Chapter 9:
Functions

What are Functions?

Function Syntax

```
def function_name(parameters):
....# statement
....return [expression] # optional
```

Function Syntax

```
def say_hello():
    pass # this function intentionally left blank
```

Function Syntax

```python
def say_hello():
    print('hello')

say_hello()
# hello
```

Function Syntax

```python
def say_hello(someone):
    print('hello', someone)

say_hello('brandi')
# hello brandi
```

Function Syntax

```
def say_hello(someone):
    print('hello ', someone)

say_hello('brandi', 'pete')
# TypeError: say_hello() takes 1 positional argument but 2 were
given
```

Function Syntax

return

```python
# Function with no return value
def say_hello(someone):
    print('hello ' + someone)

say_hello('clare') # => None
# hello clare
```

return

```python
# Function with return value
def add(a, b):
    return a + b

add(12, 17) # => 29
```

return

```python
# Function with multiple return values
def divide(a, b):
    quotient = a // b
    remainder = a % b
    return quotient, remainder

divide(15, 4) # => (3, 3)

quotient, remainder = divide(15, 4)
print(quotient) # 3
print(remainder) # 3
```

return

```python
# Function with multiple return statements
def divide(a, b):
    if b == 0:
        return
    else:
        quotient = a // b
        remainder = a % b
        return quotient, remainder

divide(15, 0) # => None
```

return

```
type(None) # => <class 'NoneType’>
```

```
thing = None

if thing:
    print("it's something")
else:
    print("it's nothing")
# it’s nothing
```

```
thing = None

if thing is None:
    print("it’s nothing")
else:
    print("it’s something else")
# it’s nothing
```

None

Positional Arguments

```
def menu(entree, dessert, wine):
    print(f"entree: {entree}")
    print(f"dessert: {dessert}")
    print(f"wine: {wine}")

menu('fish', 'cake', 'chardonnay')
# entree: fish
# dessert: cake
# wine: chardonnay
```

Positional Arguments

```python
def menu(entree, dessert, wine):
    print(f"entree: {entree}")
    print(f"dessert: {dessert}")
    print(f"wine: {wine}")

menu('chardonnay', 'fish', 'cake')
# entree: chardonnay
# dessert: fish
# wine: cake
```

Positional Arguments

Keyword Arguments

```python
def menu(entree, dessert, wine):
    print(f"entree: {entree}")
    print(f"dessert: {dessert}")
    print(f"wine: {wine}")

menu(wine='merlot', entree='beef', dessert='ice cream')
# entree: beef
# dessert: ice cream
# wine: merlot
```

Keyword Arguments

```
def menu(entree, dessert, wine):
    print(f"entree: {entree}")
    print(f"dessert: {dessert}")
    print(f"wine: {wine}")

menu('beef', wine='merlot', dessert= 'ice cream’)
# positional first works as expected

menu(wine='merlot', dessert= 'ice cream', 'beef')
# SyntaxError: positional argument follows keyword argument
```

Combining Positional & Keyword Arguments

Keyword-Only Arguments

```python
def menu(*, entree, dessert, wine):
    print(f"entree: {entree}")
    print(f"dessert: {dessert}")
    print(f"wine: {wine}")

menu(wine='sauvignon', entree='curry', dessert='pudding')
# entree: curry
# dessert: pudding
# wine: sauvignon
```

Keyword-Only Arguments

```python
def menu(*, entree, dessert, wine):
    print(f"entree: {entree}")
    print(f"dessert: {dessert}")
    print(f"wine: {wine}")

menu('curry', 'pudding', 'sauvignon')
# TypeError: menu() takes 0 positional arguments but 3 were given
```

Keyword-Only Arguments

Default Parameter Values

```python
def menu(entree, dessert, wine='house red'):
    print(f"entree: {entree}")
    print(f"dessert: {dessert}")
    print(f"wine: {wine}")

menu('pizza', 'cookie')
# entree: pizza
# dessert: cookie
# wine: house red
```

Default Parameter Values

```
def add_to_list(element, list=[]):
    list.append(element)
    print(list)

add_to_list(12)
add_to_list(15)
# Expected:          # Actual:
# [12]               # [12]
# [15]               # [12, 15]
```

Default Parameter Values

```
def add_to_list(element, list=None):
    if list is None:
        list = []
    list.append(element)
    print(list)

add_to_list(12)
add_to_list(15)
# [12]
# [15]
```

Default Parameter Values

Explode & Gather Positional Arguments with *

```python
# args is a tuple
def display(*args):
    print(args)

display()
# ()
```

Gather Positional Arguments

```python
# args is a tuple
def display(*args):
    print(args)

display(2, 4, 8, 9, 15)
# (2, 4, 8, 9, 15)
```

Gather Positional Arguments

```python
# args is a tuple
def display(required1, required2, *args):
    print('need this one:', required1)
    print('need this one too:', required2)
    print('all the rest:', args)

display(2, 4, 8, 9, 15)
# need this one: 2
# need this one too: 4
# all the rest: (8, 9, 15)
```

Gather Positional Arguments

```
def display(number1, number2, number3):
    print(number1, number2, number3)

numbers = (2, 4, 8)

display(*numbers) # like display(2, 4, 8)
# 2 4 8
```

Explode Positional Arguments

```python
# args is a tuple
def display(*args):
    print(args)

numbers = (2, 4, 8)
display(*numbers)
# (2, 4, 8)

more_numbers = (2, 4, 8, 9, 15)
display(*more_numbers)
# (2, 4, 8, 9, 15)
```

Explode Positional Arguments

Explode & Gather Keyword Arguments with **

```python
# kwargs is a dictionary
def display(**kwargs):
    print(kwargs)

display(a=1, b=2, c=3)
# {'a': 1, 'b': 2, 'c': 3}
```

Gather Keyword Arguments

```python
def display(*, a, b, c):
    print(a, b, c)

dictionary = {'a': 1, 'b': 2, 'c': 3}

display(**dictionary) # like display(a=1, b=2, c=3)
# 1 2 3
```

Explode Keyword Arguments

```python
# kwargs is a dictionary
def display(**kwargs):
    print(kwargs)

dictionary = {'a': 1, 'b': 2, 'c': 3}

display(**dictionary)
# {'a': 1, 'b': 2, 'c': 3}
```

Explode Keyword Arguments

Mutable & Immutable Arguments

```python
outside_list = ['i', 'love', 'python']

def mangle(list):
    list[2] = 'ruby!!!'

print(outside_list)
# ['i', 'love', 'python’]

mangle(outside_list)

print(outside_list)
# ['i', 'love', 'ruby!!!']
```

Mutable & Immutable Arguments

Quiz

1 Which of the following is a valid function name?

A. `my_function`

B. `1my_function`

C. `my-function`

D. `_my_function1`

2 What is the output of the following `display_person()` function call?

```
def display_person(*args):
    for i in args:
        print(i)

display_person(name="Emma", age="25")
```

3 What does `None` represent in Python?

A. `an empty value`

B. `a missing or undefined value`

C. `a False value`

D. `a zero value`

4 Given this code, what will be the output of the `print` function call?

```
def update_dict(key, value, my_dict={}):
    my_dict[key] = value
    return my_dict

result1 = update_dict('a', 5)
result2 = update_dict('b', 10)
result3 = update_dict('c', 15, {})
print(result3)
```

1 Which of the following is a valid function name?

A. my_function

B. 1my_function

C. my-function

D. _my_function1

2 What is the output of the following `display_person()` function call?

```
def display_person(*args):
    for i in args:
        print(i)

display_person(name="Emma", age="25")
# TypeError: display_person() got an
unexpected keyword argument 'name'
```

3 What does `None` represent in Python?

A. an empty value

B. a missing or undefined value

C. a False value

D. a zero value

4 Given this code, what will be the output of the `print` function call?

```
def update_dict(key, value, my_dict={}):
    my_dict[key] = value
    return my_dict

result1 = update_dict('a', 5)
result2 = update_dict('b', 10)
result3 = update_dict('c', 15, {})
print(result3)
# {'c': 15}
```