Programming Fundamental

A Hitchhiker Guide to Coding with Python

Lesson 5: Functions

Ratthaprom PROMKAM, Dr.rer.nat

Department of Mathematics
Faculty of Science and Technology, RMUTT

Lesson Outline

1. The concept of function
2. User-Defined Functions
3. Function Inputs and Outputs

The concept of function

Image Source: *John Sturtz*,

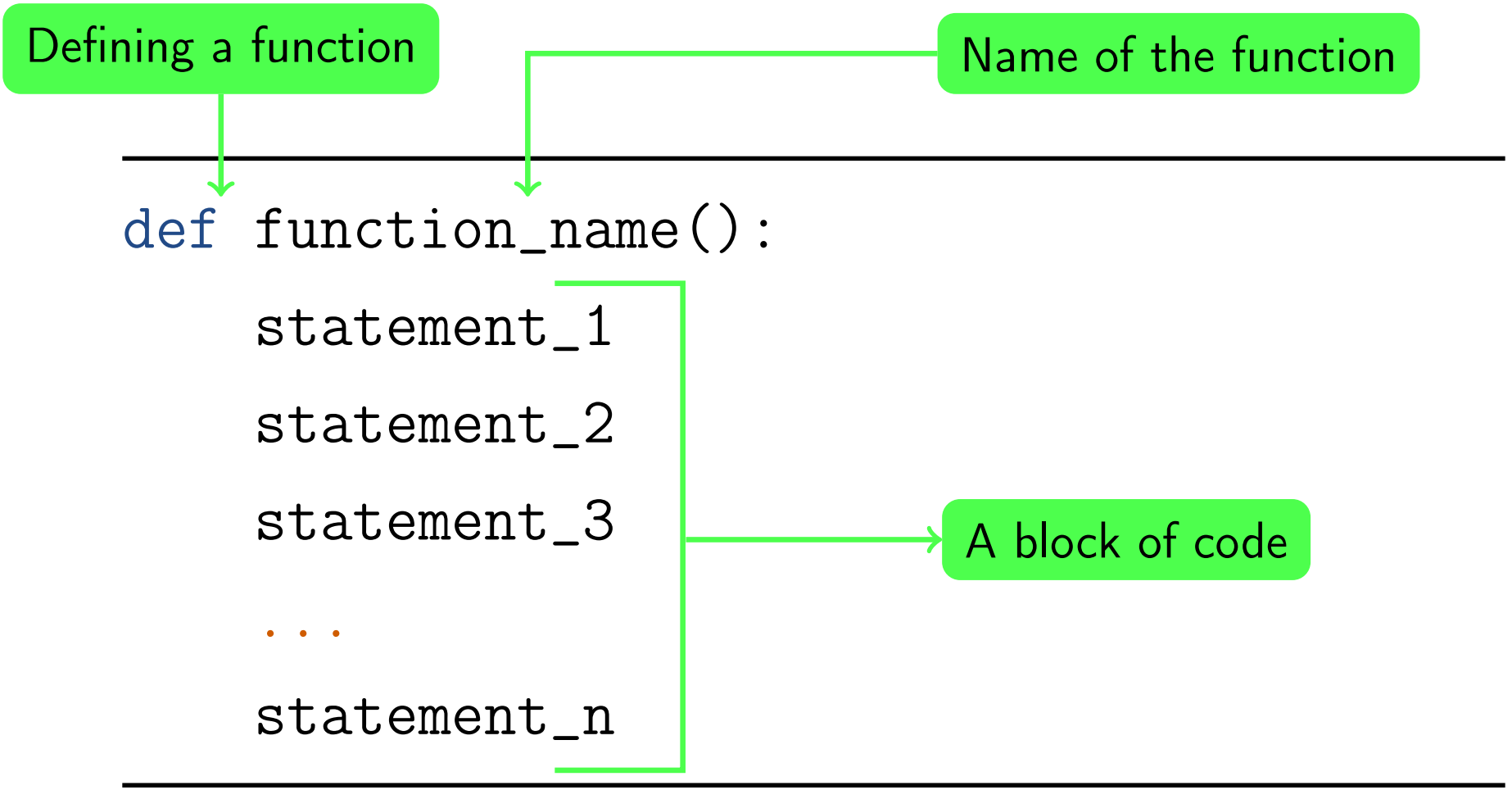
- A function is a block of code that only runs when called.
- Like a sub-program, functions can accept inputs and provide outputs.
- A function often only accomplishes one certain task.
- Functions increase the reusability and modularity of code.

Python Built-In Functions

Here are some functions you have already seen.

Function	Description
`print()`	Prints to the standard output device
`input()`	Allowing a user input and returning a string from it
`int()`	Return an integer number
`float()`	Return a floating-point number
`str()`	Return a string
`len()`	Returns the length of an object

User-Defined Functions



Example

line_fn.py

```
1  def line():
2      x = '-' * 30
3      print(x)
4
5
6  x = 10
7  print('x = ', x)
8  line()
9  x = x + 10
10 print('x = ', x)
11 line()
```

```
>_ python line_fn.py
x =  10
------------------------------
x =  20
------------------------------
```

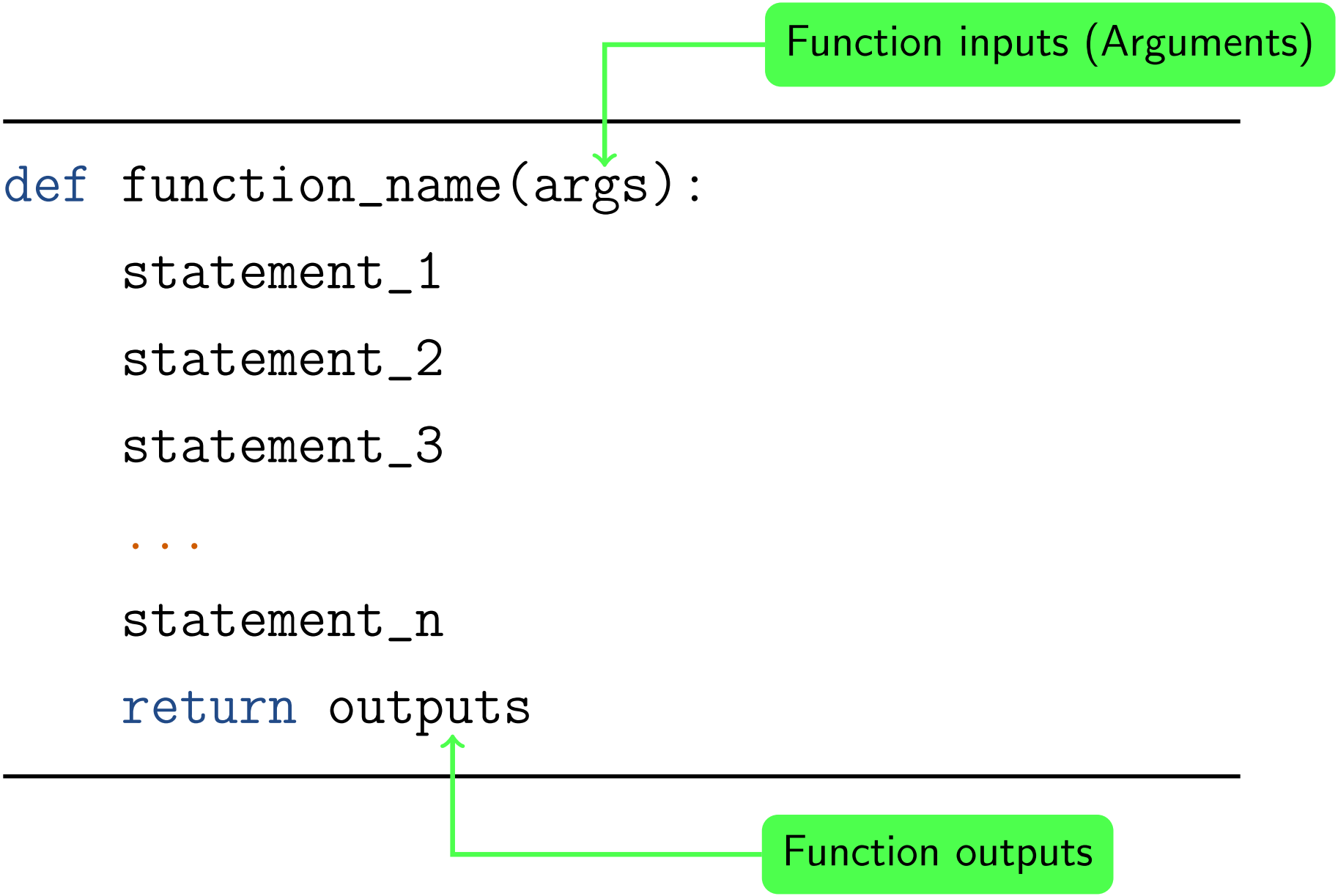
Example

hi_offer.py

```
1  def say_hi():
2      print('Hello')
3      print('Nice to meet you!')
4
5  def offer_meal():
6      print('Please have some meal')
7      print('It is very good pasta')
8
9
10 guest = 'Jame Doe'
11 print('Here is', guest)
12 say_hi()
13 offer_meal()
```

```
>_ python hi_offer.py
Here is Jame Doe
Hello
Nice to meet you!
Please have some meal
It is very good pasta
```

Function Inputs and Outputs



Example

line_fn.py

```
1  def line(symbol):
2      x = symbol * 25
3      print(x)
4
5
6  x = 10
7  print('x = ', x)
8  line('-')
9  x = x + 10
10 print('x = ', x)
11 line('*')
```

```
>_ python line_fn.py
x =  10
-------------------------
x =  20
*************************
```

Example

line_fn.py

```
1  def line(symbol, n):
2      x = symbol * n
3      print(x)
4
5
6  x = 10
7  print('x = ', x)
8  line('#', 15)
9  x = x + 10
10 print('x = ', x)
11 line('@', 25)
```

```
>_ python line_fn.py
x =  10
###############
x =  20
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Example

line_fn.py

```
1  def line(symbol, n):
2      x = symbol * n
3      return x
4
5
6  x = 10
7  print('x = ', x)
8  print(line('#', 15))
9  x = x + 10
10 print('x = ', x)
11 print(line('@', 25))
```

```
>_ python line_fn.py
x =  10
###############
x =  20
@@@@@@@@@@@@@@@@@@@@@@@@@
```

Example

circle_fn.py

```
1  def circle_area(radius):
2      pi = 3.14159265359
3      area = pi * (radius ** 2)
4      return area
5
6  def line(symbol, n):
7      x = symbol * n
8      return x
9
10
11 radius_list = [3.44, 1.56, 6.88]
12 for r in radius_list:
13     print('Radius is', r)
14     a = circle_area(r)
15     print('Area is', a)
16     print(line('-', 25))
```

```
>_ python circle_fn.py
Radius is 3.44
Area is 37.17635082552262
-------------------------
Radius is 1.56
Area is 7.645379881776625
-------------------------
Radius is 6.88
Area is 148.70540330209047
-------------------------
```

Example

circle_fn.py

```
1  def circle_cal(radius):
2      pi = 3.14159265359
3      area = pi * (radius ** 2)
4      perimeter = 2 * pi * radius
5      return area, perimeter
6
7  def line(symbol, n):
8      x = symbol * n
9      return x
10
11 radius_list = [3.44, 1.56, 6.88]
12 for r in radius_list:
13     print('Radius is', r)
14     a, p = circle_cal(r)
15     print('Area is', a)
16     print('Perimeter is', p)
17     print(line('-', 25))
```

```
>_ python circle_fn.py
Radius is 3.44
Area is 37.17635082552262
Perimeter is 21.6141574566992
-------------------------
Radius is 1.56
Area is 7.645379881776625
Perimeter is 9.8017690792008
-------------------------
Radius is 6.88
Area is 148.70540330209047
Perimeter is 43.2283149133984
-------------------------
```

Example

hi_offer.py

```
1  def say_hi(name):
2      print('Hello', name)
3      print('Nice to meet you!')
4
5  def offer_meal(name):
6      if name == 'Lucy':
7          print(name, 'needs no meal')
8      else:
9          print('Please have some meal')
10         print('It is very good pasta')
11
12
13 guests = ['James', 'Lucy', 'Susan']
14 for g in guests:
15     say_hi(g)
16     offer_meal(g)
```

```
>_ python hi_offer.py
Hello James
Nice to meet you!
Please have some meal
It is very good pasta
Hello Lucy
Nice to meet you!
Lucy needs no meal
Hello Susan
Nice to meet you!
Please have some meal
It is very good pasta
```

Exercise

Write a Python function `absolute` to find the absolute value $|x|$ of a real number x, i.e.,

$$|x| = \begin{cases} x & \text{if } x \geq 0, \\ -x & \text{if } x < 0. \end{cases}$$

The function takes an argument (input), which is a real number, and returns an output of its absolute value.

Command	Result
`absolute(5)`	5
`absolute(-4.234)`	4.234
`absolute(0)`	0
`absolute(-2.14e-3)`	2.14e-3

Exercise

Write a Python function `solve_quad` to solve a quadratic equation $ax^2 + bx + c = 0$ where $a \neq 0$, with the formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

The function takes 3 arguments (inputs), which are the coefficients a, b and c in a quadratic equation, and returns outputs of the solutions to the equations.

Command	Result
`solve_quad(1, -5, 6)`	`2.0 3.0`
`solve_quad(-2, 2, 1)`	`1.366 -0.366`
`solve_quad(1, 2, 1)`	`-1.0 -1.0`
`solve_quad(0, 5, 6)`	`None`

Exercise

Write a Python function `solve_poly` to solve a polynomial equation

$$ax^2 + bx + c = 0.$$

The function takes 3 arguments (inputs), which are the coefficients a, b and c in the equation, and returns outputs of the solutions.

Command	Result
`solve_poly(1, -5, 6)`	`2.0 3.0`
`solve_poly(-2, 2, 1)`	`1.366 -0.366`
`solve_poly(1, 2, 1)`	`-1.0`
`solve_poly(0, 5, 6)`	`-1.2`