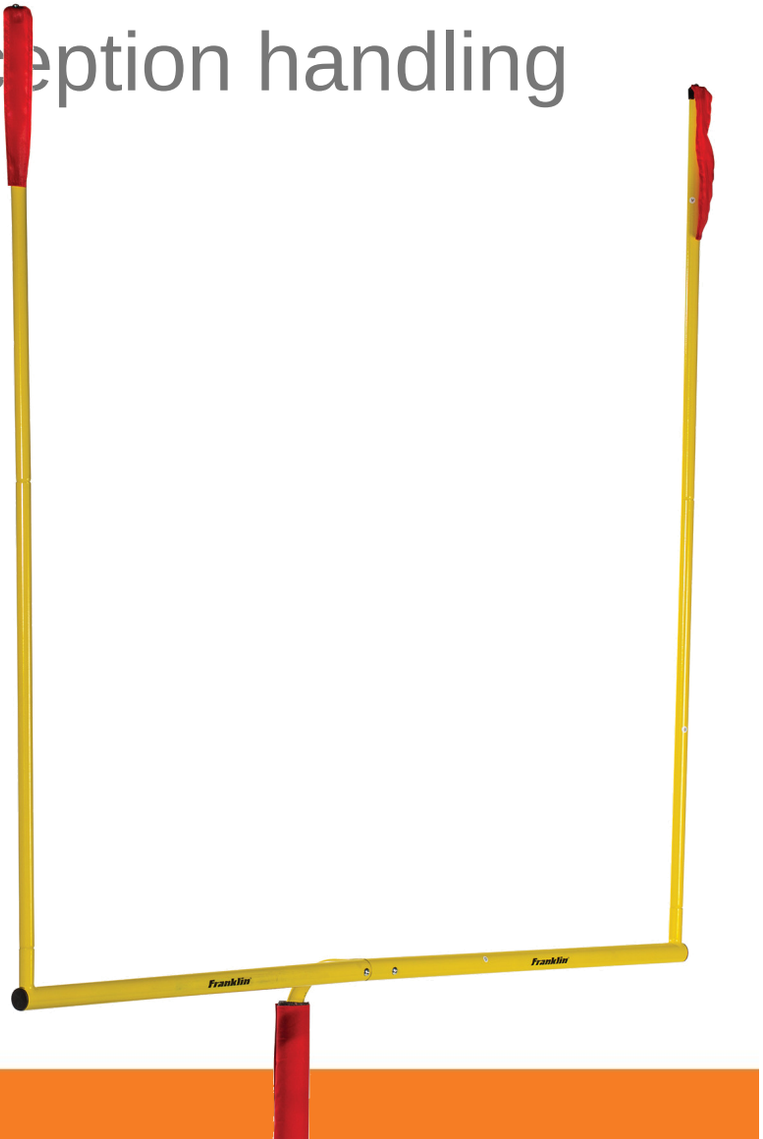
Exceptions

Goals

1. Name some pros and cons of structured exception handling
2. Explain when to raise an exception
3. Explain when to catch an exception



Roadmap

1. Overview
2. Raising
3. Catching

Develop
Intelligence

Overview

Develop Intelligence

What's an Exception?

Exception handling *is the process of responding to the occurrence of exceptions – anomalous conditions requiring special processing – during the execution of a program. An exception breaks the normal flow of execution and executes a pre-registered exception handler.*

Not Everyone's On Board

The reasoning is that I consider exceptions to be no better than “goto’s”, considered harmful since the 1960s, in that they create an abrupt jump from one point of code to another. In fact they are significantly worse than goto’s.

- Exceptions break the normal flow
- You're not forced to deal with them
- Alternatives include
  - Error messages
  - Maybe type

Motivation

- Without exceptions, callers must check return values

```
1  def divide(numerator, denominator):
2    if denominator == 0:
3      return False, None
4    return True, numerator/denominator
5
6  success, quotient = divide(x, y)
7  if not success:
8    handle_error()
9    return
10
11 # On my merry way...
12 print(quotient)
```

Motivation (Alternative)

```python
@dataclass(frozen=True)
class DivisionResult:
  failed: bool
  failureMessage: string
  quotient: float

def divide(numerator, denominator):
  if denominator == 0:
    return DivisionResult(True,'Divide by zero',None)
  return DivisionResult(False,'', numerator/denominator)
```

Develop
Intelligence

Raising

Develop Intelligence

Keyword: `raise`

- Exits the function
- Results
  - Drills down the stack to find a handler
  - Stops execution

```
1 def divide(numerator, denominator):
2    if denominator == 0:
3      raise ValueException('Denominator cant be 0.')
4   return numerator / denominator
```

Built-in Exceptions

- TypeError
- ValueError
- NotImplementedError

Defensive Programming

```
def save_file(name, contents):
  if not isinstance(name,string):
    raise TypeError('name must be of type string')
  if not isinstance(contents, string):
    raise TypeError('contents must be of type string')
  if len(name) > 10:
    raise ValueError("File name too long")

  # Do file stuff here
```

- Without static type checking, argument validation is the means of argument validation

Best Practices

- Fail early
- Belt + braces
- Use testing to catch things producing exceptions

Develop
Intelligence

Catching

Develop Intelligence

Syntax: `try... except`

```
1  def divide(numerator, denominator):
2    if denominator == 0:
3       raise ValueException('Denominator cant be 0.')
4    return numerator / denominator
5
6  x,y = 1,1
7
8  quotient = None
9
10 try:
11   quotient = divide(x, y)
12 except:
13   print('Got an exception!')
14
15
```

Getting Info

- Use the `as` keyword for access to the raised exception object

```
x,y = 1,0
try:
  quotient = x/y
except Exception as ex:
  print(ex)

print(f'Your quotient is: {quotient}')
```

Bad Ideas

- Don't 'swallow' exceptions
  - Hides errors
  - Might bork performance

```
x,y = 1,1

try:
  quotient = x/y
except:
  pass

print(f'Your quotient is: {quotient}')
```

Catching by Type

- Some exceptions you can handle
- Some you can't

```
while True:
  try:
    x = int(input("Please enter a number: "))
    break
  except (ValueError, TypeError):
    print("Oops!  That was no valid number.  Try again...")
```

Catching by Type (Alternative)

Develop Intelligence

```
while True:
  try:
    x = int(input("Please enter a number: "))
    break
  except ValueError as ex:
    print("Oops!  That was no valid number.  Try again...")
  except TypeError as ex:
    print("Wrong type, dude")
```

Finally

- Finally block runs no matter what
- Used for cleaning up resources

```
con = db_connection(name, password)

try:
  con.open()
  con.execute_scalar('DELETE dbo.User')
finally:
  con.close()
```

Best Practices

- Exceptions can be slow
- Don't use exceptions for flow control
- Only catch exceptions that you can do something about
- Most scripts need at least 1 exception in `main()`

Develop
Intelligence

Review

1. Name some pros and cons of structured exception handling
2. Explain when to raise an exception
3. Explain when to catch an exception