Functions

Agenda

1. What are functions?
2. Built-in functions
3. User defined functions
4. Functions with arguments

What are functions?

What are functions?

- A function is a group of related statements that perform a specific task.
- A function is a block of reusable code.
- Functions make our code more organized and manageable.

The advantages of using functions are:

- Duplication of code is reduced.
- Breaking complex problems into simple small pieces.
- Improving clarity of the code.
- Reusability of code.

Built-in functions

Built-in functions

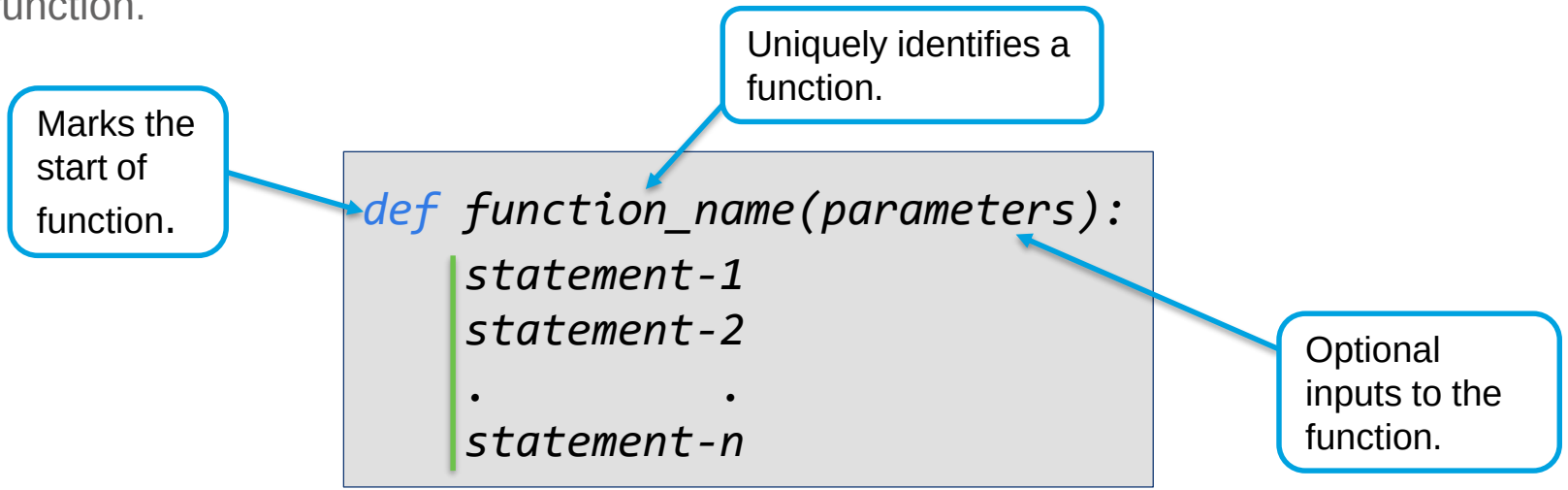
- Python has a number of built-in functions which are always ready to use.
- Built-in functions helps us in accomplishing the tasks quickly.

Frequently used built-in functions			
type	id	max	int
len	chr	min	float
abs	ord	input	bool
pow	print	super	lower
range	sum	open	upper

User defined functions

User defined functions

- You can create(define) your own functions to write and implement functionalities.
- If a functionality(set of code) needs to be executed repeatedly in your program, Function can be used to include those codes and execute it whenever we required, by calling that function.



User defined functions

```
Execute this code:
def message():
    print('Welcome to Python tutorial')


message() #calling or invoking the function
```

This piece of code will never run unless you call the function.

```
Output:

Welcome to Python tutorial
```

User defined functions : Quiz

```
Predict the output:

message() #calling or invoking the function

def message():

    print('Welcome to Python tutorial')
```


```
Traceback (most recent call last):
  File "main.py", line 1, in <module>
    message()
NameError: name 'message' is not defined
```

User defined functions : return statement

- A return statement is used to return the result(value).
- **return** is a keyword which ends the execution of the function call.

```
Program:
def message():
    return 'Hello World'

s1 = message() #print(message())
print(s1)
```

```
Output:
Hello World
```

User defined functions : return statement

- Statements written after **return** are never executed.

```
Program:
def message():
    return 'Hello World'
    print('End')

print(message())
```

```
Output:
Hello World
```

User defined functions : return statement

- When there is no return statement, by default functions in python return **None.**

```
Program:
def message():
    print('Hello World')

s1 = message() #receiving None in s1
print(s1)
```

```
Output:
Hello World
None
```

Functions with arguments

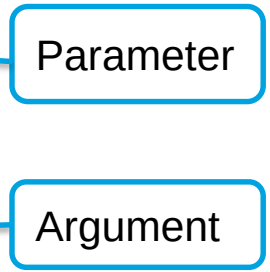
Functions with arguments

- Functions can take inputs called as **parameters**.
- Parameters are specified inside the parentheses ( ).
- Multiple parameters are separated with comma.
- The values we pass to the parameters in the function call are called **arguments** or **args**.

Function with one parameter:

```
def welcome_msg(name):
    print('Hi',name,'Welcome..')

welcome_msg('Arun')
```



Functions with arguments : Quiz

```
Predict the output:
def welcome_msg(name):
    print('Hi',name,'Welcome..')

welcome_msg( )
```

Required argument.

```
Traceback (most recent call last):
File "main.py", line 4, in <module>
TypeError: welcome_msg() missing 1 required
Positional argument: 'name'
```

Functions with arguments

Required arguments

- Required arguments are passed to a function in correct order.
- The number of arguments in the function call should match exactly with the number of parameters in the function definition.

Function definition	Right way to call
def calculate(x, y, z):	calculate(10, 20, 30) calculate(1.5, 2.5, 3.5)
def prime_check(input1):	prime_check(27) prime_check(4)

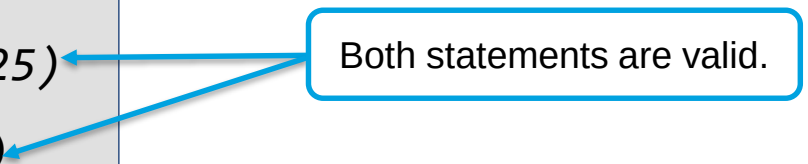
Functions with arguments

Keyword arguments

- While calling a function by passing arguments, we can name the arguments with their respective parameter name to which these arguments are being passed.
- With these keyword arguments we can pass the arguments in any order.

```
def message(name, age):
    print('Hi', name)
    print('Age is', age)

message(name='Chandu', age=25)
message(age=24, name='Ajay')
```



Functions with arguments

Default arguments

- A default argument assumes a default value if value is not passed to it in the function call.

```
Program:
def display(a, b, c=50):
    print(a, b, c)

display(10,20)
display(10,20,30)
```

```
Output:


10 20 50
10 20 30
```

- When value is passed in the function call, it overrides the default value.

Thank you