Congratulations! You passed!

Grade received	Latest Submission Grade	To pass
100%	100%	80% or higher

Go to next item

1. For a function named `callable`, how would you define it so that it requires a single argument? **1 / 1 point**

- `callable(arg=required):`
- `callable():`
  `arg`
- `callable(arg):`

Correct
Correct! To require an argument you must define it within the parentheses.

2. What is a correct statement about the order of arguments and keyword arguments in a function? **1 / 1 point**

- Arguments must always go before keyword arguments.
- Functions must have arguments or keyword arguments. Not both.
- Keyword arguments must always go before arguments.

Correct
Correct! Although you can use a mix of arguments and keyword arguments, you must always place arguments first.

3. Given the following function: **1 / 1 point**

```
def simple():
    print("this is a function")
```

What would the value of **result** be when assigned in this way:

```
result = simple()
```

- ○ **result** would be **"this is a function"**
- ○ **result** would be **False**
- ◉ **result** would be **None**

✓ **Correct**
Correct. Because the function is not returning any values, the implicit return value of **None** would be used.

4. What is a correct statement about variable arguments? **1 / 1 point**

- ◉ Variable arguments can be used as a single variable of type **tuple**
- ○ Variable arguments must be assigned a value
- ○ Variable arguments must be of the same type

✓ **Correct**
Correct. If using the variable argument as a single variable it would be of type **tuple**.

5. What is one false statement about keyword arguments? **1 / 1 point**

- ○ Keyword arguments can be assigned any type as value
- ○ Keyword arguments are of type **dictionary**
- ◉ Keyword arguments are of type **tuple**

Correct
That's right, this statement is false because keyword arguments are of type **`dictionary`**.

6. With the following code, what would be the result of running it with Python? **1 / 1 point**

```
class Dog:
    def bark():
        print("woof!")

dog = Dog()
dog.bark()
```

- A **`SyntaxError`** would be raised because the **`Dog()`** class isn't using the **`Dog(object)`** signature for classes
- **`woof!`** would be printed
- It would cause a **`TypeError`** exception because the **`bark()`** method is missing **`self`**

Correct
Correct! Because the **`self`** argument wasn't used, this call would cause an exception.

7. What is one problem to be aware of class attributes? **1 / 1 point**

- They can cause higher memory consumption
- That the value can mutate even for other objects coming from the same class
- Once defined, they can't be changed in the **`__init__`** method

Correct
Correct. Class attributes can mutate other objects created from the same class.

8. What is **`self`** in Python methods? **1 / 1 point**

- (●) It is a required argument for classes that refer to the current object
- ( ) It allows you to refer to other parent classes when using inheritance
- ( ) You must use **`self`** for methods, a special keyword for using methods in classes

✓ **Correct**
Correct. This is a requirement for methods.

9. What are Python modules? **1 / 1 point**

- ( ) They are libraries from Python you can import for code reuse and extensibility
- ( ) These are projects that can be imported later for code reuse.
- (●) Python modules are .py files where one can put functions, classes, and any other valid Python code.

✓ **Correct**
Correct! A module is a Python file.

10. What is this piece of code useful in a Python script? **1 / 1 point**

```
if __name__ == '__main__':
```

- (●) So that it can execute a specific piece of code when running with Python as a script.
- ( ) It is a special way of handling imports at the bottom of a Python file
- ( ) It is a way of finding the current path of the script so that it can be executed in the terminal.

✓ **Correct**
Correct. This would allow you to select exactly what and how to run when running a Python file in the terminal.