : Key points

- Pass by reference : Any change done by function to the variable is permanent.
- Function must be defined before calling it.
- You can call a function by using the following types of formal arguments:
  - Required arguments
  - Keyword arguments
  - Default arguments
  - Variable-length arguments

# Functions : Required arguments

- Required arguments
  - `def new_func(number, string)`  
`print number*number, string`
  - `new_func(10, 'harish')`  
`##will print 100, harish`
  - `new_func('harish', 10)` or `new_func(10)`  
`##will give error`

# Functions : Keyword arguments

- Keyword arguments
  - ```
def new_func(number, string)  
    print number*number, string
```
  - ```
new_func(number = 10, string = 'harish')
```

```
## print 100, harish
```
  - ```
new_func(string='harish', number=10)
```

```
## print 100, harish
```

# Functions : Default arguments

- Default arguments
  - `def new_func(number, string='default')`  
`print number*number, string`
  - `new_func(10)`  
`## print 100, default`
  - `new_func(number=10)`  
`## print 100, default`
  - `new_func(10,'harish')`  
`## print 100, harish`



# Functions : Variable arguments

- You may need to process a function for more arguments than you specified while defining the function.
- These arguments are called *variable-length arguments and are not named in the function definition*.

- *Syntax*

```
def functionname([formal_args,] *var_args_tuple ):  
    "function_docstring"  
    Statements  
    return [expression]
```

```
#!/usr/bin/python  
  
# Function definition is here  
def printinfo( arg1, *vartuple ):  
    "This prints a variable passed "  
  
    print "Output is: "  
    print arg1  
  
    for var in vartuple:  
        print var  
    return  
  
# Main function starts here  
printinfo(10)  
printinfo(10,20,30)  
printinfo(10,'hck',30)
```

# Exercise

- Write a function that take two numbers and return their sum and product
- Write a function equivalent to range function that return list.
- Write a function that take any number of numbers and return sum of all numbers
- Write a function that take any number of numbers and return a sorted list.
- Write a function that take any number of numbers and find smallest and biggest number.

# Exercise

- Convert all program written earlier using function.
  - Keep all functions (like greater number, fibonacci series, factorial, etc) in one file.
  - Main file/program gets number from user and call the functions.
- Identify at least 5 functions for the project you would be doing

Thanks