30 dias de Python: Dia 11 - Funções

Autor: Asabeneh Yetayeh

Segunda edição: julho de 2021

<< Dia 10 | Dia 12 >>

📘 Dia 11

Funções

Até agora, vimos muitas funções integradas do Python. Nesta seção, focaremos nas funções personalizadas. O que é uma função? Antes de começarmos a criar funções, vamos aprender o que é uma função e por que precisamos delas.

Definindo uma Função

Uma função é um bloco reutilizável de código ou instruções de programação projetadas para executar uma determinada tarefa. Para definir ou declarar uma função, Python fornece a palavra-chave *def* . A seguir está a sintaxe para definir uma função. O bloco funcional de código é executado somente se a função for chamada ou invocada.

Declarando e Chamando uma Função

Quando criamos uma função, a chamamos de declaração de função. Quando começamos a usá-lo, chamamos-lhe *chamada* ou *invocação* de uma função. A função pode ser declarada com ou sem parâmetros.

```
# syntax
# Declaring a function
def function_name():
    codes
    codes
# Calling a function
function_name()
```

Função sem parâmetros

A função pode ser declarada sem parâmetros.

Exemplo:

```
def generate_full_name ():
    first_name = 'Asabeneh'
    last_name = 'Yetayeh'
    space = ' '
    full_name = first_name + space + last_name
    print(full_name)
generate_full_name () # calling a function

def add_two_numbers ():
    num_one = 2
    num_two = 3
    total = num_one + num_two
    print(total)
add_two_numbers()
```

Função que retorna um valor - Parte 1

Function can also return values, if a function does not have a return statement, the value of the function is None. Let us rewrite the above functions using return. From now on, we get a value from a function when we call the function and print it.

```
def generate_full_name ():
    first_name = 'Asabeneh'
    last_name = 'Yetayeh'
    space = ' '
    full_name = first_name + space + last_name
    return full_name
print(generate_full_name())

def add_two_numbers ():
    num_one = 2
    num_two = 3
    total = num_one + num_two
    return total
print(add_two_numbers())
```

Function with Parameters

In a function we can pass different data types(number, string, boolean, list, tuple, dictionary or set) as a parameter

- Single Parameter: If our function takes a parameter we should call our function with an argument

```
# syntax
# Declaring a function
def function_name(parameter):
  codes
  codes
# Calling function
print(function_name(argument))
```

Example:

```
def greetings (name):
    message = name + ', welcome to Python for Everyone!'
    return message

print(greetings('Asabeneh'))

def add_ten(num):
    ten = 10
    return num + ten
print(add_ten(90))

def square_number(x):
    return x * x
print(square_number(2))

def area_of_circle (r):
    PI = 3.14
    area = PI * r ** 2
    return area
print(area_of_circle(10))

def sum_of_numbers(n):
    total = 0
    for i in range(n+1):
        total+=i
    print(total)
print(sum_of_numbers(10)) # 55
print(sum_of_numbers(100)) # 5050
```

- Two Parameter: A function may or may not have a parameter or parameters. A function may also have two or more parameters. If our function takes parameters we should call it with arguments. Let us check a function with two parameters:

```
# syntax
# Declaring a function
def function_name(para1, para2):
  codes
  codes
# Calling function
print(function_name(arg1, arg2))
```

Example:

```
def generate_full_name (first_name, last_name):
    space = ' '
      full_name = first_name + space + last_name
      return full_name
print('Full Name: ', generate_full_name('Asabeneh','Yetayeh'))

def sum_two_numbers (num_one, num_two):
    sum = num_one + num_two
    return sum
print('Sum of two numbers: ', sum_two_numbers(1, 9))
```

```
def calculate_age (current_year, birth_year):
    age = current_year - birth_year
    return age;

print('Age: ', calculate_age(2021, 1819))

def weight_of_object (mass, gravity):
    weight = str(mass * gravity)+ ' N' # the value has to be changed to a string first
    return weight
print('Weight of an object in Newtons: ', weight_of_object(100, 9.81))
```

Passing Arguments with Key and Value

If we pass the arguments with key and value, the order of the arguments does not matter.

```
# syntax
# Declaring a function
def function_name(para1, para2):
    codes
    codes
# Calling function
print(function_name(para1 = 'John', para2 = 'Doe')) # the order of arguments does not
```

Example:

```
def print_fullname(firstname, lastname):
    space = ' '
    full_name = firstname  + space + lastname
    print(full_name)
print(print_fullname(firstname = 'Asabeneh', lastname = 'Yetayeh'))

def add_two_numbers (num1, num2):
    total = num1 + num2
    print(total)
print(add_two_numbers(num2 = 3, num1 = 2)) # Order does not matter
```

Function Returning a Value - Part 2

If we do not return a value with a function, then our function is returning *None* by default. To return a value with a function we use the keyword *return* followed by the variable we are returning. We can return any kind of data types from a function.

- Returning a string: **Example:**

```
def print_name(firstname):
    return firstname
print_name('Asabeneh') # Asabeneh

def print_full_name(firstname, lastname):
```

```
    space = ' '
    full_name = firstname  + space + lastname
    return full_name
print_full_name(firstname='Asabeneh', lastname='Yetayeh')
```

- Returning a number:

Example:

```
def add_two_numbers (num1, num2):
    total = num1 + num2
    return total
print(add_two_numbers(2, 3))

def calculate_age (current_year, birth_year):
    age = current_year - birth_year
    return age;
print('Age: ', calculate_age(2019, 1819))
```

- Returning a boolean: **Example:**

```
def is_even (n):
    if n % 2 == 0:
        print('even')
        return True    # return stops further execution of the function, similar to br
    return False
print(is_even(10)) # True
print(is_even(7)) # False
```

- Returning a list: **Example:**

```
def find_even_numbers(n):
    evens = []
    for i in range(n + 1):
        if i % 2 == 0:
            evens.append(i)
    return evens
print(find_even_numbers(10))
```

Function with Default Parameters

Sometimes we pass default values to parameters, when we invoke the function. If we do not pass arguments when calling the function, their default values will be used.

```
# syntax
# Declaring a function
def function_name(param = value):
    codes
    codes
# Calling function
```

```
function_name()
function_name(arg)
```

Example:

```
def greetings (name = 'Peter'):
    message = name + ', welcome to Python for Everyone!'
    return message
print(greetings())
print(greetings('Asabeneh'))

def generate_full_name (first_name = 'Asabeneh', last_name = 'Yetayeh'):
    space = ' '
    full_name = first_name + space + last_name
    return full_name

print(generate_full_name())
print(generate_full_name('David','Smith'))

def calculate_age (birth_year,current_year = 2021):
    age = current_year - birth_year
    return age;
print('Age: ', calculate_age(1821))

def weight_of_object (mass, gravity = 9.81):
    weight = str(mass * gravity)+ ' N' # the value has to be changed to string first
    return weight
print('Weight of an object in Newtons: ', weight_of_object(100)) # 9.81 - average grav
print('Weight of an object in Newtons: ', weight_of_object(100, 1.62)) # gravity on th
```

Arbitrary Number of Arguments

If we do not know the number of arguments we pass to our function, we can create a function which can take arbitrary number of arguments by adding * before the parameter name.

```
# syntax
# Declaring a function
def function_name(*args):
    codes
    codes
# Calling function
function_name(param1, param2, param3,..)
```

Example:

```
def sum_all_nums(*nums):
    total = 0
    for num in nums:
        total += num     # same as total = total + num
    return total
print(sum_all_nums(2, 3, 5)) # 10
```

Default and Arbitrary Number of Parameters in Functions

```
def generate_groups (team,*args):
    print(team)
    for i in args:
        print(i)
print(generate_groups('Team-1','Asabeneh','Brook','David','Eyob'))
```

Function as a Parameter of Another Function

```
#You can pass functions around as parameters
def square_number (n):
    return n * n
def do_something(f, x):
    return f(x)
print(do_something(square_number, 3)) # 27
```

🌕 You achieved quite a lot so far. Keep going! You have just completed day 11 challenges and you are 11 steps a head in to your way to greatness. Now do some exercises for your brain and muscles.

Testimony

Now it is time to express your thoughts about the Author and 30DaysOfPython. You can leave your testimonial on this link

💻 Exercises: Day 11

Exercises: Level 1

1. Declare a function *add_two_numbers*. It takes two parameters and it returns a sum.
2. Area of a circle is calculated as follows: area = π x r x r. Write a function that calculates *area_of_circle*.
3. Write a function called add_all_nums which takes arbitrary number of arguments and sums all the arguments. Check if all the list items are number types. If not do give a reasonable feedback.
4. Temperature in °C can be converted to °F using this formula: °F = (°C x 9/5) + 32. Write a function which converts °C to °F, *convert_celsius_to-fahrenheit*.
5. Write a function called check-season, it takes a month parameter and returns the season: Autumn, Winter, Spring or Summer.
6. Write a function called calculate_slope which return the slope of a linear equation
7. Quadratic equation is calculated as follows: $ax^2 + bx + c = 0$. Write a function which calculates solution set of a quadratic equation, *solve_quadratic_eqn*.
8. Declare a function named print_list. It takes a list as a parameter and it prints out each element of the list.

9. Declare a function named reverse_list. It takes an array as a parameter and it returns the reverse of the array (use loops).

```
print(reverse_list([1, 2, 3, 4, 5]))
# [5, 4, 3, 2, 1]
print(reverse_list1(["A", "B", "C"]))
# ["C", "B", "A"]
```

10. Declare a function named capitalize_list_items. It takes a list as a parameter and it returns a capitalized list of items
11. Declare a function named add_item. It takes a list and an item parameters. It returns a list with the item added at the end.

```
food_staff = ['Potato', 'Tomato', 'Mango', 'Milk'];
print(add_item(food_staff, 'Meat'))     # ['Potato', 'Tomato', 'Mango', 'Milk','Meat']
numbers = [2, 3, 7, 9];
print(add_item(numbers, 5))      [2, 3, 7, 9, 5]
```

12. Declare a function named remove_item. It takes a list and an item parameters. It returns a list with the item removed from it.

```
food_staff = ['Potato', 'Tomato', 'Mango', 'Milk'];
print(remove_item(food_staff, 'Mango'))  # ['Potato', 'Tomato', 'Milk'];
numbers = [2, 3, 7, 9];
print(remove_item(numbers, 3))  # [2, 7, 9]
```

13. Declare a function named sum_of_numbers. It takes a number parameter and it adds all the numbers in that range.

```
print(sum_of_numbers(5))  # 15
print(sum_all_numbers(10)) # 55
print(sum_all_numbers(100)) # 5050
```

14. Declare a function named sum_of_odds. It takes a number parameter and it adds all the odd numbers in that range.
15. Declare a function named sum_of_even. It takes a number parameter and it adds all the even numbers in that - range.

Exercises: Level 2

1. Declare a function named evens_and_odds . It takes a positive integer as parameter and it counts number of evens and odds in the number.

```
print(evens_and_odds(100))
# The number of odds are 50.
# The number of evens are 51.
```

1. Call your function factorial, it takes a whole number as a parameter and it return a factorial of the number
2. Call your function *is_empty*, it takes a parameter and it checks if it is empty or not
3. Write different functions which take lists. They should calculate_mean, calculate_median, calculate_mode, calculate_range, calculate_variance, calculate_std (standard deviation).

Exercises: Level 3

1. Write a function called is_prime, which checks if a number is prime.
2. Write a functions which checks if all items are unique in the list.
3. Write a function which checks if all the items of the list are of the same data type.
4. Write a function which check if provided variable is a valid python variable
5. Go to the data folder and access the countries-data.py file.

- Create a function called the most_spoken_languages in the world. It should return 10 or 20 most spoken languages in the world in descending order
- Create a function called the most_populated_countries. It should return 10 or 20 most populated countries in descending order.

🎉 CONGRATULATIONS ! 🎉

<< Dia 10 | Dia 12 >>