Defining terms

Error handling

Debugging

Error types

- Syntax errors
- Runtime errors
- Logic errors

Syntax errors

```
# This code won't run at all
x = 42
y = 206
if x == y
    print('Success!!')
```

```
# output
File "syntax.py", line 3
   if x == y
           ^


SyntaxError: invalid syntax
```

Runtime errors

```python
# This code will fail when run
x = 42
y = 0
print(x / y)
```

```
# output
Traceback (most recent call last):
  File "runtime.py", line 3, in <module>
    print(x / y)
ZeroDivisionError: division by zero
```

Catching runtime errors

```
try:
    print(x / y)
except ZeroDivisionError as e:
    # Optionally, log e somewhere
    print('Sorry, something went wrong')
except:
    print('Something really went wrong')
finally:
    print('This always runs on success or failure')
```

```
# output
Sorry, something went wrong
```

When to use try/except/finally

When something might go wrong

User input

Accessing an external system

- REST call
- File system

You can act upon the error

Logging

Graceful exit

Some final words on try/except/finally

Not used to find bugs

Debugging, not error handling

You don't have to catch all errors

Let it bubble up

Someone else will deal with it

The application will crash

- Sometimes, this is exactly what you want to happen

Logic errors

```python
# This code won't run at all
x = 206
y = 42
if x < y:
    print(str(x) + ' is greater than ' + str(y))
```

```
# output
```

Figuring out what went wrong

Stack trace

- Last calls are on the top
- Your code is likely on the bottom
- Seek out line numbers

Finding your mistake

- Reread your code
- Check the documentation
- Search the internet
- Take a break
- Ask someone for help