

Error Handling

Textbook

Error Handling



In programming, we often run into bugs or problems in our programs. Odds are, you have experienced this first hand in your own programs! Programmers are no strangers to bugs in their programs. They use methods to better manage these problems. Much like a beekeeper takes precautions before working with his bees, a programmer can take precautions with their code.

In order to handle the possibility of errors in their code, programmers organize code in a way that helps to capture and manage bugs. This code organization is called error handling.

The organization of code is broken down into 4 basic blocks of code:

Name of Code Block	What it does
The try block	Tests a code for errors
The except block	Code to run if an error happens
The else block	Code to run if no error happens

The [finally_block](#)

Code to run regardless of the try/except blocks.

Exception Handling

When a program runs into a bug, we don't want the program to crash and cause issues. Have you ever tried to go to a website or run a software that has said something like, "We are experiencing a problem, try again later." This is because when the program ran into a problem, instead of crashing the computer, it merely displays a message. This is called [exception](#) (or error) handling. How will your program handle a problem?

To handle [exceptions](#), we use what's called the [try block](#).

Here is an example where we try to print a variable named greeting that hasn't been created yet.

```
1 try:
2     print(greeting)
3 except:
4     print("A problem occurred")
```

Try it!

This will return the string `A problem occurred` because the try block ran into a problem.

Without the [try block](#), this code would crash since there is no variable named greeting.

```
1 print(greeting)
```

Try it!

Multiple Except Blocks

A program can have multiple except blocks. **Important!** Only one except branch can be executed. Other except blocks need to check for a certain error. A program might have many except blocks, but only one will run.

```
1 try:
2     print(greeting)
3 except NameError:
4     print("A problem occurred")
5 except:
6     print("A different problem occurred")
```

Try it!

The default except branch should be last.

Else

The [else block](#) can be included to run if there are no errors.

```
1 greeting = "Hello"
2
3 try:
4     print(greeting)
5 except:
6     print("A problem occurred")
7 else:
8     print("The code worked without a problem")
```

Try it!

This code would return `Hello` and `The code worked without a problem`. Since the area where print statements show up (the console) is just information for the programmer, we can easily see that the code worked correctly.

The [else block](#) is optional and **MUST** come right after the [except block](#).

The else block is sometimes called the default block.

Finally

The [finally block](#) will run no matter what. It will run regardless of the results of the try/except/else blocks. The [finally block](#) is optional.

```
1 greeting = "Hello"
2
3 try:
4     print(greeting)
5 except:
6     print("A problem occurred")
7 else:
8     print("The code worked without a problem")
9 finally:
10    print("Your exception handling is complete")
```

Try it!

If any of the raised exceptions doesn't have a print statement or other resolution, the program quits and throws an error.

Why Error Handling is Important

You might wonder why this kind of organization would be important. As you might have experienced, sometimes finding the bugs in code can be challenging. In programs that have hundreds of lines of code, it can be difficult or near impossible to figure out where the problem might be.

With error handling, the code is all separated down into chunks that help narrow down where the bug is. Many programs often have many try/except blocks to help organize where the bugs are. Error handling helps programmers handle errors more easily.

Checkpoint

Error Handling

Practice adding some error handling to your code!

Create a variable and set it equal to a string.

Add the following blocks:

try

except

else

finally

For each of the blocks, add a print statement.

Requirements:

- Create a variable and set it equal to a string.
- Add a try block
- Add an except block
- Add an else block
- Add a finally block

Questions (8)

1. Which code block checks a code for errors?

MULTIPLE CHOICE

Choose the correct answer:

- A. Try
- B. Except
- C. Else
- D. Finally

2. Which code block runs no matter if there is an error or not?

MULTIPLE CHOICE

Choose the correct answer:

- A. Except
- B. Else
- C. Finally
- D. Test

SELECT MULTIPLE

3. Which code block only runs if there is no error in the code? Select all that apply.

Select all that apply:

- A. Try
- B. Except
- C. Else
- D. Finally

MULTIPLE CHOICE

4. Which code block only runs if there is an error in the code?

Choose the correct answer:

- A. Try
- B. Except
- C. Else
- D. Finally

MULTIPLE CHOICE

5. What will the following code print out?

```
friend = "Simon" try: print(pal) except: print("A problem occurred") else: print("The code worked without a problem")
```

Choose the correct answer:

- A. A problem occurred
- B. The code worked without a problem
- C. pal
- D. Simon

MULTIPLE CHOICE

6. What will the following code print out?

```
friend = "Isabella" try: print(friend) except: print("A problem occurred") else: print("The code worked without a problem")
```

Choose the correct answer:

- A. Isabella
- B. The code worked without a problem
- C. Isabella The code worked without a problem
- D. A problem occurred

7. What will the following code print out?

```
friend = "Taj"
try: print(brother)
except: print("A problem occurred")
else: print("The code worked without a problem")
finally: print("Error handling is complete")
```

Choose the correct answer:

- A. A problem occurred
- B. The code worked without a problem
- C. Error handling is complete
- D. Taj The code worked without a problem Error handling is complete
- E. A problem occurred Error handling is complete
- F. The code worked without a problem Error handling is complete

8. Edit the text box below to debug (fix) the code:

Code to Debug:

```
1 try:
2     print(greeting)
3 except:
4     print("A problem occurred")
```

Challenges (3)

1. Pizza Party

Imagine you are planning a pizza party. You are trying to figure out how many slices everyone will get. This depends on how many people will be there and how many slices of pizza you have total.

Create 2 inputs. 1 input will be the number of slices of pizza. 1 input will be the number of people at the party.

This program has a potential problem. If you entered `0` as the number of people at the party, you will run into an error, since you cannot divide by zero.

Add exception handling to look for this potential problem.

Use a **try block** on a code that will print out the number of slices per person.

Add an **except block** that will print out `Your code doesn't account for if a user tries to enter 0 people`

Add an **else block** that will print out `There is no problem`

Add a **finally block** that will print out `Your exception handling is complete`

Requirements:

- Create 2 inputs
- Add a try block that does the calculation to find the slices per person.
- Add an except block that prints "Your code doesn't account for if a user tries to enter 0 people"
- Add an else block that prints "There is no problem"
- Add a finally block that prints "Your exception handling is complete"

2. Sunflower Seeds

You are a sunflower farmer. You are trying to find out how many seeds your farm generates. For this challenge, assume that each sunflower generates 300 seeds.

Create a program that takes in the number of sunflowers as an input.

The program will calculate how many seeds the farm generates.

Add exception handling to the code to see if there're any problems.

Use a **try block** on a code that will print out the total number of sunflower seeds.

Add an **except block** that will print out `There is a problem`

Add an **else block** that will print out `There is no problem`

Add a **finally block** that will print out `Your exception handling is complete`

Requirements:

- Create 1 input
- Add a try block that does the calculation to find the total number of sunflower seeds
- Add an except block that prints "There is a problem"
- Add an else block that prints "There is no problem"
- Add a finally block that prints "Your exception handling is complete"

3. Ninja Name Generator

Have you ever heard of some cool names that Ninjas might have? Create a program that generates a Ninja name for the user.

Create a program that has **5 inputs** that ask the user for the following:

1. What is the color of your shirt?
2. What object is closest to your ear right now?
3. What object is closest to your elbow right now?
4. What were you doing at 5pm yesterday?
5. How many pairs of socks do you own? (Enter as a number)

The program will then print out the person's Ninja name by concatenating all these answers together. The fifth input must first be multiplied by 10 to make the Ninja name more epic. (Note, for this challenge use this syntax `_____ * 10`)

You notice in your code that it's possible that a person could enter a spelled out number for input 5. Create a try block to check for this possibility.

Use a **try block** on a code that will print out the Ninja name

Add an **except block** that will print out `User entered an invalid number`

Add an **else block** that will print out `There is no problem`

Add a **finally block** that will print out `Your exception handling is complete`

Hint: Put the inputs inside of your try block

Requirements:

- Inside the try block Create 5 inputs
- Inside the try block, add code that does the calculation on the fifth input and concatenates the answers together.
- Add an except block that prints "User entered an invalid number"
- Add an else block that prints "There is no problem"
- Add a finally block that prints "Your exception handling is complete"

Answer Keys & Solutions

Checkpoint Solutions

Error Handling

```
1 morning = "cockadoodle doo!"
2
3 try:
4     print(greeting)
5 except:
6     print("There is a problem")
7 else:
8     print("There is no problem")
9 finally:
10    print("Your exception handling is complete")
```

Questions

1. Which code block checks a code for errors?

MULTIPLE CHOICE

Correct Answer:

- | | |
|------------|-------------|
| A. Try | ✓ Correct |
| B. Except | ✗ Incorrect |
| C. Else | ✗ Incorrect |
| D. Finally | ✗ Incorrect |

Explanation:

This block tries out code to see if it runs okay.

2. Which code block runs no matter if there is an error or not?

MULTIPLE CHOICE

Correct Answer:

- | | |
|------------|-------------|
| A. Except | ✗ Incorrect |
| B. Else | ✗ Incorrect |
| C. Finally | ✓ Correct |

D. Test

✗ Incorrect

Explanation:

This code block comes at the end of the error handling.

3. Which code block only runs if there is no error in the code? Select all that apply.

SELECT MULTIPLE

Correct Answers:

A. Try

✓ Correct

B. Except

✗ Incorrect

C. Else

✓ Correct

D. Finally

✗ Incorrect

Explanation:

With error handling, having no error is the try or else statement.

4. Which code block only runs if there is an error in the code?

MULTIPLE CHOICE

Correct Answer:

A. Try

✗ Incorrect

B. Except

✓ Correct

C. Else

✗ Incorrect

D. Finally

✗ Incorrect

Explanation:

This code runs if there is an error (or exception)

5. What will the following code print out?

MULTIPLE CHOICE

Correct Answer:

A. A problem occurred

✓ Correct

B. The code worked without a problem

✗ Incorrect

C. pal

✗ Incorrect

D. Simon

✗ Incorrect

Explanation:

The try block is trying to print a variable that doesn't exist.

6. What will the following code print out?

MULTIPLE CHOICE

Correct Answer:

A. Isabella

✗ Incorrect

B. The code worked without a problem

✗ Incorrect

C. Isabella The code worked without a problem

✓ Correct

D. A problem occurred

✗ Incorrect

Explanation:

If the code runs correctly, the try block and the else block will run.

7. What will the following code print out?

MULTIPLE CHOICE

Correct Answer:

A. A problem occurred

✗ Incorrect

B. The code worked without a problem

✗ Incorrect

C. Error handling is complete

✗ Incorrect

D. Taj The code worked without a problem Error handling is complete

✗ Incorrect

E. A problem occurred Error handling is complete

✓ Correct

F. The code worked without a problem Error handling is complete

✗ Incorrect

Explanation:

The finally block always runs. If no error happens, the try block, else block, and finally block all run.

8. Edit the text box below to debug (fix) the code:

DEBUG CODE

Incorrect Code:

```
1 try:
2     print(greeting)
3 except;
4     print("A problem occurred")
```

Correct Solution:

```
1 try:
2     print(greeting)
3 except:
4     print("A problem occurred")
```

Explanation:

Don't forget colons after the try and except blocks.

Challenges

1. Pizza Party

Solution:

```
1 slices = int(input("How many slices of pizza?"))
2 people = int(input("How many people?"))
3
4 try:
5     total = slices/people
6     print(total)
7 except:
8     print("Your code doesn't account for if a user tries to enter 0 people")
9 else:
10    print("There is no problem")
11 finally:
12    print("Your exception handling is complete")
```

2. Sunflower Seeds

Solution:

```
1 sunflower = int(input("How many sunflowers do you have?"))
2
3
4 try:
```

```
5     total = sunflower * 300
6     print(total)
7 except:
8     print("There is a problem")
9 else:
10    print("There is no problem")
11 finally:
12    print("Your exception handling is complete")
```

3. Ninja Name Generator

Solution:

```
1 try:
2     shirt = input("What is the color of your shirt?")
3     ear = input("What object is closest to your elbow right now?")
4     action = input("What were you doing at 5pm yesterday?")
5     number = int(input("How many pairs of socks do you own? (Enter as a number)"))
6     num = number * 10
7     print(shirt + ear + action + "er" + str(num))
8 except:
9     print("User entered an invalid number")
10 else:
11     print("There is no problem")
12 finally:
13     print("Your exception handling is complete")
```