Errors and Exceptions

?????

- A file doesn’t exist
- A network or database connection fails
- Your code receives invalid input

Try-Except

- `except Exception` : ????
- `print(, file=sys.stderr)` : ? STDERR ????

```
def main():
    if len(sys.argv) < 2:
        return usage()


    try:
        date, title, emails = sys.argv[1].split('|')
        message = message_template(date, title)
        send_message(message, emails)
        print("Successfully sent reminders to:", emails)
    except Exception as e:
        print("Failure to send email", file=sys.stderr)
    except Exception as e:
       print("Failure to send email: {}".format(e), file=sys.stderr)
```

- `except OSError`

```
def character_frequency(filename):
  """Counts the frequency of each character in the given file."""
  # First try to open the file
  try:
    f = open(filename)
  except OSError:
    return None
```

```
  # Now process the file
  characters = {}
  for line in f:
    for char in line:
      characters[char] = characters.get(char, 0) + 1
  f.close()
  return characters
```

- `finally`

```
def calculate_average(numbers):
    try:
        return sum(numbers) / len(numbers)
    except TypeError:
        raise InvalidInputError(f"Expected a list or tuple, but got {type(numbers)}")
    except ZeroDivisionError:
        raise EmptyInputError("The list is empty. Cannot calculate the average.")
    finally:
        print("Execution of calculate_average function completed.")
```

Raise

- `raise ValueError("Some custom error messages")`

```
def validate_user(username, minlen):
  assert type(username) == str, "username must be a string"
  if minlen < 1:
    raise ValueError("minlen must be at least 1")

  if len(username) < minlen:
    return False
  if not username.isalnum():
    return False
  return True
```

For unit test

- `.assertRaises()`

```
import unittest

from validations import validate_user

class TestValidateUser(unittest.TestCase):
  def test_valid(self):
    self.assertEqual(validate_user("validuser", 3), True)

  def test_too_short(self):
    self.assertEqual(validate_user("inv", 5), False)

  def test_invalid_characters(self):
    self.assertEqual(validate_user("invalid_user", 1), False)

  def test_invalid_minlen(self):
    self.assertRaises(ValueError, validate_user, "user", -1)



# Run the tests
unittest.main()
```

- FileNotFoundError : The file might not exist
- IndexError : The file might not have enough lines of data
- ValueError : The data in the file might not be convertible to integers
- ZeroDivisionError : The second number might be zero

```
def enhanced_read_and_divide(filename):
    try:
        with open(filename, 'r') as file:
            data = file.readlines()
            
            # Ensure there are at least two lines in the file
            if len(data) < 2:
                raise ValueError("Not enough data in the file.")
            
            num1 = int(data[0])
            num2 = int(data[1])
            
            # Check if second number is zero
            if num2 == 0:
```

```
            raise ZeroDivisionError("The denominator is zero.")
        
        return num1 / num2



except FileNotFoundError:
    return "Error: The file was not found."
except ValueError as ve:
    return f"Value error: {ve}"
except ZeroDivisionError as zde:
    return f"Division error: {zde}"
```

Examples

User's emails

user_emails.csv :

```
Blossom Gill,blossom@abc.edu
Hayes Delgado,nonummy@abc.edu
Petra Jones,ac@abc.edu
Oleg Noel,noel@abc.edu
Ahmed Miller,ahmed.miller@abc.edu
Macaulay Douglas,mdouglas@abc.edu
Aurora Grant,enim.non@abc.edu
Madison Mcintosh,mcintosh@abc.edu
Montana Powell,montanap@abc.edu
Rogan Robinson,rr.robinson@abc.edu
Simon Rivera,sri@abc.edu
Benedict Pacheco,bpacheco@abc.edu
Maisie Hendrix,mai.hendrix@abc.edu
Xaviera Gould,xlg@abc.edu
Oren Rollins,oren@abc.edu
Flavia Santiago,flavia@abc.edu
Jackson Owens,jacksonowens@abc.edu
Britanni Humphrey,britanni@abc.edu
Kirk Nixon,kirknixon@abc.edu
Bree Campbell,breee@abc.edu
```

emails.py : Main program

```
#!/usr/bin/env python3

import sys
import csv

def populate_dictionary(filename):
  """Populate a dictionary with name/email pairs for easy lookup."""
  email_dict = {}
  with open(filename) as csvfile:
    lines = csv.reader(csvfile, delimiter = ',')
    for row in lines:
      name = str(row[0].lower())
      email_dict[name] = row[1]
  return email_dict

def find_email(argv):
  """ Return an email address based on the username given."""
  # Create the username based on the command line input.
  try:
    fullname = str(argv[1] + " " + argv[2])
    # Preprocess the data
    email_dict = populate_dictionary('/home/student/data/user_emails.csv')
    # Find and print the email
    if email_dict.get(fullname.lower()):
      return email_dict.get(fullname.lower())
    else:
      return "No email address found"
  except IndexError:
    return "Missing parameters"

def main():
  print(find_email(sys.argv))

if __name__ == "__main__":
  main()
```

emails_test.py : For unit test

```
#!/usr/bin/env python3
import unittest
```

```
from emails import find_email

class EmailsTest(unittest.TestCase):
  def test_basic(self):
    testcase = [None, "Bree", "Campbell"]
    expected = "breee@abc.edu"
    self.assertEqual(find_email(testcase), expected)

  def test_one_name(self):
    testcase = [None, "John"]
    expected = "Missing parameters"
    self.assertEqual(find_email(testcase), expected)

  def test_two_name(self):
    testcase = [None, "Roy", "Cooper"]
    expected = "No email address found"
    self.assertEqual(find_email(testcase), expected)

if __name__ == '__main__':
  unittest.main()
```

Revision #9
Created 5 December 2024 16:55:39 by Admin
Updated 17 December 2024 16:21:19 by Admin