Вступ до програмування на Python для фізматівців

Ця презентація розроблена для студентів факультету математики, фізики та комп'ютерних наук ВДПУ ім. М. Коцюбинського. Ми розглянемо основи програмування на Python, зосередившись на його практичному застосуванні у розв'язанні типових задач з вашої спеціальності.

Програма курсу

Базові конструкції Python

Ознайомлення з синтаксисом, змінними, типами даних та операторами.

Робота з даними

Вивчення структур даних, таких як списки, кортежі, словники та множини.

Алгоритми та їх реалізація

Розробка простих алгоритмів для розв'язання математичних та фізичних задач.

Практичні застосування

Приклади застосування Python у наукових обчисленнях та візуалізації даних.

Чому саме Python?

- **Простота та читабельність:** Python має дуже чистий і інтуїтивно зрозумілий синтаксис, що полегшує його вивчення для початківців.
- **Універсальність:** Використовується в наукових обчисленнях, веб-розробці, штучному інтелекті, аналізі даних та багатьох інших сферах.
- **Велика екосистема бібліотек:** NumPy, SciPy, Matplotlib, Pandas – це лише кілька з тисяч доступних бібліотек для математики, статистики та візуалізації.

Базові конструкції Python

Змінні та типи даних

У Python не потрібно оголошувати тип змінної, він визначається автоматично. Основні типи включають цілі числа (int), числа з плаваючою комою (float), рядки (str) та булеві значення (bool).

```
# Приклад змінних
x = 10        # int
pi = 3.14159   # float
name = "Alice" # str
is_student = True # bool
```

Оператори

Python підтримує арифметичні, порівняльні, логічні та оператори присвоєння. Розуміння їх використання є ключовим для побудови виразів.

```
# Арифметичні операції
result = 15 + 7
product = 5 * 3.5

# Порівняння
is_greater = 10 > 5

# Логічні операції
if is_greater and is_student:
 print("Успіх!")
```

Керуючі структури

Керуючі структури дозволяють контролювати потік виконання програми, роблячи її інтерактивною та здатною приймати рішення.

1

Умовні оператори (if/else)

Використовуються для виконання певного блоку коду за заданої умови. Це основа для розгалужень у програмах.

```
if temperature > 25:
    print("Жарко")
else:
    print("Нормально")
```

2

Цикли (for/while)

Цикли дозволяють повторювати блок коду певну кількість разів або до виконання певної умови. Незамінні для обробки колекцій даних.

```
for i in range(5):
    print(i) # 0, 1, 2, 3, 4
```

3

Функції

Функції дозволяють організувати код у логічні блоки, підвищуючи його повторне використання та читабельність. Допомагають розбивати складні задачі на менші частини.

```
def calculate_area(radius):
 return 3.14 * radius**2
```

Робота з основними структурами даних

Python пропонує гнучкі вбудовані структури даних, які є фундаментом для ефективної обробки інформації.

Списки (List)

Упорядковані, змінювані колекції елементів. Можуть містити елементи різних типів. Ідеальні для динамічних наборів даних.

```
my_list = [1, "hello", 3.14]
my_list.append(True)
```

Кортежі (Tuple)

Упорядковані, **незмінювані** колекції елементів. Використовуються для зберігання даних, які не повинні змінюватися після створення.

```
my_tuple = (10, 20, 30)
```

Словники (Dictionary)

Невпорядковані колекції пар "ключ-значення". Ключі повинні бути унікальними та незмінними. Дуже ефективні для швидкого пошуку даних.

```
my_dict = {"name": "Bob",
"age": 30}
```

Прості алгоритми для фізико-математичних задач

Python чудово підходить для реалізації алгоритмів, що розв'язують типові задачі з математики та фізики.

Обчислення коренів квадратного рівняння

```
import math

def solve_quadratic(a, b, c):
    delta = b**2 - 4*a*c
    if delta >= 0:
        x1 = (-b + math.sqrt(delta)) / (2*a)
        x2 = (-b - math.sqrt(delta)) / (2*a)
        return x1, x2
    else:
        return "Немає дійсних коренів"

print(solve_quadratic(1, -3, 2))
```

Симуляція вільного падіння

Моделювання простих фізичних явищ, наприклад, вільного падіння тіла, може бути реалізовано за допомогою циклів та основних рівнянь кінематики.

```
def free_fall(initial_height, time_step, total_time):
 g = 9.81 # прискорення вільного падіння
 height = initial_height
 velocity = 0
 t = 0
 while t <= total_time and height >= 0:
 print(f"Час: {t:.2f} с, Висота: {height:.2f} м")
 height -= velocity * time_step + 0.5 * g * time_step**2
 velocity += g * time_step
 t += time_step
 print(f"Час: {t:.2f} с, Висота: {height:.2f} м (земля)")

free_fall(100, 0.5, 5)
```

Аналіз даних та візуалізація

Python, завдяки бібліотекам NumPy та Matplotlib, стає потужним інструментом для аналізу та представлення даних.

NumPy для числових обчислень

NumPy є основою для наукових обчислень у Python, надаючи високоефективні масиви та функції для роботи з ними. Він дозволяє виконувати складні матричні операції швидко.

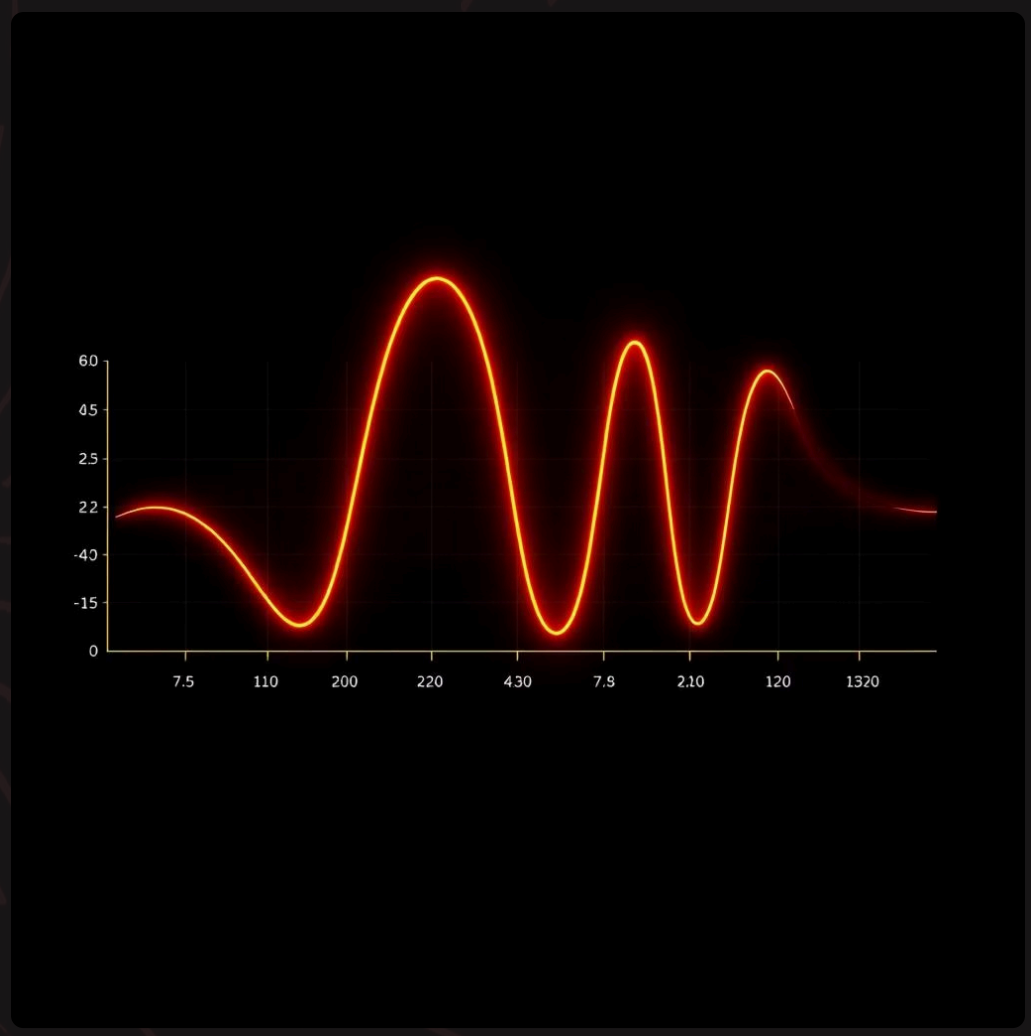
```
import numpy as np

# Створення масиву
a = np.array([1, 2, 3, 4])
b = np.array([5, 6, 7, 8])

# Поелементне додавання
c = a + b
print(c)
```

Matplotlib для візуалізації

Matplotlib – це бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python.




```
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y)
plt.title("Графік синуса")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

Додаткові можливості та ресурси

Для поглиблення знань та розширення можливостей Python існує безліч ресурсів.

Інтегровані середовища розробки (IDE)

Використання PyCharm, VS Code або Jupyter Notebook значно спрощує процес написання та відлагодження коду.

Онлайн-курси та документація

Ресурси, такі як Coursera, edX, або офіційна документація Python, пропонують структуровані знання та практичні завдання.

Спільнота та форуми

Активна спільнота Python готова допомогти з будь-якими питаннями. Stack Overflow та тематичні форуми – ваші найкращі друзі.

Підсумки та наступні кроки

Ключові висновки

Python є потужним, простим у вивченні та універсальним інструментом, ідеально придатним для розв'язання наукових та інженерних задач.

Досліджуйте бібліотеки

Поглиблюйте свої знання, вивчаючи такі бібліотеки, як SciPy для наукових обчислень, Pandas для аналізу даних та Scikit-learn для машинного навчання.

Практикуйтеся щодня

Найкращий спосіб вивчити програмування – це постійна практика. Розв'язуйте невеликі задачі, експериментуйте з кодом.

Залишайтеся на зв'язку

Не соромтеся звертатися до викладачів та колег. Спільне навчання та обмін досвідом – запорука успіху.