I take exception to your exceptions

Using custom errors to get your point across

Joe Kaufeld, @itsthejoker on github and fosstodon

General Overview

- Intro (🔴 you are here)
- Exceptions in Python
- Custom Exceptions
- Using exceptions as part of your architecture
- Fun with exceptions

What is an exception?

Definition time!

An exception is a error.

~~Definition~~ Redefinition time!

An exception is a error notifying that a logical failure has occurred.

Which is more useful?

```
TypeError: unsupported operand type(s) for +: 'int' and `str'
```

VS

```
ConfigurationParseError: The value `appending_data` must have
                    a `str` ending, not `int`.
```

Building a Custom Exception

```
class ConfigurationParseError(Exception):
    pass
```

Building a Custom Exception

```
class ConfigurationParseError(Exception):
    pass


raise ConfigurationParseError(“Oops.”)
```

How can we use custom exceptions?

How can we use Custom Exceptions?

starting with the base function

```
def get_data() -> dict:
  resp = httpx.get("https://dummyjson.com/products/1")
  resp.raise_for_status()
  return resp.json()
```

How can we use Custom Exceptions?

guard clauses for fun and profit

```
def get_data() -> dict:
  resp = httpx.get("https://dummyjson.com/products/1")
  resp.raise_for_status()
  return resp.json()
```

```
...
mydata = get_data()
if mydata:
  do_stuff(mydata)
```

How can we use Custom Exceptions?

truly exceptional

```
def get_data() -> dict:
  try:
    resp = httpx.get("https://dummyjson.com/products/1")
    resp.raise_for_status()
  except httpx.HTTPError:
    raise DummyJSONConnectionError

  if not resp.json():
    raise EmptyResponseError("No data returned!")
  return resp.json()
```

How can we use Custom Exceptions?

once more with feeling

```
def get_data() -> dict:
  try:
    resp = httpx.get("https://dummyjson.com/products/1")
    resp.raise_for_status()
  except httpx.HTTPError:
    raise DummyJSONConnectionError

  try:
    if not resp.json():
      raise EmptyResponseError("No data returned!")
  except json.decoder.JSONDecodeError:
    raise GarbageResponseError
return resp.json()
```

How can we use Custom Exceptions?

a breath of contextual fresh air

From…

* TimeoutException
* ConnectTimeout
* ReadTimeout
* ConnectError
* ReadError
* DecodingError
* HTTPStatusError
* JSONDecodeError

To…

* DummyJSONConnectionError
* EmptyResponseError
* GarbageResponseError

How can we use Custom Exceptions?

once more with feeling

```
def get_data() -> dict:
  try:
    resp = httpx.get(“https://dummyjson.com/products/1")
    resp.raise_for_status()
  except httpx.HTTPError as e:
    raise DummyJSONConnectionError from e

  try:
    if not resp.json():
      raise EmptyResponseError(“No data returned!”)
  except json.decoder.JSONDecodeError as e:
    raise GarbageResponseError from e
return resp.json()
```

Using the Function

the easy way

```
try:
  mydata = get_data()

except (
  DummyJSONConnectionError,
  EmptyResponseError,
  GarbageResponseError
) as e:

  my_data = get_default_data()
```

Using the Function

the cooler way

```
try:
  mydata = get_data()

except (
  DummyJSONConnectionError,
  GarbageResponseError
) as e:
  sentry.capture_event(e, “they violated SLA again lol”)
  raise ApplicationError(“Cannot connect.”)

except EmptyResponseError:
  my_data = get_default_data()
```

The cool stuff

Cool Stuff You Might Not Know

try / except / else

```
try:
  thing_that_didnt_explode()

except:
  print(“Crap.”)

else:
  print(“Yay! We made it!”)
```

Cool Stuff You Might Not Know

try / except / finally

```
try:
  thing_that_didnt_explode()

except:
  print(“Crap.”)

finally:
  print(“This is the end. I will always run.”)
```

Cool Stuff You Might Not Know

ExceptionGroups

```
exceptions = []

exceptions.append(
  OSError(“Your computer might be on fire.")
)
exceptions.append(
  ZeroDivisionError(“Math is hard.”)
)

raise ExceptionGroup(“Errors everywhere, man.”, exceptions)
```

Cool Stuff You Might Not Know

grabbing single exceptions out of a group

```
try:
  raise ExceptionGroup(
    “oops.”,
    [OSError(), TypeError()]
  )
except* OSError:
  print(“We managed to save this one...”)

# boom for the TypeError
```

Cool Stuff You Might Not Know

adding additional context to exceptions

```
boom = OSError("I'm sorry, Dave.")

boom.add_note("I'm afraid I can't let you do that.")
boom.add_note(
  "I think you know what the problem is just as well as I do."
)

raise boom
```

that’s all I’ve got, thanks for your time

I hope you learned something cool

Joe Kaufeld, @itsthejoker on github and fosstodon

Get the slides here!