Functions

Dr. Seema Gupta Bhol

Functions in Python

- A function is a block of code which only runs when it is called.
- A function can return data as a result.
- A function helps avoiding code repetition.
- A function is defined using the def keyword, followed by a function name and parentheses

```
def my_function():
  print("Hello from a function")
```

- To call a function, write its name followed by parentheses
- The code inside the function must be indented. Python uses indentation to define code blocks.

Example: Fahrenheit to Celsius Conversion

```
def fahrenheit_to_celsius(fahrenheit):
  return (fahrenheit - 32) * 5 / 9

print(fahrenheit_to_celsius(77))
print(fahrenheit_to_celsius(95))
print(fahrenheit_to_celsius(50))
```

Return Values

Functions can send data back to the code that called them using the return statement.
When a function reaches a return statement, it stops executing and sends the result back

```
def get_greeting():
  return "Hello from a function"
message = get_greeting()
print(message)
```

The returned value can be used directly

```
def get_greeting():
  return "Hello from a function"
print(get_greeting())
```

If a function doesn't have a return statement, it returns None by default.

Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. Arguments are separated by comma.

```
def my_function(fname):
  print("I am" + fname)
my_function("Tom")
my_function("Jerry")
my_function("Pluto")
```

Parameters vs Arguments

- The terms parameter and argument can be used for the same thing: information that are passed into a function.
- A parameter is the variable listed inside the parentheses in the function definition.
- An argument is the actual value that is sent to the function when it is called.

```
def my_function(name): # name is a parameter
  print("Hello", name)
my_function(“Pluto")  # “Pluto" is an argument
```

Default Parameter Values

- Default values can be assigned to parameters.
- If the function is called without an argument, it uses the default value

```
def my_function(name = "friend"):
  print("Hello", name)


my_function("Tom")
my_function("Jerry")
my_function()
```

Passing Different Data Types

Any data type can be sent as an argument to a function (string, number, list, dictionary, etc.).

The data type will be preserved inside the function:

```
def my_function(fruits):
  for fruit in fruits:
    print(fruit)
my_fruits = ["apple", "banana", "cherry"]
my_function(my_fruits)
```

Example: Sending a dictionary as an argument

```
def my_function(person):
  print("Name:", person["name"])
  print("Age:", person["age"])

first_person = {"name": “Tom", "age": 25}
my_function(first_person)
```

Return Values

Functions can return values using the return statement:

```
def my_function(x, y):
  return x + y
result = my_function(5, 3)
print(result)
```

Function Returning multiple values

```
def my_function(x, y):
  return (x + y, x*y)

s,p = my_function(5, 3)
print(“The sum is”,s)
print(“The product is”,p)
```

Exercise

1. Write a function to find square of a number.
2. Write a function to calculate simple interest.
3. Write a function to calculate factorial of a number.
4. Write a function to check whether a number is prime.
5. Function to return both maximum and minimum of list of numbers.
6. Write a function to count even numbers in a list.
7. Write a function to find length of a tuple.
8. Write a function to print all values of dictionary.
9. Write a function to count occurrences of an element in tuple.
10. Write a function to reverse a list.