Лекція 1. Базові поняття програмування на мові Python

Лектор – доцент, кандидат фізико-математичних наук
Кириченко Віктор Вікторович

ПЛАН ЛЕКЦІЇ

1. Примітивні типи даних та змінні в Python. Неявна типізація. Перетворення типів.
2. Команди вводу виводу даних в консоль.
3. Реалізація алгоритмів з розгалуженням.
4. Оператори циклів. Ітератори і генератори.
5. Винятки та їх обробка.
6. Робота з бібліотеками

Примітивні типи даних та змінні в Python. Неявна типізація. Перетворення типів.

Основні типи даних у Python:

- `int` - цілі числа
- `float` - числа з плаваючою крапкою
- `str` - рядки
- `bool` - логічні значення
- `list`, `tuple`, `dict`, `set` - базові колекції.

Змінні створюються автоматично при присвоєнні значення, тип вказувати не потрібно (динамічна типізація).

```
name = 'User'
age = 12
```

Примітивні типи даних та змінні в Python. Неявна типізація. Перетворення типів.

Неявна типізація та перетворення типів

- Python сам визначає тип змінної під час виконання (неявна типізація).
- Неявне перетворення типів відбувається автоматично, наприклад, при додаванні int і float результат буде float.
- Явне перетворення (casting) здійснюється функціями: `int()`, `float()`, `str()`.

Команди вводу виводу даних в консоль

Вивід:

```
print("Текст або змінна")
```

Ввід:

```
name = input("Введіть ім'я: ")
```

Всі дані, отримані через input(), мають тип str.

```
age = int(input("Введіть вік: "))
```

Реалізація алгоритмів з розгалуженням

Для умов використовують оператори if, elif, else:

```
if x > 0:
    print("Додатнє")
elif x == 0:
    print("Нуль")
else:
    print("Від’ємне")
```

Оператори циклів. Ітератори і генератори.

Два основних типи циклів:

while - виконується, поки умова істинна:

```
i = 0
while i < 5:
    print(i)
    i += 1
```

for - перебирає елементи ітерованого об'єкта:

```
for i in range(5):
    print(i)
```

Оператори циклів. Ітератори і генератори.

- Ітератори — об'єкти, які підтримують протокол ітерації (__iter__ і __next__).

```
class MyRange:
    def   init  (self, start, end):
        self.current = start
        self.end = end

    def   iter  (self):
        return self  # ітератор має повертати себе

    def   next  (self):
        if self.current >= self.end:
            raise StopIteration  # сигнал завершення ітерації
        value = self.current
        self.current += 1
        return value


for number in MyRange(1, 5):
    print(number)
```

Оператори циклів. Ітератори і генератори.

- Генератори - спеціальні функції з ключовим словом yield, що повертають значення послідовно без збереження всієї послідовності в пам’яті.

```
def my range(start, end):
    current = start
    while current < end:
        yield current
        current += 1

# Використання
for number in my range(1, 5):
    print(number)
```

Винятки та їх обробка

Для обробки помилок використовують конструкцію try-except:

```
try:
    x = int(input("Введіть число: "))
except ValueError:
    print("Це не число!")
```

Робота з бібліотеками

Бібліотеки в Python - це готові набори функцій і класів, які значно спрощують розробку програм. Вони дозволяють не писати все з нуля, а використовувати вже створені рішення для роботи з математикою, вебом, даними, графікою тощо

```
pip install назва_бібліотеки
```

```
pip install назва_бібліотеки==номер_версії
```

Для встановлення кількох бібліотек одночасно або з файлу залежностей:

```
pip install -r requirements.txt
```

ДЯКУЮ ЗА УВАГУ