Python Programming

Day 8: Functions

Functions

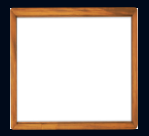
def	return
elif	while

Color and symbol meaning

Hint

Preferred

Student's activity



Practice code

	Keyword
	In-built functions
	Strings
	Output

Functions

A function is a block of **organized**, **reusable** code that is used to perform a single, related action.

Functions provide better modularity for your application and a high degree of code reusing.

Functions

The syntax for a function definition is:

```
def NAME( LIST OF PARAMETERS ):
    STATEMENTS
```

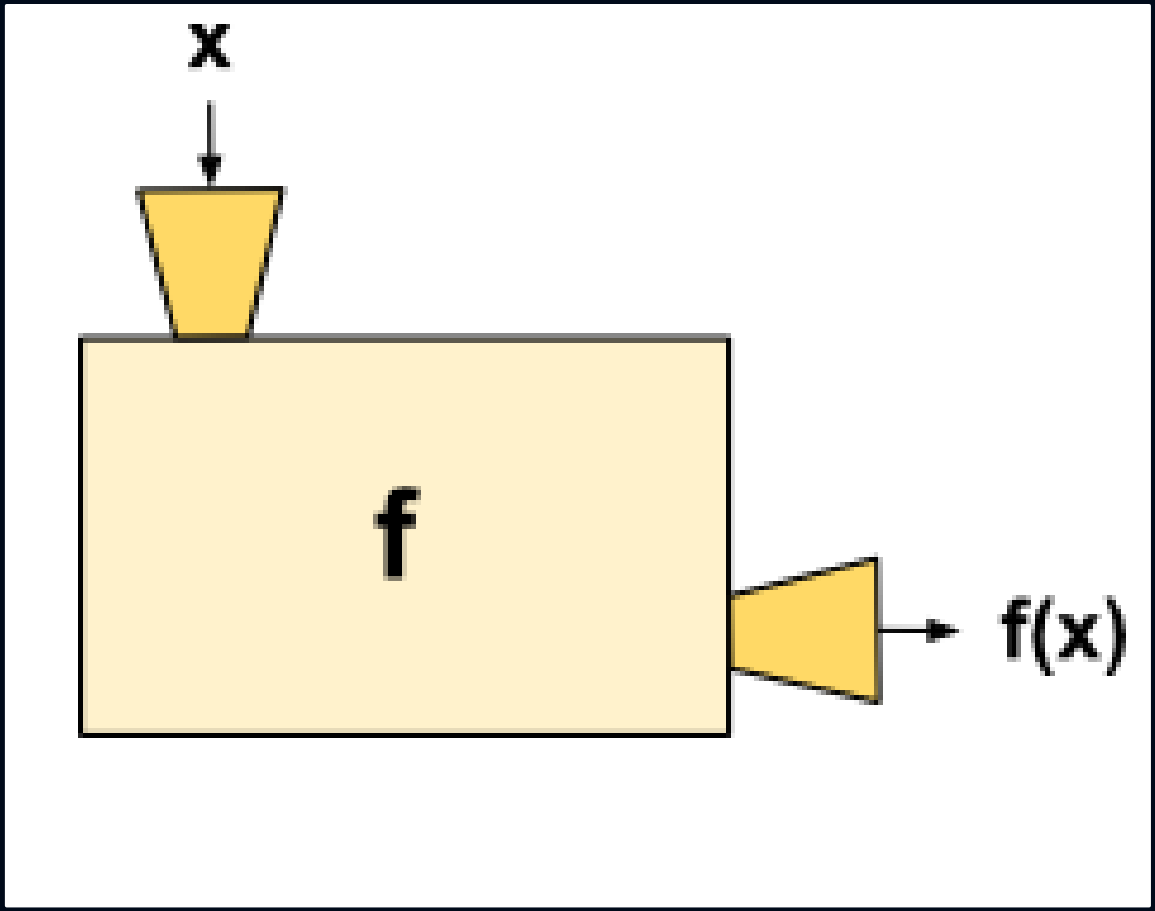
Each of the statements inside the body are executed in order if the Boolean expression evaluates to True. The entire block is skipped if the Boolean expression evaluates to False.

Defining a Function

- Function blocks begin with the keyword **def** followed by the function name and parentheses (( ))
- Any input parameters or arguments should be placed within these parentheses.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Defining a Function

The idea behind this diagram is that a function is like a machine that takes an input, *x*, and transforms it into an output, *f(x)*.



The light yellow box ***f*** is an abstraction of the process used to do the transformation from x to *f(x)*.

Defining a Function

The following function takes a string as input parameter and prints it on standard screen.

```
def printme(str):
  "This prints a passed string into this function"
  print (str)
  return
```

Defining a Function

Sample Code

The following quadratic function is an example:

$$f(x) = 3x\ 2 - 2x + 5$$

```
def f(x):

    return 3 * x ** 2 - 2 * x + 5
```

Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

Once the basic structure of a function is finalized, you can execute it by **calling** it from another function or directly from the Python prompt.

Calling a Function

Sample Code

The function definition must first be entered into the Python shell before it can be called:

```
# Function definition is here
def printme(str):
   "This prints a passed string into this function"
   print (str)
   return;

# Now you can call printme function
printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

Class Activity 1

Write a Python function to sum all the numbers in a list.

Sample List : (8, 2, 3, 0, 7)
Expected Output : 20

Pass by reference vs value

All parameters (arguments) in the Python language are passed by reference.

It means if you change what a parameter refers to within a function, the change also reflects back in the calling function

```
# Function definition is here
def changeme(mylist):
   "This changes a passed list into this function"
   mylist = [1,2,3,4]; # This would assig new reference in mylist
   print ("Values inside the function: ", mylist)
   return
# Now you can call changeme function
mylist = [10,20,30];
changeme(mylist);
print ("Values outside the function: ", mylist)
```

Function Arguments

You can call a function by using the following types of formal arguments:

- **Required** arguments
- **Default** arguments
- **Variable-length** arguments

Function Arguments

Required arguments

Required arguments are the arguments passed to a function in correct positional order. Here, the number of arguments in the function call should match exactly with the function definition.

```
# Function definition is here
def printme(str):
  "This prints a passed string into this function"
  print (str)
  return;
# Now you can call printme function
printme()
```

Function Arguments

When the above code is executed, it produces the following result

```
Traceback (most recent call last):
  File "test.py", line 11, in <module>
    printme();
TypeError: printme() takes exactly 1 argument (0 given)
```

Class Activity 2

Write a Python program that accepts a hyphen-separated sequence of words as input and prints the words in a hyphen-separated sequence after sorting them alphabetically.

Sample Items : green-red-yellow-black-white
Expected Result : black-green-red-white-yellow

Function Arguments

Sample Code

Default arguments

A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.

```
def printinfo( name, age = 35 ):
  "This prints a passed info into this function"
  print ("Name: ", name)
  print ("Age ", age)
  return;

# Now you can call printinfo function
printinfo( age=50, name="miki" )
printinfo( name="miki" )
```

Function Arguments

Variable-length arguments

Variable-length arguments make it possible to define a function without specifying the number of arguments required.

```
def printinfo(arg1, *vartuple):
  "This prints a variable passed arguments"
  print ("Output is: ")
  print (arg1)
  for var in vartuple:
    print (var)
  return;

# Now you can call printinfo function
printinfo(10)
printinfo(70, 60, 50)
```

Function Arguments

Variable-length arguments

Single * (*arg) is used to pass a non keyworded variable-length argument list.

Double ** (**kwarg) is used to pass a keyworded (key-value pair argument) variable-length argument list.

The return Statement

The return statement causes a function to immediately stop executing statements in the function body and to send back (or return) the value after the keyword return to the calling statement.

```
def sum( arg1, arg2 ):
  # Add both the parameters and return them."
  total = arg1 + arg2
  print ("Inside the function : ", total)
  return total;

# Now you can call sum function
total = sum( 10, 20 );
print ("Outside the function : ", total)
```

Built-in Functions

Certain types, functions, and variables are always available to the interpreter and can be used in any source module.

No need of imports to access these functions, they are contained in a module **builtins**.

Built-in Functions

		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

Class Activity 3

Write a Python function to find the Max of three numbers.

Next Lecture ...

Python Programming

Day 9: Classes and Objects