functions

[1]:
```
def generate_full_name ():
    first_name = 'abc'
    last_name = 'xyz'
    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()
```

```
abc xyz
5
```

1 Function with Parameters

[2]:
```
# syntax
#def function_name(parameter):
def greetings (name):
    message = name + ', welcome to Python for Everyone!'
    return message
print(greetings('abc'))


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
```

```
abc, welcome to Python for Everyone!
100
4
314.0
55
None
5050
None
```

[3]:
```
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))
```

```
Sum of two numbers:  10
Age:  202
```

[4]:
```
def print_fullname(firstname, lastname):
    space = ' '
    full_name = firstname  + space + lastname
    print(full_name)
print(print_fullname(firstname = 'abc', lastname = 'xyz'))

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

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

```
abc xyz
None
5
None
```

[5]:
```
#Returning a string:
def print_name(firstname):
    return firstname
print_name('abc')

def print_full_name(firstname, lastname):
    space = ' '
    full_name = firstname  + space + lastname
    return full_name
print_full_name(firstname='abc', lastname='xyz')
```

[5]:
```
'abc xyz'
```

[6]:
```
#Returning a number:
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))
```

```
5
Age:  200
```

[7]:
```
#Returning a boolean:

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

```
even
True
False
```

[8]:
```
#Returning a list:
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))
```

```
[0, 2, 4, 6, 8, 10]
```

2 Function with Default Parameters

[9]:
```
#syntax
#Declaring a function
#def function_name(param = value):
 #   codes
#    codes
#Calling function
#function_name()
#function_name(arg)

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

def generate_full_name (first_name = 'abc', last_name = 'xyz'):
    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))
```

```
Peter, welcome to Python for Everyone!
abc, welcome to Python for Everyone!
abc xyz
David Smith
Age:  200
```

[10]:
```
#Arbitrary Number of Arguments
# syntax
# Declaring a function
#def function_name(*args):
#   codes
#    codes
# Calling function
#function_name(param1, param2, param3,..)



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
```

[11]:
```
def square_number (n):
    return n * n
def do_something(f, x):
    return f(x)
print(do_something(square_number, 9))
```

```
81
```

3 Lambda Function

[12]:
```
# syntax
#x = lambda param1, param2, param3: param1 + param2 + param2
#print(x(arg1, arg2, arg3))


# Named function
def add_two_nums(a, b):
    return a + b

print(add_two_nums(2, 3))



add_two_nums = lambda a, b: a + b
print(add_two_nums(2,3))


# Self invoking lambda function
```

```
(lambda a, b: a + b)(2,3)

square = lambda x : x ** 2
print(square(3))


cube = lambda x : x ** 3
print(cube(3))

# Multiple variables
multiple_variable = lambda a, b, c: a ** 2 - 3 * b + 4 * c
print(multiple_variable(5, 5, 3))
```

```
5
5
9
27
22
```

[13]:
```
#Lambda Function Inside Another Function

def power(x):
    return lambda n : x ** n

cube = power(2)(3)
print(cube)                # 8
two_power_of_five = power(2)(5)
print(two_power_of_five)  # 32
```

```
8
32
```

4 List Comprehension

[14]:
```
# syntax
#[i for i in iterable if expression]


language = 'Python'
lst = list(language) # changing the string to list
print(type(lst))     # list
print(lst)           # ['P', 'y', 't', 'h', 'o', 'n']

# Second way:
lst = [i for i in language]
print(type(lst)) # list
print(lst)       # ['P', 'y', 't', 'h', 'o', 'n']
```

```
<class 'list'>
['P', 'y', 't', 'h', 'o', 'n']
<class 'list'>
['P', 'y', 't', 'h', 'o', 'n']
```

[15]:
```python
# Generating numbers
numbers = [i for i in range(11)]
print(numbers)                    # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

#  mathematical operations during iteration
squares = [i * i for i in range(11)]
print(squares)                    # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

#  make a list of tuples
numbers = [(i, i * i) for i in range(11)]
print(numbers)                      # [(0, 0), (1, 1), (2, 4), (3, 9), (4, 16),␣
 ↪(5, 25)]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49), (8, 64),
(9, 81), (10, 100)]
```

[16]:
```python
# Generating even numbers
even_numbers = [i for i in range(21) if i % 2 == 0]
print(even_numbers)
# Generating odd numbers
odd_numbers = [i for i in range(21) if i % 2 != 0]
print(odd_numbers)
# Filter numbers: let's filter out positive even numbers from the list below
numbers = [-8, -7, -3, -1, 0, 1, 3, 4, 5, 7, 6, 8, 10]
positive_even_numbers = [i for i in numbers if i % 2 == 0 and i > 0]
print(positive_even_numbers)


list_of_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flattened_list = [ number for row in list_of_lists for number in row]
print(flattened_list)    # [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
[4, 6, 8, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```