ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

@ZTUEDUUA

АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРИЗОВАНИХ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНИХ СИСТЕМ

Лекція 8

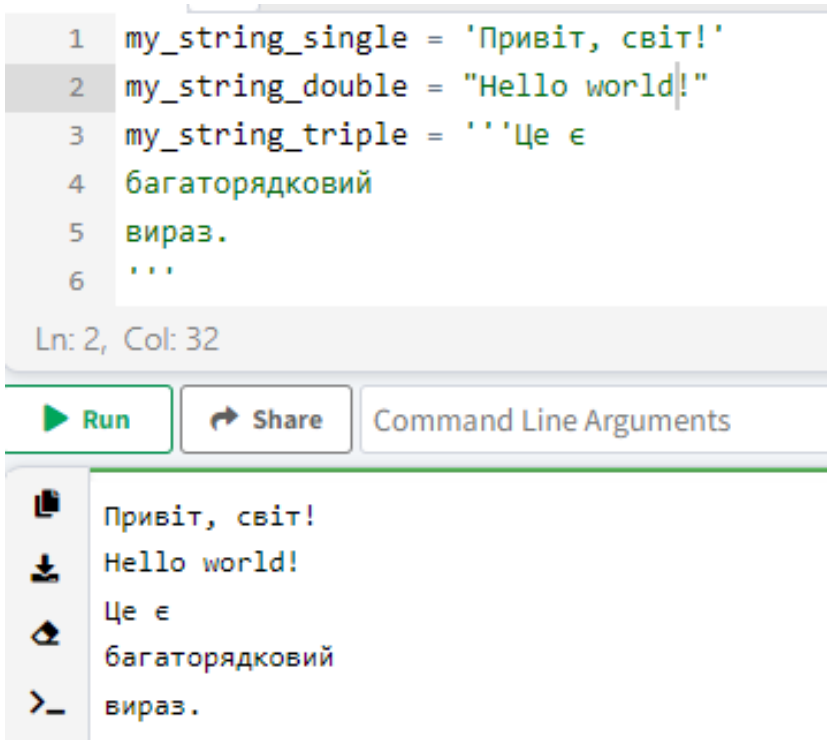
Тема:Рядки. Коди символів. Форматування рядків.

1. Рядки. Доступ до символів рядка в Python.
2. Операції з рядками в Python.
3. Методи для роботи з рядками в Python.
4. Форматування рядків в Python.
5. Escape-послідовності в Python.
6. Коди симовлів.

1. Рядки . Доступ до символів рядка в Python.

У Python рядки є послідовностями символів, які можуть бути визначені в лапках. Рядки можуть містити букви, цифри, символи пунктуації та спеціальні символи. Вони є незмінними об'єктами, що означає, що їх не можна змінювати після створення.

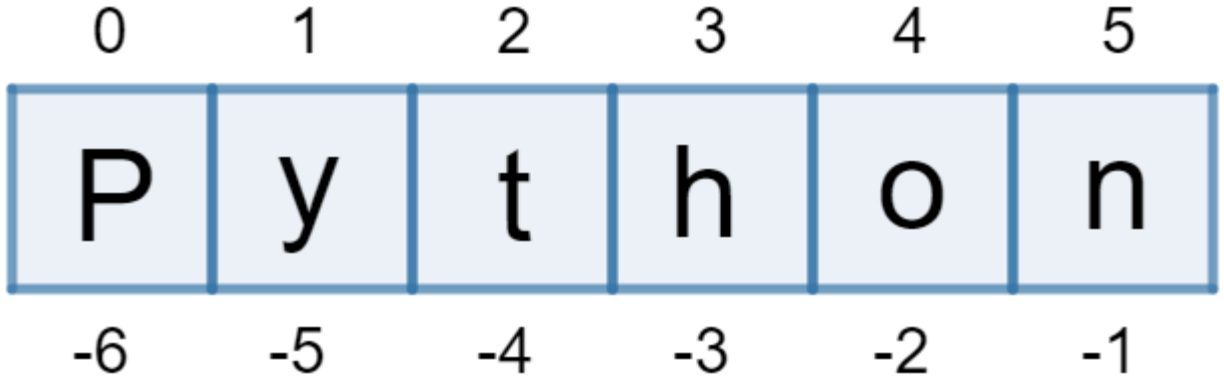
Створення рядків: Рядки можуть бути визначені за допомогою одинарних, подвійних або потрійних лапок. Подвійні та потрійні лапки дозволяють включати в рядки багаторядкові вирази.



Доступ до символів рядка в Python

Доступ до символів рядка в Python може здійснюватися трьома способами:

- Індексація;
- Від’ємна індексація;
- Зріз.



Індексація: Рядки можна індексувати, щоб отримати доступ до окремих символів за його індексом.

```
1  greet = 'hello'
2
3  # Доступ до символу під першим індексом
4  print(greet[1]) # виведе "e"
```

Ln: 4, Col: 29

Run | Share | Command Line Arguments

```
e
```

Звичайна індексація

```
1  greet = 'hello'
2
3  # Отримання доступу до 4-го символу з кінця
4  print(greet[-4]) # виведе "e"
```

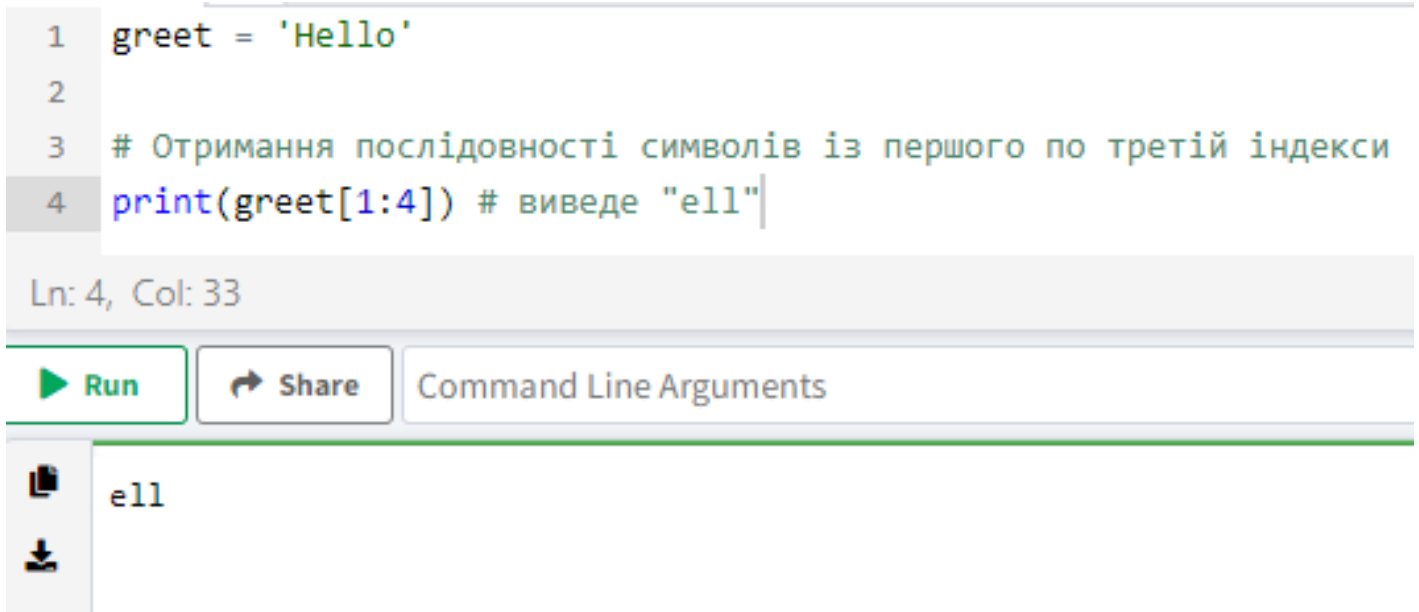
Ln: 3, Col: 44

Run | Share | Command Line Arguments

```
e
```

Від’ємна індексація

Зріз: відноситься до вилучення підрядка (підмножини) з ітерабельного об'єкта, такого як рядок, список, кортеж або будь-який інший ітерабельний тип даних. Зріз дозволяє отримувати підмножину елементів з ітерабельного об'єкта, викликається за допомогою оператора зрізу “ : ”



2. Операції з рядками в Python

- Порівняння двох рядків;
- Поєднання (конкатенація) двох рядків;
- Ітерація - це процес послідовного перебору елементів у колекції або послідовності;
- Довжина рядка;
- Перевірка на належність до рядка.

Можливість здійснення багатьох операцій з рядками робить цей тип даних одним з найчастіше використовуваних в Python.

Порівняння двох рядків.
Для порівняння двох рядків використовується оператор “ == ”. Якщо рядки однакові, оператор поверне True, в протилежному випадку — False.

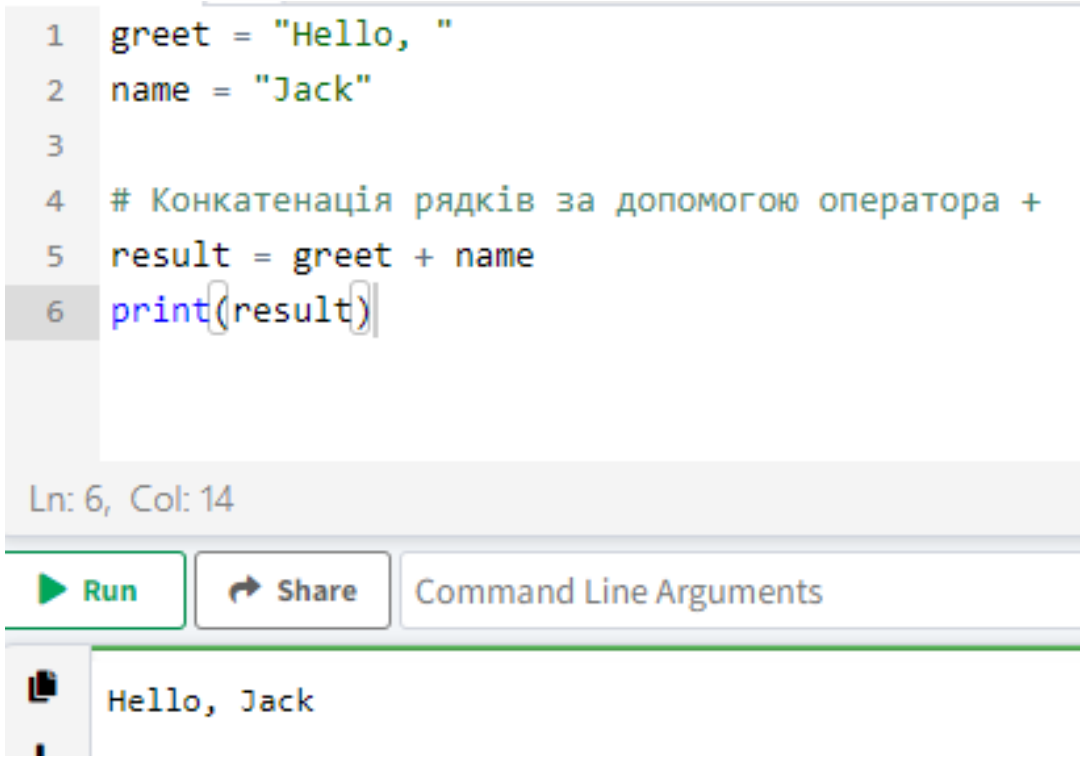
```
1  str1 = "Hello, world!"
2  str2 = "I love Zhytomyr Polytechnic."
3  str3 = "Hello, world!"
4
5  # Порівняння рядків str1 та str2
6  print(str1 == str2)
7
8  # Порівняння рядків str1 та str3
9  print(str1 == str3)
```

Ln: 2, Col: 36

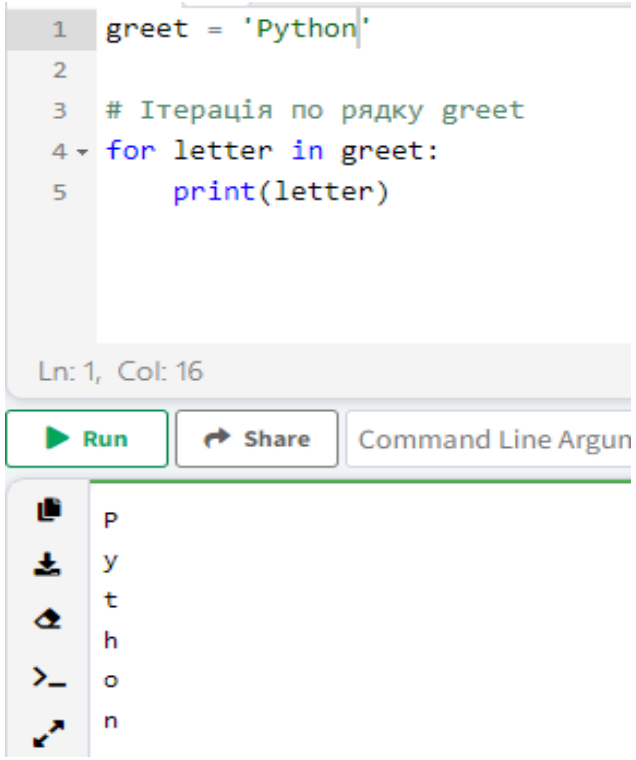
Run | Share | Command Line Arguments

```
False
True
```

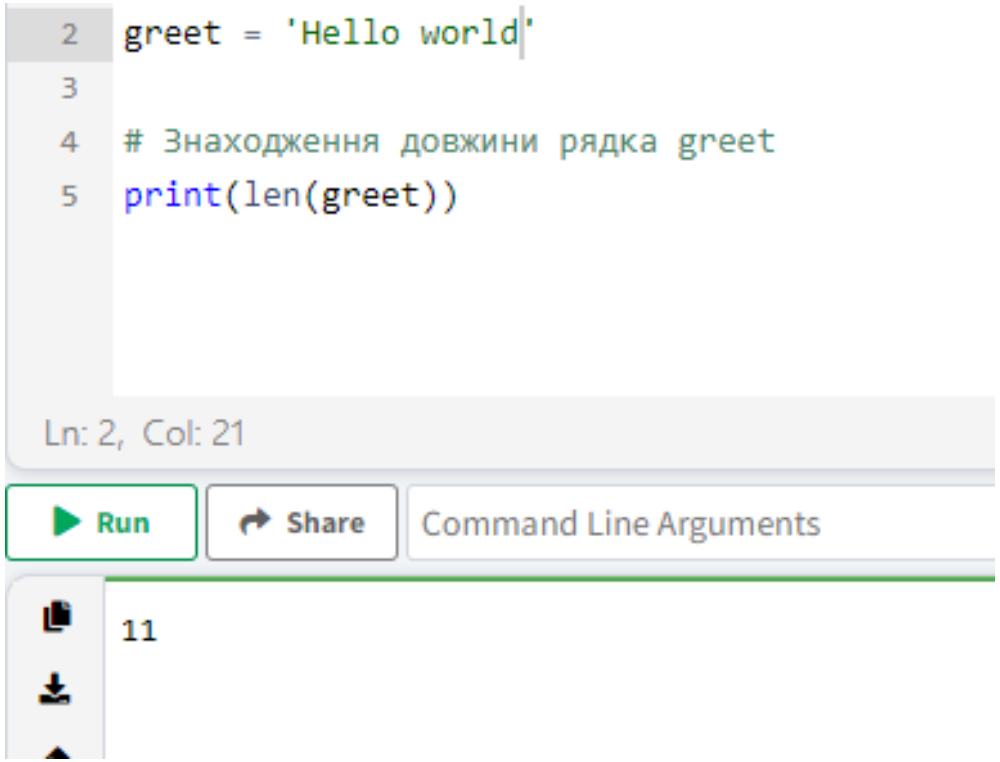
Поєднання (конкатенація) двох і більше рядків. В Python два і більше рядки можуть бути об’єднані (конкатеновані) за допомогою оператора “ + ”.



Ітерація по рядку: в Python можна ітеруватися по рядку за допомогою циклу for



Довжина рядка в Python: дізнатися довжину рядка в Python можна за допомогою методу “ len() ”



Перевірка на належність до рядка: перевірити, чи міститься вказаний підрядок у рядку, можна за допомогою ключового слова “ in ” .

```
1
2 print('am' in 'program')     # виведе True
3 print('at' not in 'battle') # виведе False
```

Ln: 3, Col: 43

Run | Share | Command Line Arguments

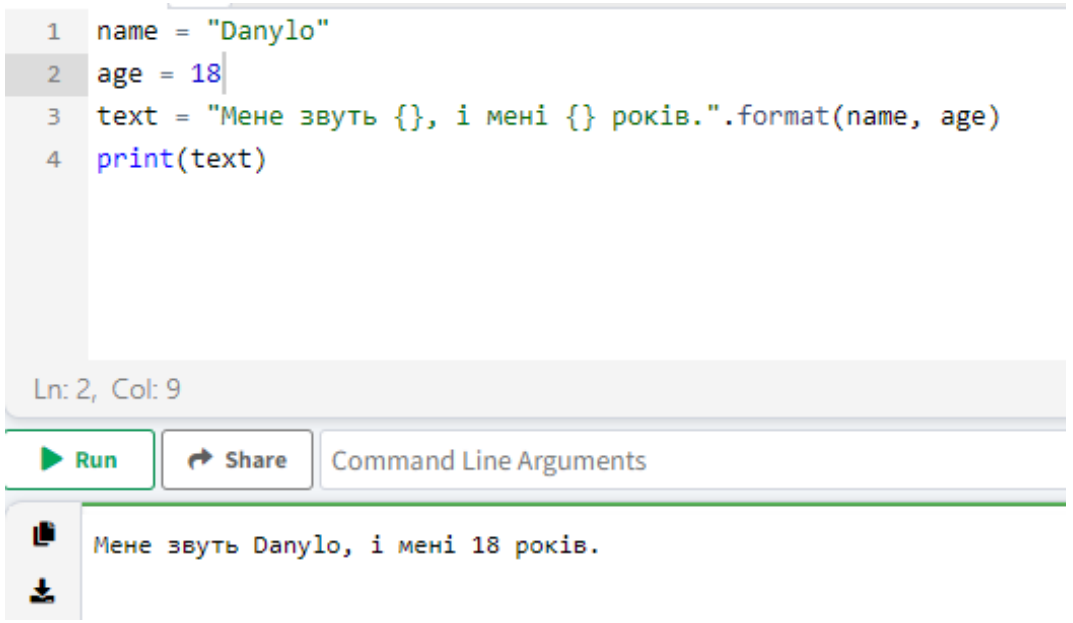
```
True
False
```

3. Методи для роботи з рядками в Python

- **Upper()** – Приводить рядок до верхнього регістра;
- **Lower ()** – Приводить рядок до нижнього регістру;
- **Partition ()** – Повертає кортеж із трьох частин рядка, згідно з вказаним роздільником;
- **Replace ()** – Замінює підрядок усередині рядка;
- **Find ()** – Повертає індекс першого входження заданого підрядка в рядок;
- **Rstrip ()** – Видаляє всі вказані символи, починаючи з кінця рядка;
- **Split ()** – Розділяє рядок згідно з вказаним роздільником;
- **Startswith ()** – Перевіряє, чи починається рядок із вказаного рядка;
- **Insnumeric ()** – Перевіряє, чи всі символи рядка є цифрами;
- **Index ()** – Повертає індекс підрядка.

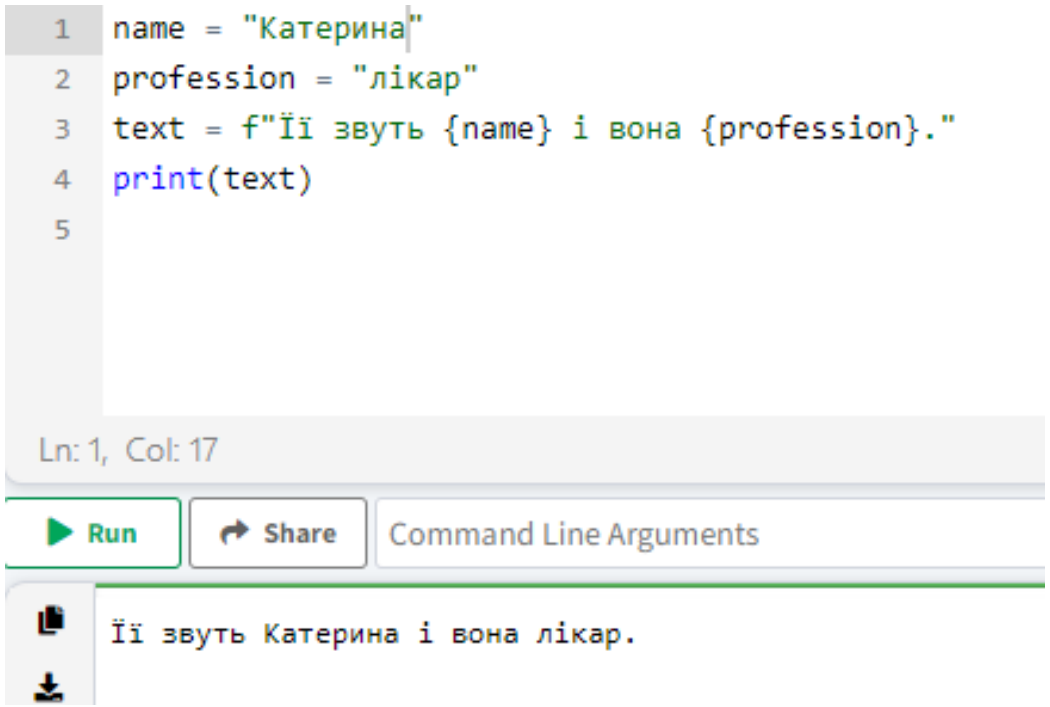
4.Форматування рядків в Python

Використання методу “format()”. Метод format() в Python дозволяє вставляти значення змінних у рядки. Він дозволяє більш гнучке форматування рядків, дозволяючи вставляти значення змінних у певні місця рядка з використанням фігурних дужок {}. Приклад використання методу format():



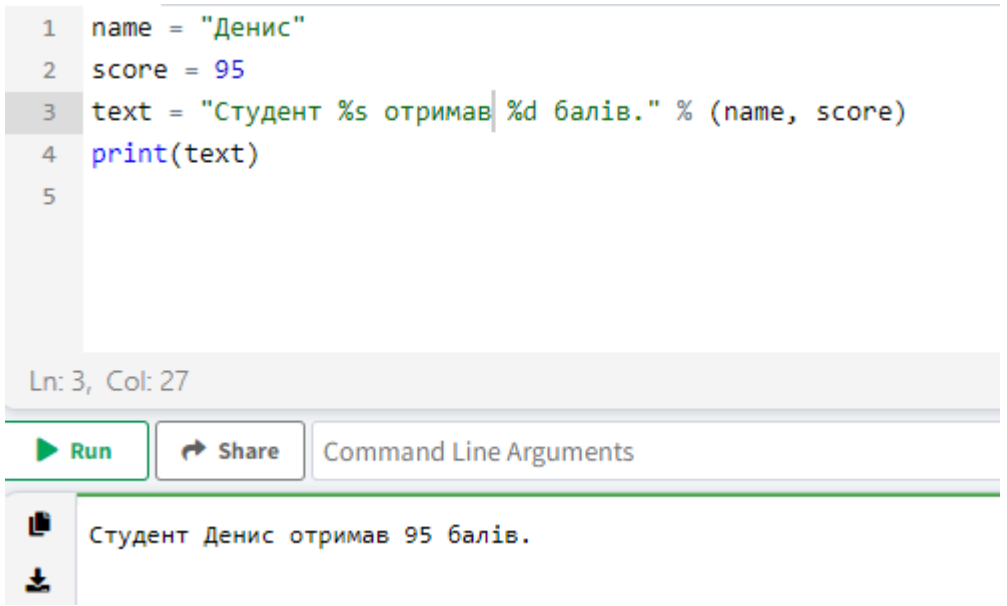
Використання f-строк (f-strings):

f-строки є зручним і новим способом форматування рядків в Python, який дозволяє вставляти значення змінних безпосередньо в рядок. Їх особливістю є використання префіксу f перед лапками рядка та включення змінних в фігурних дужках безпосередньо всередині рядка.
Приклад використання f-строк:



Старий спосіб форматування з використанням оператора %:

Існує старий, але все ще підтримуваний спосіб форматування рядків в Python, використовуючи оператор %. Він дозволяє вставляти значення змінних у рядок, вказуючи специфікатори форматування.



5. Escape-послідовності в Python

Escape-послідовності в Python відносяться до спеціальних послідовностей символів, які використовуються для вставки спеціальних символів або зміни тлумачення рядків. Вони починаються з символа зворотного слешу “ \ ” , за яким слідує один або кілька символів. Вони дозволяють вставляти у рядок символи, такі як лапки, зворотні слеші, табуляції, переведення рядка та інші, не порушуючи синтаксису рядка.

Escape-послідовності

- \\ - оберненна скісна риска;
- \` - одинарна лапка;
- \`` - подвійна лапка;
- \a – символ дзвінка;
- \b – backspace;
- \f – розрив сторінки;
- \n – розрив рядка;
- \r – повернення каретки;
- \t – горизонтальна табуляція;
- \v - вертикальна табуляція;
- \ooo – дозволяє вставляти символ за його восьмиричним кодом;
- \xHH - дозволяє вставляти символ за його шістнадцятковим кодом.

Приклади Escape-послідовності

```
# Вставка символу 'a' за його восьмиричним кодом
my_string = "\141pple"
print(my_string)
```

Ln: 4, Col: 1

Run | Share | Command Line Arguments

```
apple
```

```
# Вставка символу 'A' за його шістнадцятковим кодом
my_string = "\x41pple"
print(my_string)
```

Ln: 4, Col: 1

Run | Share | Command Line Arguments

```
Apple
```

```
print("Це\nприклад\nвикористання\nEscape-послідовності.")
```

Ln: 2, Col: 1

Run | Share | Command Line Arguments

```
Це
приклад
використання
Escape-послідовності.
```

```
print("Це є приклад використання лапок всередині рядка: \"Лапки\"")
```

Ln: 1, Col: 68

Run | Share | Command Line Arguments

```
Це є приклад використання лапок всередині рядка: "Лапки"
```

```
my_string = "Це текст з використанням символу повернення каретки.\rI це нова частина рядка."
print(my_string)
```

Ln: 4, Col: 1

Run | Share | Command Line Arguments

```
Це текст з використанням символу повернення каретки.
I це нова частина рядка.
```

6.Коди символів

Коди символів в Python відносяться до числових значень, які використовуються для представлення окремих символів у вигляді числових значень. У багатьох випадках це відповідає ASCII-кодам або значенням кодування Unicode, які визначають числове представлення кожного символу.

У більшості кодувань, включаючи ASCII та Unicode, кожен символ має відповідне числове значення, що визначає його унікальний ідентифікатор. Наприклад, символ 'A' має відповідне числове значення 65 у ASCII та Unicode.

У Python ви можете використовувати ці числові значення для роботи зі символами та їх кодуванням. Використовуючи ці коди символів, ви можете взаємодіяти зі стрічками, змінювати та обробляти їх залежно від вашої потреби.

В Python, для представлення символів існує кілька різних форматів кодування, які можуть бути використані. Деякі з них включають:

1. ASCII: Використовується для представлення стандартного набору символів ASCII (American Standard Code for Information Interchange);
2. UTF-8: Широко використовується формат кодування, який підтримує майже всі символи світових мов;
3. UTF-16: Інший формат кодування, який підтримує збільшений діапазон символів;
4. UTF-32: Формат кодування, який використовує 4 байти для кожного символу, що дозволяє представляти ще більший діапазон символів.

Python використовує UTF-8 для представлення рядків за замовчуванням, що дозволяє йому працювати з широким спектром символів та мов. Ви можете використовувати символи зі всього цього набору при створенні та роботі зі стрічками в Python.

@ZTUEDUUA

ЖИТОМИРСЬКА ПОЛІТЕХНІКА

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

- **Розвиваємо лідерів**
- **Створюємо інновації**
- **Змінюємо світ на краще**