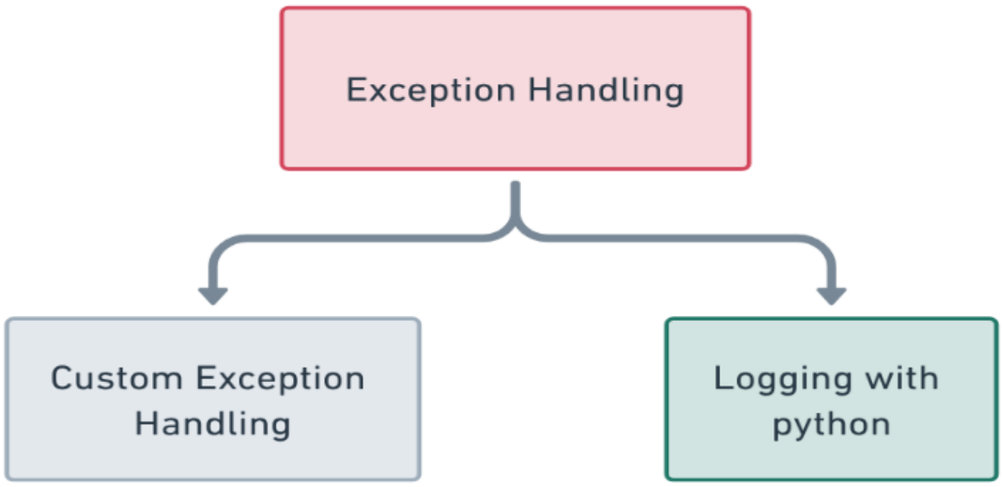
PW SKILLS

Lesson Plan

Custom Exception Handling

Topics to be covered :

- Custom Exception Handling
- Logging with python



Custom Exception Handling

You can define your own exception classes in Python to create custom exceptions. Custom exceptions might be handy when you want to raise certain issues in your code that are not handled by the built-in exception classes. Here's an example of how to make a custom exception:

Syntax of Custom Exceptions:

```
class CustomException(Exception):
 pass
try:
 #code
```

except CustomException:

Ex:

```
class NumNotInRangeError(Exception):

    def __init__(self, num):
      message="Number not present in (5, 20)range values"
      self.num = num
      self.message = message
      super().__init__(self.message)


num = int(input("Enter number: "))
if not 5 < num < 20:
    raise NumNotInRangeError(num)
```

```
Enter number: 25
---------------------------------------------------------------------------
NumNotInRangeError                        Traceback (most recent call last)
<ipython-input-6-d4ecbd878473> in <cell line: 11>()
     10 num = int(input("Enter number: "))
     11 if not 5 < num < 20:
---> 12   raise NumNotInRangeError(num)

NumNotInRangeError: Number not present in (5, 20)range values
```

Logging with python :

Logging is a crucial aspect of the coding process since it allows you to keep track of everything that happens when you run the code. We can construct custom messages/events to signify different stages of the code by logging. Logging facilitates code monitoring, troubleshooting, and debugging.

Using a "print" function to perform logging is the simplest method. However, using the "print" function has a number of drawbacks, and Python's "logger" method has a number of useful features. While the code is still in the development stage, the "Print" command might allow you to debug the code in the event of an error, but once the code has been sent to production, utilizing logger is the recommended approach.

To record various occurrences, the logger offers several severity levels (warning, errors, critical, info, etc.). According to the official Python document, the various severity levels are shown below along with a brief description:

- **DEBUG** - Detailed information, typically of interest only when diagnosing problems.
- **INFO** - Confirmation that things are working as expected.
- **WARNING** - .An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low'). The software is still working as expected.
- **ERROR** - Due to a more serious problem, the software has not been able to perform some function.
- **CRITICAL** - A serious error, indicating that the program itself may be unable to continue running.

Ex:

```
# We need to import logging python module
import logging
```

```
# let's create a log file with name "logger" and set the severity level to "info"
logging.basicConfig(filename='Logger1.log',level=logging.INFO)

```

```
logging.info("Info message being logged!!")
logging.warning("Warning message being logged!!")
logging.error("Error!!")
```

Let's open the log file created and see if the messages are logged.

PythonLogger.ipynb ● Logger.log ×

```
INFO:root:Info message being logged!!
WARNING:root:Warning message being logged!!
ERROR:root:Error!!

```

Real-time Analogy Code:

Example-1

```
#Analogy: ATM Withdrawal
class ATMWithdrawalException(Exception):
    def __init__(self, balance, amount):
        self.balance = balance
        self.amount = amount
        super().__init__(f"Insufficient funds. Cannot withdraw ${amount}.
Balance available: ${balance}")

def withdraw_money(balance, amount):
    try:
        if amount > balance:
            raise ATMWithdrawalException(balance, amount)
        else:
            print(f"Withdrawal successful! Remaining balance: ${balance -
amount}")
    except ATMWithdrawalException as e:
        print(e)

# Example usage
balance = 500
withdraw_amount = 700
withdraw_money(balance, withdraw_amount)

Output:
Insufficient funds. Cannot withdraw $700. Balance available: $500
```

Example-2

```
#Analogy: StudentGradeException
class StudentGradeException(Exception):
    def __init__(self, grade):
        self.grade = grade
        super().__init__(f"Invalid grade. Grade '{grade}' is out of range
(0-100)")

def evaluate_grade(grade):
    try:
        if not 0 ⩽ grade ⩽ 100:
            raise StudentGradeException(grade)
        else:
            print("Grade evaluation successful!")
    except StudentGradeException as e:
        print(e)

# Example usage
student_grade = 120
evaluate_grade(student_grade)

Output:
Invalid grade. Grade '120' is out of range (0-100)
```

Example-3

```
#Analogy: EmailSendingException
class EmailSendingException(Exception):
    def __init__(self, email, message):
        self.email = email
        self.message = message
        super().__init__(f"Failed to send email to '{email}': {message}")

def send_email(email, message):
    try:
        # Simulating a failed email sending scenario
        raise EmailSendingException(email, message)
    except EmailSendingException as e:
        print(e)

# Example usage
recipient_email = "example@email.com"
email_message = "This is a test message."
send_email(recipient_email, email_message)

Output:
Failed to send email to 'example@email.com': This is a test message.
```

Example-4

```
#Analogy: PaymentProcessingException
class PaymentProcessingException(Exception):
    def __init__(self, payment_method, amount):
        self.payment_method = payment_method
        self.amount = amount
        super().__init__(f"Payment processing failed using '{payment_method}'
for amount ${amount}")

def process_payment(payment_method, amount):
    try:
        # Simulating a failed payment scenario
        raise PaymentProcessingException(payment_method, amount)
    except PaymentProcessingException as e:
        print(e)

# Example usage
payment_type = "Credit Card"
payment_amount = 1000
process_payment(payment_type, payment_amount)

Output:
Payment processing failed using 'Credit Card' for amount $1000
```

Example-5

```
#Analogy: NetworkConnectionException
class NetworkConnectionException(Exception):
    def __init__(self, url):
        self.url = url
        super().__init__(f"Failed to establish a connection to '{url}'")

def establish_connection(url):
    try:
        # Simulating a scenario where the network connection fails
        raise NetworkConnectionException(url)
    except NetworkConnectionException as e:
        print(e)

# Example usage
website_url = "https://pwskills.com"
establish_connection(website_url)
Output:
Failed to establish a connection to 'https://pwskills.com'
```