Лабораторна робота №4

Тема: Робота з рядками у мові Python

Мета: набути навичок роботи з вбудованими функціями для роботи з рядками у Python

Теоретична частина

Рядок складається з послідовності символів.

Тип рядка str.

```
>>> type('2')
<class 'str'>
```

У мові Пітон немає окремого символьного типу. Символ – це просто рядок довжини 1.

Довжина рядків в Пітоні не обмежена або, строго кажучи, обмежена обсягом виділеної оперативної пам'яті.

Рядок з кількох поспіль літералів буде неявно конкатенованим навіть при відсутності знаку +:

```
s = 'Infinite' "blue" 'sky'
```

Рядки в потрійних лапках можуть містити кілька рядків тексту. Пробіли від початку рядка входять в текст:

```
s = """Happy birthday to you,
        My darling Findus!
                Your hostess."""
```

Можна домогтися того ж ефекту і в одинарних лапках, залишаючи в кінці кожного рядка зворотний слеш (але рекомендується так не робити; після \ не повинно бути пробілу в кінці рядка):

```
s = "Happy birthday to you,\
My darling Findus!\
Your hostess."
```

Базові операції

Конкатенація (додавання)

```
>>>
>>> S1 = 'spam'
>>> S2 = 'eggs'
>>> print(S1 + S2)
'spameggs'
```

- Дублювання рядка

```
>>>
>>> print('spam' * 3)
spamspamspam
```

- Довжина рядка

```
>>>
>>> len('spam')
4
```

- Доступ за індексом

```
>>>
>>> S = 'spam'
>>> S[0]
's'
>>> S[2]
'a'
>>> S[-2]
'a'
```

Як видно з прикладу, в Python є можливість доступу по негативного індексу, при цьому відлік йде від кінця рядка.

- Отримання зрізу

Оператор вилучення зрізу: [X: Y]. X - початок зрізу, а Y - закінчення; символ з номером Y в зріз не входить. За умовчанням перший індекс дорівнює 0, а другий - довжині рядка.

```
>>>
>>> s = 'spameggs'
>>> s[3:5]
'me'
>>> s[2:-2]
'ameg'
>>> s[:6]
'spameg'
>>> s[1:]
'pameggs'
>>> s[:]
'spameggs'
```

Крім того, можна задати крок, з яким потрібно витягувати зріз.

```
>>>
>>> s[::-1]
'sggemaps'
>>> s[3:5:-1]
''
>>> s[2::2]
'aeg'
```

Інші функції та методи рядків

При виклику методів необхідно пам'ятати, що рядки в Python відносяться до категорії незмінюваних послідовностей, тобто всі функції і методи можуть лише створювати новий рядок.

```
>>>
>>> s = 'spam'
>>> s[1] = 'b' # Помилка!
Traceback (most recent call last):
  File "", line 1, in
    s[1] = 'b'
TypeError: 'str' object does not support item assignment
```

```
>>> s = s[0] + 'b' + s[2:]
>>> s
'sbam'
```

Тому всі строкові методи повертають новий рядок, який потім слід привласнити змінній.

Таблиця "Функції та методи рядків"

Функція або метод	**Призначення**
S = 'str'; S = "str"; S = '''str'''; S = """str"""	літерали рядків
S = "s\np\ta\nbbb"	екрановані послідовності
S = r"C:\temp\new"	неформатовані рядки (пригнічують екранування)
S = b"byte"	рядок байтів
S1 + S2	конкатенація (додавання рядків)
S1 * 3	повторення рядка
S[i]	звернення за індексом
S[i:j:step]	витяг зрізу
len(S)	довжина рядка
S.find(str, [start],[end])	Пошук підрядка в рядку. Повертає номер першого входження або -1
S.rfind(str, [start],[end])	Пошук підрядка в рядку. Повертає номер останнього входження або -1
S.index(str, [start],[end])	Пошук підрядка в рядку. Повертає номер першого входження або викликає ValueError
S.rindex(str, [start],[end])	Пошук підрядка в рядку. Повертає номер останнього входження або викликає ValueError
S.replace(шаблон, заміна)	Заміна шаблону
S.split(символ)	Розбиття рядка по роздільнику
S.isdigit()	Чи складається рядок з цифр
S.isalpha()	Чи складається рядок з цифр літер
S.isalnum()	Чи складається рядок з цифр або літер
S.islower()	Чи складається рядок із символів в нижньому регістрі
S.isupper()	Чи складається рядок із символів у верхньому регістрі

S.isspace()	Чи складається рядок з невідображаючихся символів (пробіл, символ переводу сторінки ('\f'), "новий рядок" ('\n'), "перевод каретки" ('\r'), "горизонтальна табуляція" ('\t') і "вертикальна табуляція" ('\v'))
S.istitle()	чи починаються слова в рядку з великої літери
S.upper()	Перетворення рядка до верхнього регістру
S.lower()	Перетворення рядка до нижнього регістру
S.startswith(str)	Чи починається рядок S з шаблону str
S.endswith(str)	Чи закінчується рядок S з шаблону str
S.join(список)	Збірка рядка зі списку з роздільником S
ord(символ)	код символу в ASCII
chr(число)	код ASCII в символ
S.capitalize()	Переводить перший символ рядка в верхній регістр, а всі інші в нижній
S.center(width, [fill])	Повертає відцентрований рядок, по краях якого стоїть символ fill (пробіл за замовчуванням)
S.count(str, [start],[end])	Повертає кількість неперетинаючихся входжень підрядка в діапазоні [початок, кінець] (0 і довжина рядка за замовчуванням)
S.expandtabs([tabsize])	Повертає копію рядка, в якій всі символи табуляції замінюються одним або декількома пробілами, в залежності від поточного стовпця. Якщо TabSize не вказано, розмір табуляції вважається рівним 8 пробілів
S.lstrip([chars])	Видалення символів пробілів на початку рядка
S.rstrip([chars])	Видалення символів пробілів в кінці рядка
S.strip([chars])	Видалення символів пробілів на початку і в кінці рядка
S.partition(шаблон)	Повертає кортеж, що містить частину перед першим шаблоном, сам шаблон, і частина після шаблону. Якщо шаблон не знайдений, повертається кортеж, що містить самий рядок, а потім два порожніх рядка
S.rpartition(sep)	Повертає кортеж, що містить частину перед останнім шаблоном, сам шаблон, і частину після шаблону. Якщо шаблон не знайдений, повертається кортеж, що містить два порожні рядки, а потім сам рядок

S.swapcase()	Переводить символи нижнього регістра в верхній, а верхнього - в нижній
S.title()	Першу букву кожного слова переводить в верхній регістр, а всі інші в нижній
S.zfill(width)	Робить довжину рядка не меншою width, в разі потреби заповнюючи перші символи нулями
S.ljust(width, fillchar=" ")	Робить довжину рядки на меншою width, в разі потреби заповнюючи останні символи символом fillchar
S.rjust(width, fillchar=" ")	Робить довжину рядки не меншою width, в разі потреби заповнюючи перші символи символом fillchar
S.format(*args, **kwargs)	форматування рядка

Екрановані послідовності, так звані escape-послідовності, можуть складатися з одного або декількох символів після зворотної косої риски:

Послідовність	Призначення
\ в самому кінці рядка	ігнорується, рядок продовжується на новому рядку
\\	сам символ зворотного слеша (залишається один символ \)
\'	апостроф (залишається один ')
\"	лапки (залишається один символ ")
\n	новий рядок (новий рядок)
\r	повернення каретки
\t	горизонтальна табуляція
\u…	16-бітовий символ Юнікоду в 16-ковий поданні
\U…	32-бітовий символ Юнікоду в 32-ковий поданні
\x…	16-кове значення
\o…	8-кове значення
\0	Символ Null (не ознака кінця рядка)

Зріз (slice) – вилучення з цього рядка одного символу або деякого фрагмента (підрядка).

Є три форми зрізів. Найпростіша форма зрізу: взяття одного символу рядка, а саме, S[i] – це зріз, що складається з одного символу, який має номер i,

при цьому вважаючи, що нумерація починається з числа 0. Тобто якщо S = 'Hello' , то S[0] == 'H', S[1] == 'e', S[2] == 'l', S[3] == 'l', S[4] == 'o '.

Номери символів в рядку (а також в інших структурах даних: списках, кортежі) називаються індексом.

Якщо вказати від'ємне значення індексу, то номер буде відраховуватися з кінця, починаючи з номера 1. Тобто

S [-1] == 'o', S [-2] == 'l', S [-3] == 'l', S [-4] == 'e', S [-5] = = 'H'.

Або у вигляді таблиці:

Рядок S	**H**	**e**	**l**	**l**	**o**
Індекс	S[0]	S[1]	S[2]	S[3]	S[4]
Індекс	S[-5]	S[-4]	S[-3]	S[-2]	S[-1]

Якщо ж номер символу в зрізі рядка S більше або дорівнює len(S), або менше, ніж -len(S), то при зверненні до цього символу рядка відбудеться помилка IndexError: string index out of range.

Зріз з двома параметрами: S[a: b] повертає підрядок з b-a символів, починаючи з символу с індексом a, тобто до символу з індексом b, не включаючи його. Наприклад, S[1: 4] == 'ell', те ж саме вийде якщо написати S[-4: -1]. Можна використовувати як позитивні, так і негативні індекси в одному зрізі, наприклад, S[1: -1] - це рядок без першого і останнього символу (зріз починається з символу з індексом 1 і закінчуватись індексом -1, не включаючи його).

При використанні такої форми зрізу помилки IndexError ніколи не виникає. Наприклад, зріз S[1: 5] поверне рядок 'ello', таким же буде результат, якщо зробити другий індекс дуже великим, наприклад, S[1: 100] (якщо в рядку не більше 100 символів).

Якщо опустити другий параметр (але поставити двокрапку), то зріз береться до кінця рядка. Наприклад, щоб видалити з рядка перший символ (його індекс дорівнює 0, тобто взяти зріз, починаючи з символу з індексом 1), то можна взяти зріз S[1:], аналогічно якщо опустити перший параметр, то зріз береться від початку рядка. Тобто видалити з рядка останній символ можна за допомогою зрізу S[: - 1]. Зріз S[:] збігається з самим рядком S.

Якщо задати зріз з трьома параметрами S[a: b: d], то третій параметр задає крок, як у випадку з функцією range, тобто будуть взяті символи з індексами a, a + d, a + 2 * d і т.д. Якщо вказати значення третього параметра, рівне 2, в зріз потрапить кожний другий символ, а якщо взяти значення зрізу, рівне -1, то символи будуть йти у зворотному порядку.

Приклади зрізів

Введено рядок:

s = input ()

Виведемо третій символ цього рядка:

print (s[2])

Виведемо передостанній символ цього рядка:

print (s[-2])

Виведемо перші п'ять символів цього рядка:

print (s[0: 5])

Виведемо весь рядок, крім останніх двох символів:

print (s[: - 2])

Виведемо всі символи з парними індексами (вважаючи, що індексація починається з 0, тому символи виводяться починаючи з першого):

print (s[:: 2])

Виведемо всі символи з непарними індексами, тобто починаючи з другого символу рядка:

print (s[1 :: 2])

Виведемо всі символи у зворотному порядку:

print (s[:: - 1])

Виведемо всі символи рядка через один в зворотному порядку, починаючи з останнього:

print (s[:: - 2])

Мова програмування Python – сучасна мова програмування, тому вона працює виключно з Unicode-символами (це відноситься до версії 3.x).

Код символу можна визначити за допомогою функції ord. Ця функція отримує на вхід рядок, який повинен складатися рівно з одного символу. Функція повертає код цього символу. Наприклад, ord('A') поверне число 65.

Зворотна функція отримання по числовому коду його номера називається chr.

Словники

Словники в Python – невпорядковані колекції довільних об'єктів з доступом по ключу. Їх іноді ще називають асоціативними масивами або хеш-таблицями.

Щоб працювати зі словником, його потрібно створити. Створити його можна кількома способами. По-перше, за допомогою літерала:

```
>>>
>>> d = {}
>>> d
{}
>>> d = {'dict': 1, 'dictionary': 2}
>>> d
{'dict': 1, 'dictionary': 2}
```

По-друге, за допомогою функції **dict**:

```
>>>
>>> d = dict(short='dict', long='dictionary')
>>> d
{'short': 'dict', 'long': 'dictionary'}
>>> d = dict([(1, 1), (2, 4)])
>>> d
{1: 1, 2: 4}
```

По-третє, за допомогою методу **fromkeys**:

```
>>>
>>> d = dict.fromkeys(['a', 'b'])
>>> d
{'a': None, 'b': None}
>>> d = dict.fromkeys(['a', 'b'], 100)
>>> d
{'a': 100, 'b': 100}
```

По-четверте, за допомогою **генераторів словників**, які дуже схожі на генератори списків.

```
>>>
>>> d = {a: a ** 2 for a in range(7)}
>>> d
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}
```

Тепер спробуємо додати записи в словник і витягти значення ключів:

```
>>>
>>> d = {1: 2, 2: 4, 3: 9}
>>> d[1]
2
>>> d[4] = 4 ** 2
>>> d
{1: 2, 2: 4, 3: 9, 4: 16}
>>> d['1']
Traceback (most recent call last):
  File "", line 1, in
    d['1']
KeyError: '1'
```

Як видно з прикладу, привласнення по новому ключу розширює словник, привласнення за існуючим ключем перезаписує його, а спроба вилучення неіснуючого ключа породжує виключення. Для уникнення виключення є спеціальний метод, або можна перехоплювати виняток.

Що ж можна ще робити зі словниками? Те ж саме, що і з іншими об'єктами: вбудовані функції, ключові слова (наприклад, цикли for і while), а також спеціальні методи словників.

Методи словників

dict.clear() – очищає словник.

dict.copy() – повертає копію словника.

dict.fromkeys (seq [, value]) – створює словник з ключами з seq і значенням value (за замовчуванням None).

dict.get(key [, default]) – повертає значення ключа, але якщо його немає, не викидає виняток, а повертає default (за замовчуванням None).

dict.items() – повертає пари (ключ, значення).

dict.keys() – повертає ключі в словнику.

dict.pop(key [, default]) – видаляє ключ і повертає значення. Якщо ключа немає, повертає default (за замовчуванням кидає виняток).

dict.popitem() – видаляє і повертає пару (ключ, значення). Якщо словник порожній, кидає виняток KeyError. Пам'ятайте, що словники невпорядковані.

dict.setdefault(key [, default]) – повертає значення ключа, але якщо його немає, не кидає виняток, а створює ключ із значенням default (за замовчуванням None).

dict.update([other]) – оновлює словник, додаючи пари (ключ, значення) з other. Існуючі ключі перезаписуються. Повертає None (не новий словник!).

dict.values() – повертає значення в словнику.

Завдання 1:

Створіть програму, яка буде складати випадкові фрази на основі трьох списків зі словами. З кожного списку вона повинна брати випадковим чином слова і поєднувати їх в одну фразу.

Завдання 2:

Візьміть текстовий файл, що містить Вашу улюблену художню книгу.

1. Визначте загальну кількість символів у тексті з пробілами та без пробілів.

2. Визначте загальну кількість слів у тексті, загальну кількість різних слів (без повторів) та кількість унікальних слів, що зустрічаються тільки один раз.

Завдання 3:

Виконайте наступне завдання відповідно до свого варіанту.

Варіант 1. Знайдіть у тексті найдовшу послідовність слів, що повторюється більше одного разу.

Варіант 2. Порівняйте два тексти на наявність однакових послідовностей, що містять не менше 5 слів. (Знадобиться ще один текст, напр., того ж автора).

Варіант 3. Знайдіть у тексті послідовності, що містять не менше 3 слів та повторюються не менше 5 раз, створіть список таких послідовностей та підрахуйте кількість їх повторів.

Варіант 4. Підрахуйте загальну кількість речень у тексті, кількість окличних речень, кількість питальних речень, кількість речень, що закінчуються трикрапкою.

Варіант 5. Складіть списки слів, що характеризують різні емоції - не менше 3 емоцій та не менше 10 слів у кожному списку. Підрахуйте частоту появи у тексті слів з кожного списку. Зробіть висновок про те, які емоції з досліджуваних переважають у тексті.

Варіант 6. Визначте максимальну, мінімальну та середню довжину слів, речень та абзаців у тексті.

Варіант 7. Визначте частоту появи питальних речень у тексті та частоту появи слів в цих реченнях.

Варіант 8. Визначте, які слова найчастіше зустрічаються поряд (перед або після) з вказаним користувачем словом.

Варіант 9. Визначте, 10 найчастіше зустрічаємих послідовностей з N слів у тексті. N вказується користувачем.

Варіант 10. Визначте, з якого слова найчастіше починаються речення у тексті, а також яким найчастіше закінчуються.

Варіант 11. Визначте частоту появи слів з різною довжиною.

Варіант 12. Визначте частоту появи речень з різною кількістю слів.

Варіант 13. Складіть список коренів слів, що характеризують колір; визначте кількість слів у тексті, що характеризують колір, а також те, який колір серед них переважає.

Варіант 14. Визначте процент води у тексті (це кількість "стоп-слів" поділена на загальну кількість слів). Стоп-слова – це слова, які ігноруються при індексації сторінок пошуковими системами, не несуть смислового навантаження, замінюються маркерами і негативно впливають на якість текстів, знижуючи їх корисність. Списки стоп-слів див. в Інтернеті.

Варіант 15. Складіть список слів, що характеризують романтичні почуття (не менше 30 слів у списку) та визначте кількість появи даних слів у тексті і виведіть 5 перших абзаців, де таких слів найбільше.

Контрольні питання:

1. Які базові операції роботи з рядками є у мові Python?
2. Як можна одержувати зрізи рядків? Якими бувають зрізи? Наведіть приклади.
3. Які методи роботи з рядками є у мові Python?
4. Що таке словники? Як з ними працювати? Для чого вони потрібні?
5. Які методи роботи зі словниками є у мові Python?