Integrita dat, bezpečnost, logování, kontrola vstupů, zpracování chyb

Integrita dat
Koncept, který zajišťuje to, aby data zůstala přesná a nedocházelo k neoprávněným změnám či poškození. Například aby datový typ int měl opravdu hodnotu int, nebo varchar měl délku 32 bitů atd…..

Základní principy
Typová integrita = data musí odpovídat správným definovaným datovým typům
Doménová integrita = hodnoty musí dodržovat stanovený rozsah pro svojí velikost
Referenční integrita = odkazy mezi daty musí být jednoznačné
Entitní integrita = každý záznam musí být identifikovatelný

Jak zajistit integritu
Validace vstupů = zabránění neplatných nebo nebezpečných dat, které by mohly narušit funkčnost
Transakce = zajišťuje, že série operací proběhne kompletně celá, nebo vůbec
Zálohování = zajištění integrity dat v případě havárie
Normalizace = odstranění redudantních dat

Kontrola vstupů
Proces, který ověřuje a validuje data před jejich zpracováváním. Je to nutné pro správný chod aplikace.

Pomáhá chránit náš program před:
Pádům programu = odhalí zda uživatel zadal správný datový typ, pokud ne bez kontroly by to mohlo způsobit pád celého programu
Útoky = SQL injection
Integrita dat = Do systému pouze putují platná a správná data

Typy kontrol vstupů
Typová = správný datový typ
Rozsahová = hodnoty se pohybují v daném rozmezí
Formátová = data odpovídají svému účelu (email=email, nikoliv telefonní číslo atd…

Kontrolování v praxi
Podmínky if a else = nejzákladnější, pokud se něco splní = provede se akce

```
print("Enter number")
input_number = int(input())

if input_number > 0:
   print("number is positive")
else:
   print("number is negative")
```

Výjimky try a except = zachycují výjimky a provadí akci v případě výjimky

```
while True:
   try:
       user_number = int(input())
   except:
       print("Invalid input")
```

Regex = definují podmínky na základě vzorů, nezapomenout importovat knihovnu re!

```
import re

email = "email@gmail.com"

if re.match(r"^[\w\.-]+@[\w\.-]+\.\w+$", email):
   print("valid")
else:
   print("invalid")
```

Bezpečnost
Zaměřuje se na ochranu dat a systému před neoprávněným systémem, zneužitím nebo útokem.

Hlavní oblasti zabezpečení
Kontrola datových typů = brání neplatným datovým typům a formátům
Šifrování dat = chrání citlivá data před neoprávněným přístupem (hesla atd..), Symterické(AES), Asymetrické(RSA) a Hašovací funkce(SHA)
Ověřování identity = práva, zajištuje přistup pouze oprávněným userum, ověření pomocí hesla, práva ….

Typy útoků
SQL injection = útok, využívající chyby v aplikaci k manipulaci s databází, útočník vkládá SQL příkazy do vstupních polí, ochrana pomocí Escapování výstupů, HTTPOnlu nebo Secure cookies

```
# NEBEZPEČNÉ: Útočník může zadat admin' OR '1'='1
username = input("Zadej uživatelské jméno: ")
query = f"SELECT * FROM users WHERE username = '{username}'"

# BEZPEČNÉ: Parametrizovaný dotaz
username = input("Zadej uživatelské jméno: ")
query = "SELECT * FROM users WHERE username = %s"
cursor.execute(query, (username,))  # Parametry jako tuple
```

XSS (cross-site scripting) = vložení škodlivého scriptu nejčastěji do webobvých stránek, krádeže cookies, phising, přesměrování, ochrana pomocí Escapování výrazů….
Broken authentication = slabiny v implementaci loginu, zahranuje slabá hesla, převídatelné tokeny…, prevence - silná hesla, dvoufázovka, omezení počtu pokusů

samozřejmě mnohem víc….

Logování
Proces zaznamenávající údalosti, aktivity a stav během běhu aplikace. Reprezentuje něco jako černou skřínku, která poskytuje zápis všeho co se ho událo. Logování se používá kvůli jednoduché detekci chyb, analýzu výkonu, monitorování chování….Knihovna logging v pythonu to docela pěkně řeší

Úrovně logování
Debug = Velmi detailní informace, užitečné především při vývoji a ladění
Info = Běžné informace o normálním chodu aplikace
Warning = Upozornění na potenciální problémy, které ale nebrání normální funkčnosti
Error = Chyby, které znemožnily provedení konkrétní operace
Critical = Kritické chyby, které mohou vést k pádu celé aplikace

```
2025-04-03 16:41:07,749 - INFO - Collecting data for Python on 2020-03-04
2025-04-03 16:41:09,683 - INFO - Collecting data for Python on 2020-03-11
2025-04-03 16:41:13,115 - INFO - Collecting data for Python on 2020-03-18
2025-04-03 16:41:51,140 - INFO - Collecting data for Python on 2020-03-25
2025-04-03 16:41:51,140 - INFO - Progress: 10/3300 (0.3%)
2025-04-03 16:41:51,141 - INFO - Est. remaining: 15:16:16
2025-04-03 16:41:51,727 - INFO - Collecting data for Python on 2020-04-01
2025-04-03 16:41:54,468 - INFO - Collecting data for Python on 2020-04-08
2025-04-03 16:42:32,047 - INFO - Collecting data for Python on 2020-04-15
2025-04-03 16:42:33,457 - INFO - Collecting data for Python on 2020-04-22
2025-04-03 16:42:38,041 - INFO - Collecting data for Python on 2020-04-29
2025-04-03 16:43:12,801 - WARNING - Secondary rate limit hit. Waiting 132 seconds...
2025-04-03 16:43:16,618 - INFO - Collecting data for Python on 2020-05-06
2025-04-03 16:43:20,141 - INFO - Collecting data for Python on 2020-05-13
2025-04-03 16:45:32,264 - INFO - Progress: 20/3300 (0.6%)
2025-04-03 16:45:32,264 - INFO - Est. remaining: 17:41:09
2025-04-03 16:45:32,279 - INFO - Saved 20 records to github_language_data.csv
2025-04-03 16:45:32,493 - INFO - API Rate Limit: 29 remaining, resets at 2025-04-03 16:46:32
2025-04-03 16:45:32,494 - INFO - Collecting data for Python on 2020-05-20
2025-04-03 16:45:32,494 - INFO - Collecting data for Python on 2020-05-27
2025-04-03 16:45:32,494 - INFO - Collecting data for Python on 2020-06-03
2025-04-03 16:46:12,439 - INFO - Collecting data for Python on 2020-06-10
```

Třeba u Omegy, když jsme scrapovali data, tak jsem používal ten modul logging a takhle to nějak vypadá.

```python
import logging

logging.basicConfig(
    level=logging.INFO,                 # Úroveň logování (DEBUG, INFO, WARNING, ERROR, CRITICAL)
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',  # Formát zprávy
    datefmt='%Y-%m-%d %H:%M:%S',        # Formát data a času
    filename='aplikace.log',            # Soubor pro zápis logů
    filemode='a'                        # Režim zápisu (a = append/přidávání)
)
```

Zpracovávání chyb

Try catch

```
try:
   # nebezpecnej kod
except:
   #co ma stat kdyz dojde k chybe
try:
   # Potenciálně problematický kód
   x = 10 / 0  # Vyvolá ZeroDivisionError
except ZeroDivisionError:
   # Kód pro zpracování konkrétní výjimky
   print("Nelze dělit nulou!")
```

Except blok = zpracovává konkrétní výjimky, různé bloky pro různé výjimky

Else blok = vykonává se pokud, žádná výjimka nenastala, nakonci za všema except blokama,

```
try:
   výsledek = 10 / 2
except ZeroDivisionError:
   print("Dělení nulou!")
else:
   print(f"Výsledek: {výsledek}")  # Spustí se jen když nenastane výjimka
```

finally blok = vykoná se vždy bez ohledu na to jestli výjimka nastala neebo ne

```
try:
   soubor = open("data.txt", "r")
   obsah = soubor.read()
except FileNotFoundError:
   print("Soubor nenalezen!")
finally:
   # Tento kód se provede vždy - s výjimkou i bez ní
   print("Operace dokončena")
   # Zde by bylo zavření souboru, pokud existuje
```

Je možný i zachytit více výjimek

```
try:
   # Rizikový kód
   hodnota = int(input("Zadej číslo: "))
   výsledek = 100 / hodnota
except ValueError:
   print("To není platné číslo!")
except ZeroDivisionError:
   print("Nelze dělit nulou!")
except:  # Zachytí všechny ostatní výjimky
   print("Nastala jiná chyba")
```

A i možný udělat svojí vlastní výjimku