Lecture 8
Functions

Objectives

- After completing the lesson, the student will be able to:
  - Differentiate library functions and user defined functions.
  - Use library functions in programs.
  - Define a user defined function.
  - Call a user defined function.
  - Describe the output of programs with functions.

Built-in Functions

- Library functions are built-in functions available in the application.
- Use the function name with the required arguments.
- Arguments are the values included in the functions.
  - Example: pow() calculates the power of a value:
    - pow(4,2)
    - Here 4 and 2 are **arguments**, which are the base and power respectively.
    - pow(4,2) evaluates to 16 (i.e. 4^2)

Built-in Functions

		Built-in Functions		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

Built-in Functions

- Built-in functions can be listed by using the built-in function dir(__builtins__).
- The built-in function help() can be used to find more about the other built-in functions.
- Example:
  - help(abs)

```
Help on built-in function abs in module builtins:

abs(x, /)
    Return the absolute value of the argument.
```

Built-in Functions

- Example:

```
abs(-23.5)
23.5
```

  - When we pass a floating point argument, the result is of type float.

- Example:

```
abs(-54)
54
```

  - When we pass an integer argument, the result is of type integer.

Built-in Functions

- The `max()` function can be used to find the maximum number from the list of arguments.

- max() function takes any number of arguments which are separated by comma and returns the largest number.

- Example:

```
max_number = max(124,-52,300)
print(max_number)
```

- This will print 300 as the value of max_number

Built-in Functions

- The `min()` function can be used to find the minimum number from the list of arguments.

- min() function takes any number of arguments which are separated by comma and returns the smallest number.

- Example:

```
min_number = min(124,-52,300)
print(min_number)
```

- This will print -52 as the value of min_number

Built-in Functions

- round() function can be used to round a number to a given precision in decimal digits.

- This always returns a floating point number.

- Example:

```
round(12.75245,2)
12.75
```

- Example:

```
round(12.7554,2)
12.76
```

Built-in Functions

- sum() function can be used to calculate the sum of the sequence.
- It returns the sum of the sequence of numbers plus the value of the parameter ‘start’.
- When the sequence is empty, returns start.
- General syntax:

```
sum(sequence[, start])
```

- Example:

```
List = [1,2,3,4,5,6,7,8,9,10]
add = sum(List)
print(add)
```

Built-in Functions

- Example:

```
List = [1,2,3,4,5,6,7,8,9,10]
add = sum(List,5)
print(add)
```

- Example:

```
list = []
add = sum(list,2)
print(add)
```

Built-in Functions

- Modules are python files which contain different function definitions.
- To use a module you have to `import` it first.
- Example:

```
import math
math.sqrt(16)
```

- In this example, math is the module and the sqrt() is the function.
- After importing math module, more mathematical function will be available.

Built-in Functions

- Use `help()` function to learn more about the functions.
- Example: `sqrt()` function
  - Import the math module first
  - Type `help(math.sqrt)` at the prompt

```
sqrt(x)
 Return the square root of x.
```

- Example:

```
import math
math.sqrt(25)
5.0
```

Built-in Functions

- Other important modules available
  - graphics - to create GUI and graphics.
  - string – to manipulate strings
  - Other modules include Databases, GUIs, Images, Sound, OS interaction, Web, and more.

User Defined Functions

- Functions which are defined by the programmer.
- Functions can be used to reduce code duplication and make programs more understandable and easier to maintain.
- The basic idea of a function is that we write a sequence of statements and give that sequence a name.
- The instructions can then be executed at any point in the program by referring to the function name.

User Defined Functions

- The part of the program that creates a function is called a function definition.
- When a function is subsequently used in a program, we say that the definition is called or invoked.
- A single function definition may be called at many different points of a program.

Function Definition

- General syntax:

```
def function name(parameters):
    Statements of the function
```

- Example:

```
def happy():
    print("Happy birthday to you!")
    print("Happy birthday to you!")
    print("Happy birthday, dear xxxx.")
    print("Happy birthday to you!")
```

Function Definition

- To run the function, type the name of the function at the command prompt of the python shell editor.

- Example:
  - To run the above function, type `happy()` and then press enter key.
- In the above function, some statements are repeating.
- This can be avoided by creating a function for that statement.

Calling a Function

```
def birthday():
    print("Happy birthday to you! ")
```

- Create another function to call above function.

```
def wishingxxx():
    birthday()
    birthday()
    print("Happy birthday, dear xxx.")
    birthday()
```

Calling a Function

```
def wishingyyy():
    birthday()
    birthday()
    print("Happy birthday, dear yyy.")
    birthday()
```

- Now we can combine the two wishing functions as a single function.
- What should we do if there are more people to wish?

Calling a Function

- This function can be used to wish both xxx and yyy.

```
def wish():
   wishingxxx()
   print()
   wishingyyy()
```

- The print statement between the two function call prints a blank line.

Calling Functions

- Example: calculate sum and difference of two numbers.

```
def s():
    a = 5
    b = 6
    add = a+b
    print("Sum  = ", add)
```

Calling Functions

```
def subtract():
    a = 15
    b = 6
    sub = a-b
    print("Difference = ", sub)

def calculate():
     s()
     subtract()
```