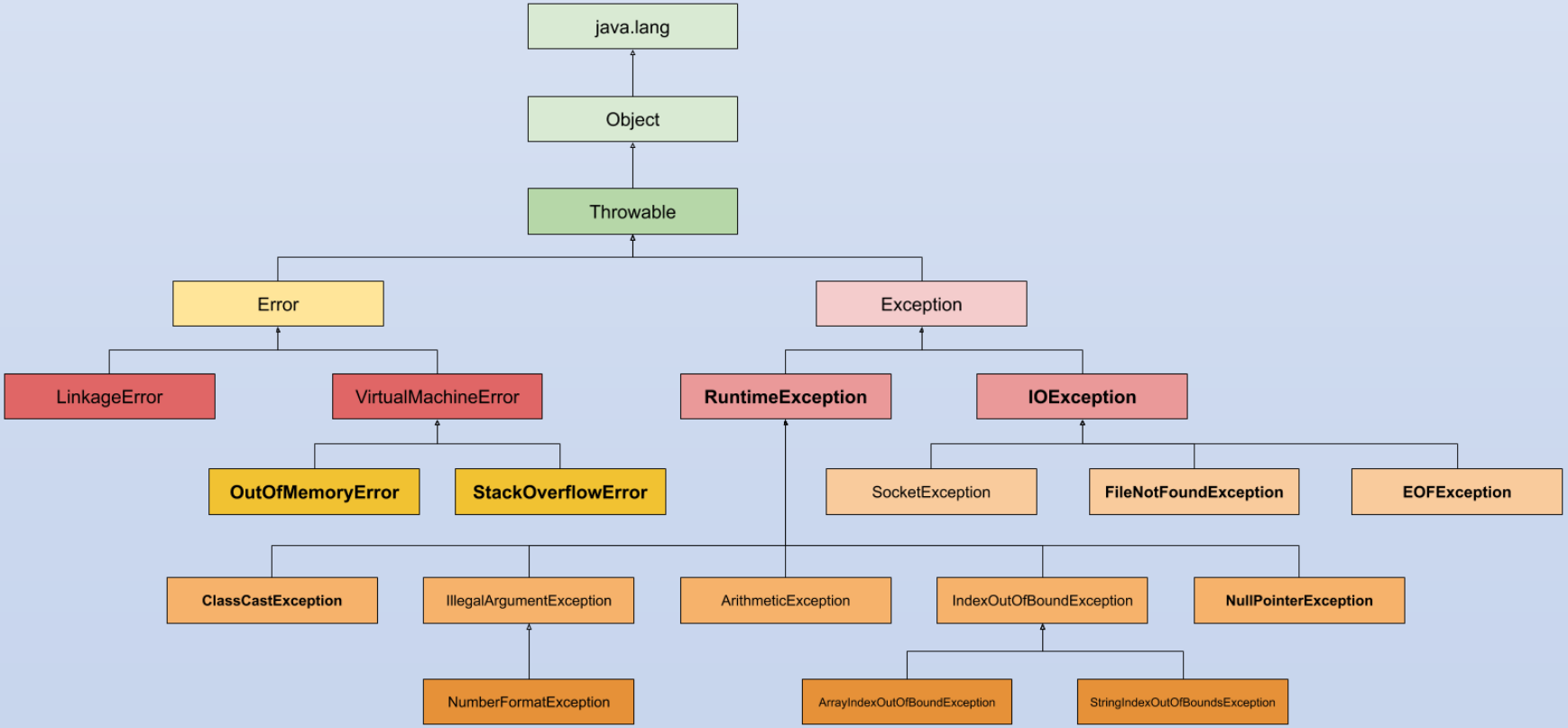
# ОСНОВИ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ, НЕЙРОННИХ МЕРЕЖ ТА ГЛИБОКОГО НАВЧАННЯ

## Частина 6. ВИСОКОРІВНЕВА МОВА ПРОГРАМУВАННЯ PYTHON

## Лекція 6.8. Убудовані функції та вбудовані класи виняткових ситуацій.

# ВИНЯТКОВІ СИТУАЦІЇ

Приклад виняткових ситуацій з мови програмування Java.



# ВИНЯТКОВІ СИТУАЦІЇ

**Виняток (виняткова ситуація, exception) – спеціальний даних в Python.**
Винятки повідомляють про помилки програмі.

Програмна обробка необхідна щоб програма не завершувалося аварійно кожен раз, коли виникає виняток. Для цього блок коду, в якому можлива поява виняткової ситуації необхідно помістити всередину спеціальної синтаксичної конструкції

**try … except …**
**try … except … else ….**
**try … except … else … finally …**

# ВИНЯТКОВІ СИТУАЦІЇ

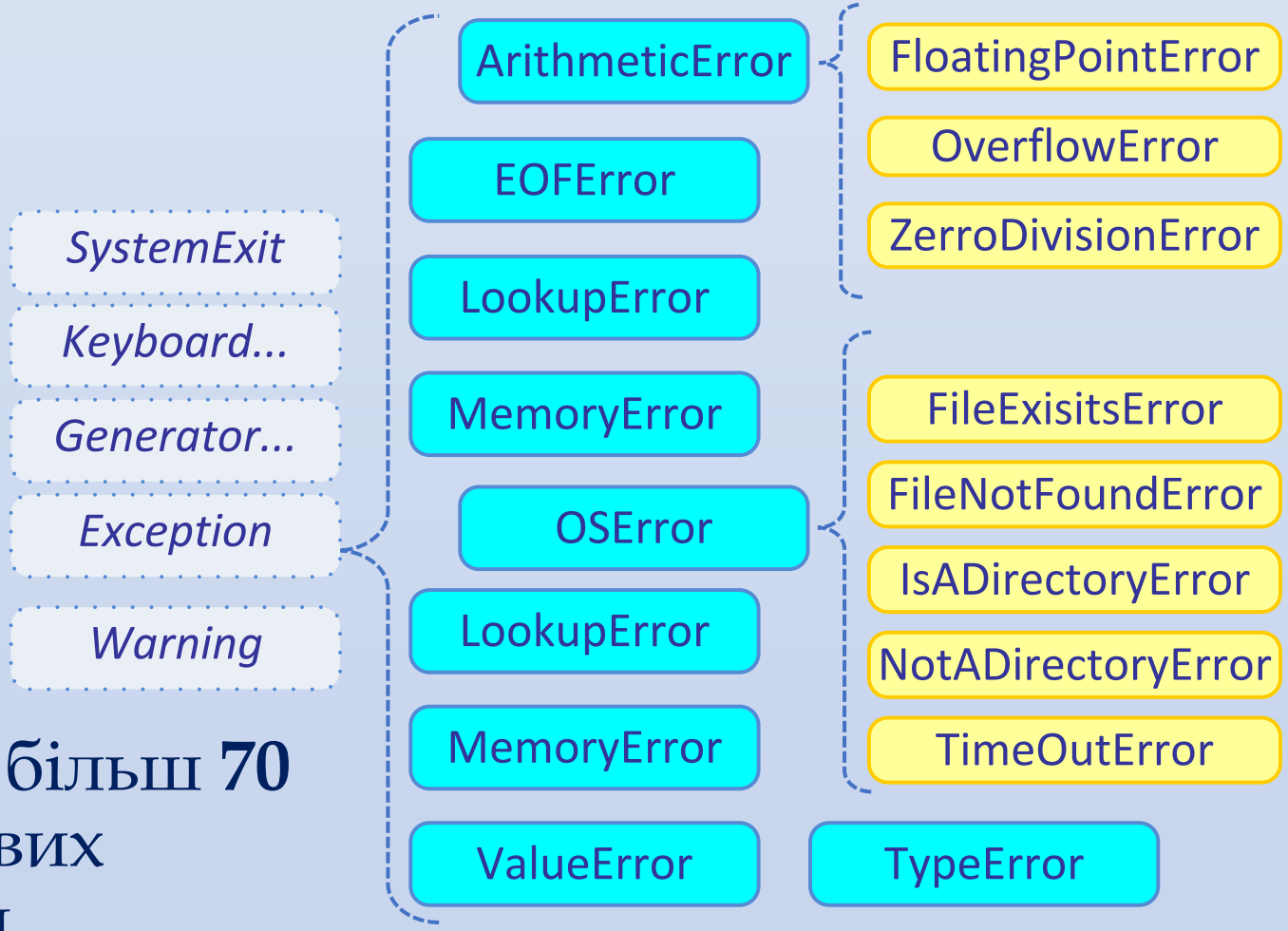
```
try:
   block-1
except Exception1:
   handler-1
except  Exception2:
   handler-2
else:
   else-block
finally:
   final-block
```

Виконується **block-1**. Якщо створюється виняткова ситуація перевіряється `except` блок.
Якщо виняток - **Exception1** – виконується блок **handler-1**.
Якщо виняток - **Exception2** – виконується блок **handler-2**
і так далі.

Якщо виняток не створюється, виконується **else-block** (якщо присутній). В будь якому випадку один раз виконується **final-block** (якщо присутній).

# ВИНЯТКОВІ СИТУАЦІЇ

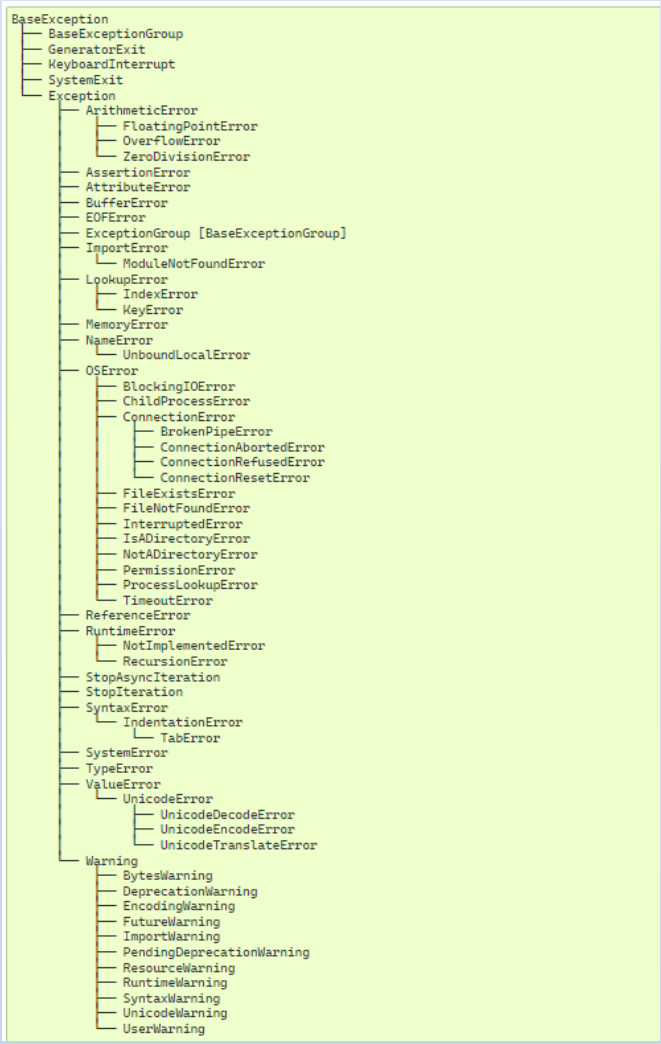
**Exception1, Exception2** – ім'я винятковоі ситуації (спеціальний тип даних **Python**)




Загалом більш **70** винятковых ситуацій

# ВИНЯТКОВІ СИТУАЦІЇ

## Ієрархія класів для вбудованих винятків




Посилання https://docs.python.org/3/library/exceptions.html

# ГЕНЕРУВАННЯ ВИНЯТКІВ

**Інструкція `raise`** дозволяє примусово генерувати виняткової ситуації

**`raise` Exception** — **Exception –** вказує на тип виняткової ситуації.

**Python** дозволяє створювати власні виняткові ситуації Для цього потрібно створити клас, який є підкласом класу **Exception**().
Важливо: в класі **Exception** визначено метод **__str__**, що дозволяє виводити значення його атрибутів.

# PEP

**Python Enhancement Proposal** (PEP) -
є механізмом документування проектних рішень, які пройшли в Python, і для пропозиції нових можливостей мови.

## Meta-PEPs (PEPs about PEPs or Processes)

| | PEP | PEP Title | PEP Author(s) |
|---|---|---|---|
| P | 1 | PEP Purpose and Guidelines | Warsaw, Hylton, Goodger, Coghlan |
| P | 4 | Deprecation of Standard Modules | Cannon, von Löwis |

## Other Informational PEPs

| | PEP | PEP Title | PEP Author(s) |
|---|---|---|---|
| I | 13 | Python Language Governance | and community |
| I | 20 | The Zen of Python | Peters |

**> 8000**

| | | | |
|---|---|---|---|
| I | 8101 | 2020 Term steering council election | Jodlowska, III |

Посилання https://www.python.org/dev/peps/

# PEP 8 – Стиль коду Python

## PEP 8 -- Style Guide for Python Code

Ключова ідея: код **читається** набагато більше разів, ніж пишеться. Рекомендації про стиль написання коду спрямовані на те, щоб поліпшити читабельність коду і зробити його узгодженим між великим числом проектів. В ідеалі, весь код повинен бути написан в єдиному стилі, і будь-хто може легко його прочитати.

**«Читабельність має значення».**

Посилання https://www.python.org/dev/peps/pep-0008/

# PEP 8 – Стиль коду Python

**Відступи:**
4 **пробіла** на один рівень відступу. Ніколи не змішувати символи табуляції і пробіли. Рекомендовано тільки **пробіли.**

**Максимальна довжина рядка.**
Рекомендовано обмежити **79** символами. Переважний спосіб перенесення довгих рядків - **зворотний слеш.**
Рекомендовано ставити перенесення рядка після бінарного оператора, але не перед ним (математично перед бінарним оператором).

# PEP 8 – Стиль коду Python

**Пусті рядки:**

Відокремлювати функції (верхнього рівня, що не функції всередині функцій) і визначення класів **двома** порожніми рядками.

Визначення методів всередині класу відокремлювати **одним** порожнім рядком.

Допускається використання порожніх рядків в коді функцій, щоб відокремити один від одного логічні частини.

# PEP 8 – Коментарі

**Блок коментарів:**

Блок коментарів зазвичай пояснює код (весь, або тільки деяку частину), що йде після блоку, і повинен мати той же відступ, що і сам код. Кожен рядок такого блоку повинен починатися з символу # і одного пробілу після нього.

Абзаци всередині блоку коментарів краще відокремлювати рядком, що складається з одного символу #.

# PEP 8 – Коментарі

**Коментарі в рядку коду:**
Коментарі в рядку з кодом не потрібні і лише відволікають від читання, якщо вони пояснюють очевидне. Намагайтеся рідше використовувати подібні коментарі.

Він повинен відділятися хоча б двома пробілами від інструкції. Він повинні починатися з символу # і пробілу.

# PEP 8 – Документування

**Рядки документації:**
Повна угода про написання документації (docstrings) викладена в PEP 257.

Пишіть документацію для всіх модулів, функцій, класів, методів, які оголошені як public. Коментар потрібно писати після рядка з **`def name()`**:

```
""" вертає список імен
Додатковий аргумент – рік народження

"""
```

# PEP 8 – Стилі імен Python

**Визначення стилю:**

- `b` - одиночна маленька буква;
- `B` - одиночна заголовна буква;
- `lowercase` - слово в нижньому регістрі;
- `lower_case_with_underscores` - слова з маленьких букв з підкресленнями;
- `UPPERCASE` - великі літери;
- `UPPERCASE_WITH_UNDERSCORES` - слова з великої літери з підкресленнями;
- `CapitalizedWords` - слова з великими літерами, або `CapWords`, або `CamelCase 5`. Іноді називається `StudlyCaps`. HTTPServerError краще, ніж HttpServerError.

# PEP 8 – Стилі імен Python

**Визначення стилю:**

- **`CapitalizedWords`** - слова з великими літерами, або **`CapWords`**, або **`CamelCase 5`**. Іноді називається **`StudlyCaps`**. HTTPServerError краще, ніж HttpServerError.
- **`mixedCase`** - відрізняється від **`CapitalizedWords`** тим, що перше слово починається з маленької літери; **`Capitalized_Words_With_Underscores`** - слова з великими літерами і підкресленнями
- **`__double_leading_and_trailing_underscore__`** - подвійне підкреслення на початку і в кінці імені. Наприклад, __init__, __import__ або __file__. Використовувати тільки так, як написано в документації.

# PEP 8 – Стилі імен Python

**Загальні рекомендації:**

Ніколи не використовуйте символи `l` (малелькая латинська буква «ель»), `O` (заголовна латинська буква «o») або `I` (заголовна латинська буква «ай») як однобуквені ідентифікатори.

**Модулі та пакети – `b`**
**Класи – `CapWords`**
**Винятки – `CapWords (+Error)`**
**Глобальні змінні , функції –**
`lower_case_with_underscores`
**Константи – `UPPERCASE`,**
`UPPERCASE_WITH_UNDERSCORES`

# PEP 8 – Стилі імен Python

**Загальні рекомендації:**

**Аргументи функцій (методів) -** завжди використовуйте `self` як перший аргумент методу екземпляру об'єкта, - завжди використовуйте `cls` як перший аргумент методу класу.

**Завжди дотримуйтеся прийнятого Python однакового стилю (на базі PEP). Особливо при колективній роботі над проектом.**

# Рекомендована ЛІТЕРАТУРА


- **Програмування числових методів мовою Python:** підруч. / А. В. Анісімов, А. Ю. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий ; за ред. А. В. Анісімова. – К. : Видавничо-поліграфічний центр "Київський університет", 2014. – 640 с.
- **Програмування числових методів мовою Python:** навч. посіб. / А. Ю. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий, Є. В. Глушко ; за ред. А. В. Анісімова. – К. : Видавничо-поліграфічний центр "Київський університет", 2013. – 463 с.
- **Основи програмування Python:** Підручник для студ. спеціальності 122 «Компютерні науки» / А.В.Яковенко; КПІ.- Київ: КПІ, 2018 . – 195 с.
- **Бейдер Д.** Чистый Python. Тонкости программирования для профи.-СПб.: Питер. 2018.-288 с.: ил.

# Посилання

- https://pep8.ru/doc/pep8/
- https://github.com/python/peps/blob/master/pep-0008.txt

# Контрольні запитання

- Визначте поняття виняткової ситуації при виконанні операторів, надайте приклади виняткових ситуацій.
- Наведіть оператори обробки виняткових ситуацій, визначте порядок обробки винятку операторами try … except
- Надайте мету та визначте порядок використання оператора генерації виняткових ситуацій raise, надайте приклади.
- Надайте  основні положення стилю коду Python (PEP-8)

# The END
# Частина 6. Лекція 6.8.