

TABLE OF CONTENTS

- ▶ **1. Passing Multiple Arguments to a Function**
- 2. Using the Python args Variable in Function Definitions
- 3. Using the Python kwargs Variable in Function Definitions
- 4. Ordering Arguments in a Function
- 5. Unpacking With the Asterisk Operators: * & **
- 6. Conclusion

TABLE OF CONTENTS

1. Passing Multiple Arguments to a Function
- ▶ 2. **Using the Python args Variable in Function Definitions**
3. Using the Python kwargs Variable in Function Definitions
4. Ordering Arguments in a Function
5. Unpacking With the Asterisk Operators: * & **
6. Conclusion

TABLE OF CONTENTS

1. Passing Multiple Arguments to a Function
2. Using the Python args Variable in Function Definitions
- ▶ **3. Using the Python kwargs Variable in Function Definitions**
4. Ordering Arguments in a Function
5. Unpacking With the Asterisk Operators: * & **
6. Conclusion

TABLE OF CONTENTS

1. Passing Multiple Arguments to a Function
2. Using the Python args Variable in Function Definitions
3. Using the Python kwargs Variable in Function Definitions
- ▶ **4. Ordering Arguments in a Function**
5. Unpacking With the Asterisk Operators: * & **
6. Conclusion

TABLE OF CONTENTS

1. Passing Multiple Arguments to a Function
2. Using the Python args Variable in Function Definitions
3. Using the Python kwargs Variable in Function Definitions
4. Ordering Arguments in a Function
- ▶ 5. **Unpacking With the Asterisk Operators: * & ****
6. Conclusion

TABLE OF CONTENTS

1. Passing Multiple Arguments to a Function
2. Using the Python args Variable in Function Definitions
3. Using the Python kwargs Variable in Function Definitions
4. Ordering Arguments in a Function
5. Unpacking With the Asterisk Operators: * & **



6. Conclusion

THE CORRECT ORDER FOR YOUR PARAMETERS

1. Standard arguments

THE CORRECT ORDER FOR YOUR PARAMETERS

1. Standard arguments
2. *args arguments

THE CORRECT ORDER FOR YOUR PARAMETERS

1. Standard arguments
2. `*args` arguments
3. `**kwargs` arguments

UNPACKING OPERATORS

1. * & ** unpacking operators were introduced in Python 2

UNPACKING OPERATORS

1. * & ** unpacking operators were introduced in Python 2
2. As of 3.5, they became even more powerful, thanks to PEP 448

UNPACKING OPERATORS

1. * & ** unpacking operators were introduced in Python 2
2. As of 3.5, they became even more powerful, thanks to PEP 448
3. They are operators that unpack the values from iterable objects in Python

UNPACKING OPERATORS

1. * & ** unpacking operators were introduced in Python 2
2. As of 3.5, they became even more powerful, thanks to PEP 448
3. They are operators that unpack the values from iterable objects in Python
4. * can be used on any iterable that Python provides

UNPACKING OPERATORS

1. `*` & `**` unpacking operators were introduced in Python 2
2. As of 3.5, they became even more powerful, thanks to PEP 448
3. They are operators that unpack the values from iterable objects in Python
4. `*` can be used on any iterable that Python provides
5. `**` can only be used on dictionaries

WHAT YOU HAVE LEARNED

- What `*args` and `**kwargs` actually mean

WHAT YOU HAVE LEARNED

- What `*args` and `**kwargs` actually mean
- How to use `*args` and `**kwargs` in function definitions

WHAT YOU HAVE LEARNED

- What `*args` and `**kwargs` actually mean
- How to use `*args` and `**kwargs` in function definitions
- How to use a single asterisk (`*`) to unpack iterables

WHAT YOU HAVE LEARNED

- What `*args` and `**kwargs` actually mean
- How to use `*args` and `**kwargs` in function definitions
- How to use a single asterisk (`*`) to unpack iterables
- How to use two asterisks (`**`) to unpack dictionaries