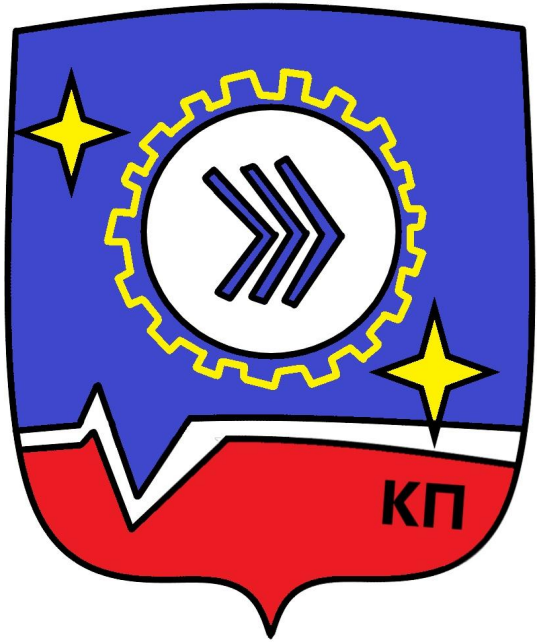
КУРС “Программирование на PYTHON”
13.10.2018 ЛЕКЦИЯ 03



ПОВТОРЯЕМ ПРОЙДЕННОЕ

ТИПЫ ДАННЫХ

- int - 1,2,3
- bool - 0,1
- float - 3.14
- str - “hello”

СТРУКТУРЫ

- list - [1,2,3]
- tuple - (1,2,3)
- set - {1,2,3}
- dict - {1:2,3:4}

УПРАВЛЕНИЕ ПОТОКОМ


```
if
elif
else
```

```
while
```

```
for
```

```
try
except
finally
```

ФУНКЦИИ


```
def имя(аргументы):
    операции
    return результат
```

import


```
import random
print(random.randint(1, 6))
print(random.choice(“любит”, “не любит”))

from random import randint
print(randint(0, 41))
```

ПАРСЕР


```
with open('lt1.txt') as wp_file:
    wp_text = wp_file.read()
    wp_list = wp_text.split('.')
    while True:
        search_word = input('Введите слово для поиска: ')
        search_word = search_word.strip()
        for index, sentence in enumerate(wp_list):
            if search_word in sentence:
                print(index,
                      sentence.replace('\n', ' ').strip())
                break
```

ПАРСЕР
Сложный поиск


```
while True:
    search_word_1 = input('Введите первое слово
      для поиска: ')
    search_word_1 = search_word_1.strip()
    search_word_2 = input('Введите второе слово
      для поиска: ').strip()
    for index, sentence in enumerate(wp_list):
        if search_word_1 in sentence and
          search_word_2 in sentence:
            print(index, sentence.replace('\n',
              ' ').strip())
            break
```

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

МОДУЛЬ re

```
import re

re.findall()

re.findall(шаблон, строка) -> list
```

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

https://docs.python.org/3/library/re.html
https://tproger.ru/translations/regular-expression-python/#

```
.   Один любой символ, кроме новой строки \n.
?   0 или 1 вхождение шаблона слева
+   1 и более вхождений шаблона слева
*   0 и более вхождений шаблона слева
\w  Любая цифра или буква
\W  Все, кроме буквы или цифры
\d  Любая цифра [0-9]
\D  Все, кроме цифры
\s  Любой пробельный символ
\S  Любой непробельный символ)
```

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

https://docs.python.org/3/library/re.html
https://tproger.ru/translations/regular-expression-python/#

\b Граница слова
[..] Один из символов в скобках
[^..] – любой символ, кроме тех, что в скобках
\ Экранирование специальных символов
\. означает точку или \+ – знак «плюс»
^ и $ Начало и конец строки соответственно
{n,m} От n до m вхождений ({,m} – от 0 до m)
a|b Соответствует a или b

ПАРСЕР
Использование RE


```
with open('lt1.txt') as wp_file:
    wp_text = wp_file.read()
    wp_list = wp_text.split('.')
    print(len(wp_list))


with open("lt1.txt") as wp_file:
    wp_text = wp_file.read()
    wp_list = re.split(r'[.?!]+', wp_text)
    print(len(wp_list))
```

ПАРСЕР
Функции


```
def file_to_list():
    pass

def find_sentences():
    pass

def main():
    file_to_list()
    find_sentences()

if __name__ == '__main__':
    main()
```

ПАРСЕР
Функции


```
def file_to_list(file_name):
    with open(file_name) as file_desc:
        file_text = file_desc.read()
        return re.split(r'[.?!]+', file_text)

def find_sentences():
    pass

def main():
    wp_list = file_to_list("lt1.txt")
    find_sentences()

if __name__ == '__main__':
    main()
```

ПАРСЕР
Функции


```
def file_to_list(file_name):
    with open(file_name) as file_desc:
        file_text = file_desc.read()
        return re.split(r'[.?!]+', file_text)
```

```
def find_sentences(init_list, search_word_1,
search_word_2):
    while True:
        for sen_index, sentence in enumerate(init_list):
            if search_word_1 in sentence and
              search_word_2 in sentence:
                return True, sen_index,
                  sentence.replace('\n', ' ').strip()
            else:
                return False, '', ''
```

ПАРСЕР
Функции


```
def main():
    wp_list = file_to_list("lt1.txt")
    search_word_1 = input('Введите первое слово для
поиска: ').strip()
    search_word_2 = input('Введите второе слово для
поиска: ').strip()
    result, sen_index, sentence =
find_sentences(wp_list, search_word_1, search_word_2)
    if result:
        return sen_index, sentence

if __name__ == '__main__':
    main()
```

СЕРТИФИКАЦИЯ
ВОПРОС

Укажите результат выполнения скрипта:

```
foo = (1,)
bar = foo
bar += (1,)
print (foo)
```

Вариант 1 скрипт не будет выполнен, так как содержит ошибки

Вариант 2 (1)

Вариант 3 (1,1)

Вариант 4 (1,)

СЕРТИФИКАЦИЯ
ВОПРОС

Укажите результат выполнения скрипта:

```
foo = (1,)
bar = foo
bar += (1,)
print (foo)
```

Вариант 1 скрипт не будет выполнен, так как содержит ошибки

Вариант 2 (1)

Вариант 3 (1,1)

Вариант 4 (1,)

int

- размер ограничен только объемом памяти
- Системы счисления:
  - двоичная 0b00011101
  - восьмеричная 0o54372310
  - шестнадцатеричная 0xDEF45

int
ОПЕРАЦИИ


```
abs(int) - абсолютное значение
+    +=
-    -=          a = a + 1
/    /=          a += 1
//   //=
%    %=          a = int(7.6)
**   **=         a = int('7')
```

int
ФУНКЦИИ


```
abs(int) - абсолютное значение
divmod(x, y) - частное и остаток
pow(x, y) - **
round(x, [n]) - округление до n

x = int() # x = 0
```

int
ПРЕОБРАЗОВАНИЯ


```
bin(int) - двоичное представление
hex(int) - 16-ное представление
oct(int) - 8-ное представление
int(x)
int('0001010', 2)
```

int
БИТОВЫЕ ОПЕРАЦИИ


```
a | b - OR (ИЛИ)
a ^ b - XOR (исключающая ИЛИ)
a & b - AND (И)
a << b - сдвиг влево на b
a >> b - сдвиг вправо на b
-a - инверсия
```

int
БИТОВЫЕ ОПЕРАЦИИ

a	b	OR	XOR	AND
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

int
БИТОВЫЕ ОПЕРАЦИИ


```
0b000010 << 1 = 0b000100
2 << 1 = 4

0b1100 >> 1 = 0b0110
12 >> 1 = 6
0b1100 >> 2 = 0b0011
12 >> 2 = 3
```

int
ЛОГИЧЕСКИЕ ОПЕРАЦИИ


```
True
False
a = bool('some')
```

Ленивые операции

```
exp1 and exp2 - если exp1 = 0
exp1 or exp2  - если exp1 = 1
```

СКРИПТ
ПОИСК COS

Не используя стандартные функции, вычислить с точностью eps > 0

Считать, что требуемая точность достигнута, если очередное слагаемое по модулю меньше eps; все последующие слагаемые можно уже не учитывать. Вложенные циклы не использовать.

Подсказка: в двух разных переменных храните отдельночислитель и знаменатель очередного слагаемого и на каждом шаге вычисляйте новые числитель и знаменатель через предыдущие значения.

$$y = \cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + \frac{(-1)^n x^{2n}}{(2n)!} + \ldots$$

СКРИПТ
ПОИСК ПОДСТРОКИ

В заданную непустую строку входят только цифры и буквы.

Определить, удовлетворяет ли строка следующему свойству: строка содержит (помимо букв) только одну цифру, причем ее числовое значение равно длине строки.