ЛАБОРАТОРНАЯ РАБОТА 4

Содержание

Стандартные структуры данных: списки `list`

Инициализация и индексация списков

Задание № 0.1. Определить результат вывода функцией `print()` значений следующих переменных:

(a) `tens = [10, 20, 30, 40, 50, 60, 70, 80, 90]`

(b) `prices = [3999.99, 6999.99]`

(c) `planets = ["Mercury", "Venus", "Earth"]`

(d) `mix = [1000, 9.99, "hello ", [1, 2, 3]]`

(e) `repeats = [1, 0, 1, 1, 0, 0, 0, 1]`

(f) `empty = []`

Задание № 0.2. Определить результат вывода функцией `print()` значений следующих выражений (считать, что переменные определены в задании 0.1):

(a) `tens[0]`

(b) `tens[1::2]`

(c) `prices[1]`

(d) `planets[::-1]`

(e) `planets[0][0] + planets[2][1:3] + planets[1][-1]`

(f) `mix[2] * mix[3][2]`

(g) `len(tens)`

(h) `len(mix)`

(i) `len(empty)`

Свойство изменяемости списков

Задание № 0.3. Определить результат вывода функцией `print()` значения переменной `zoo = ["giraffe", "panda", "lion"]`:

(a) `zoo[0] = "monkey"`

(b) `zoo[0:2] = ["crocodile"]`

(c) `zoo = zoo + ["tiger", "wolf"]`

Методы обработки списков

Метод	Действие
`append(`*`element`*`)`	Добавляет *`element`* в конец списка
`extend(`*`list2`*`)`	Расширяет список с использованием элементов из списка *`list2`*
`index(`*`element`*`)`	Возвращает наименьший индекс списка, содержащего *`element`*
`insert(`*`index, element`*`)`	Вставляет в список *`element`* по индексу *`index`*
`pop()`	Удаляет и возвращает самый последний элемент из списка
`reverse()`	Изменяет порядок элементов списка на обратный
`remove(`*`element`*`)`	Удаляет первое вхождение *`element`* из списка
`sort()`	Сортирует список
`copy()`	Возвращает копию списка
`count(`*`element`*`)`	Возвращает количество элементов, равных *`element`*, в списке

Задание № 0.4. Определить результат следующих выражений. Считать определенным список `colors = ["red", "green", "blue"]`:

(a) `colors.append("yellow")`

(b)
```
darks = ["black", "navy blue"]
colors.extend(darks)
```

(c) `colors.index("blue")`

(d) `colors.insert(-2, "white")`

(e) `last = colors.pop()`

(f) `colors.reverse()`

(g) `colors.remove("green")`

(h)
```
bw = colors.copy()
bw[2:] = []
```

(i) `bw.count("red")`

(j) `colors.sort()`

Методы обработки строк, связанные со списками

Метод	Возвращаемое значение
`split([`*`sep`*`])`	Возвращает список подстрок из исходной строки, которые разделены заданной строкой *`sep`*. Если строка *`sep`* не задана, то разделителем является любое количество пробельных символов
`join([`*`source_list`*`])`	Использует строку как разделитель при объединении списка *`source_list`* строк

Задание № 0.5. Определить результат следующих выражений:

(a)
```
numerals = "one two three four five six seven eight nine ten"
numerals.split()
```

(b)
```
best_books = "Moby Dick, War and Peace, Hamlet"
best_books.split(", ")
```

(c)
```
authors = "Michael Park AND Erin Leahey AND Russell J. Funk"
authors.split(" AND ")
```

(d)
```
directories = ["home", "study", "programming", "python"]
"/".join(directories)
```

(e)
```
terms = ["x", "y", "z"]
" + ".join(terms)
```

Оператор `in` и списки

Задание № 0.6. Определить результат следующих выражений:

(a)
```
1 in [1, 2, 3]
```

(b)
```
0 in [1, 2, 3]
```

(c)
```
user = "Michelangelo"
registered_users = ["Leonardo", "Raphael", "Donatello"]
user in registered_users
```

(d)
```
element = "fifth"
element not in ["first", "second", "third", "fourth"]
```

```python
# Пример использования оператора in в ветвлении if

commands = ["start", "show", "exit"]

print(f"{commands[0]} - пройти опрос")
print(f"{commands[1]} - показать статистику")
print(f"{commands[2]} - выйти")

user_choice = input("\nВыберите действие: ")

if user_choice in commands:
    print("Ваш выбор: ", user_choice)
else:
    print("Нет такой команды")
```

```python
# Пример использования оператора in в цикле while

answer = "yes"

while answer.lower() not in ["нет", "хватит", "n", "no", "exit"]:
    user_string = input("Введите строку:  ")
    print("Строка наоборот:", user_string[::-1])

    answer = input("Хотите продолжить? (да|нет) ")
```

Управление ходом выполнения программы: цикл for

```
for элемент in список:
    блок_инструкций_для_каждого_элемента
```

```python
# Простой пример цикла for

sentence = "Каждый охотник желает знать, где сидит фазан"
for word in sentence.split():
    print(word)
```

```python
# Пример оформления длинного списка в коде

courses = [
    "Algebra",
    "Data Analysis",
    "Network Analysis",
    "English Language",
    "Demography",
    "Discrete Mathematics",
    "Big Data in the Social Sciences",
    "Programming in Python for Data Analysis"
]

for course in courses:
    print(course)
```

Задание № 0.7. С помощью цикла `for` по заданному списку слов сформировать новый список, добавив в конец каждого слова заданную подстроку.

Например, для исходного списка `["temp", "todo", "task"]` и строки `".txt"` должен получиться список `["temp.txt", "todo.txt", "task.txt"]`.

Стандартные структуры данных: кортежи `tuple`, словари `dict`, множества `set`

```python
# Пример работы с кортежами tuple

zero = (0, "zero")
print(zero[0], zero[1])

#
x, y = 1, 2
print(f"x = {x}")
print(f"y = {y}")

#
x, y = y, x
```

```python
# Пример работы со словарями dict

ages = {"Jack" : 19, "Mary" : 20, "Adam" : 18, "Bob" : 20}

name = "Adam"
print(f"{name} is {ages[name]} year old\n")

print("Люди, которым 20 лет:")
for name in ages:
    if ages[name] == 20:
        print(name)
```

```python
# Пример работы с множествами set

basket = {"молоко", "хлеб", "колбаса", "хлеб"}
print(basket)       # {'молоко', 'колбаса', 'хлеб'}

"огурцы" in basket  # False

#
your_favorite_film_genres = [
    "horror",   "comedy",           "thriller",
    "comedy",   "science fiction",  "comedy",
    "comedy",   "action",           "drama",
    "mystery",  "comedy",           "science fiction"
]

film_genres = set()
for genre in your_favorite_film_genres:
    film_genres.add(genre)

print(film_genres)
```

Требования к программам

1. Файл с исходным кодом называть по шаблону: «Фамилия_номер_задания» английским алфавитом (пример: `Ivanov_4_42.py`).
2. Файл с исходным кодом должен начинаться с многострочного описательного комментария (см. базовый шаблон ниже).
3. Имена переменных выбирать разумными.
4. Оформлять понятный ввод данных / вывод результата.
5. Задавать списки и строки таким образом, чтобы изменение значения списка или строки требовало внесения правки кода только в одном месте кода.

Базовый шаблон для всех лабораторных

```
'''Фамилия Имя. Номер задания

Краткая формулировка задания
'''

# Код программы
```

Пример программы

```
'''Фамилия Имя. Задание № 4.0

Дан список с некоторыми ключевыми словами языка Python.
Написать программу, которая
спрашивает пользователя по этому списку, знаком ли он со словом,
и формирует новый список неизвестных для пользовтеля слов.
'''
keywords = ['False', 'True', 'break', 'for', 'if', 'in', 'not']

yes_word  = "да"
no_word   = "нет"
exit_word = "выход"

to_learn = []

print(f"Вы знаете это слово в Python? ({yes_word}|{no_word}|{exit_word})")
for keyword in keywords:
    answer = input(keyword.ljust(10))
    if answer == exit_word:
        break
    elif answer != yes_word:
        to_learn.append(keyword)

if len(to_learn) > 0:
    print("Слова, которые нужно подучить:\n", to_learn)
elif answer != exit_word:
    print("Вы молодец!")
```

Индивидуальные задания

Задание № 4.1. Дан список с целыми положительными числами. Получить новый отсортированный список со всеми четными числами из первого списка.

Задание № 4.2. Дан список с целыми числами. Получить новый отсортированный список со всеми отрицательными числами из первого списка.

Задание № 4.3. Дан список с целыми числами. Получить новый список с квадратами этих чисел.

Пример:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Результирующий список

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Задание № 4.4. Даны числа дня (day), месяца (month) и года (year). Вывести дату в формате: `"Month dd, yyyy"`, где `dd` и `yyyy` — данные числа, а `Month` — название месяца, соответствующего числу `month`.

Пример:

```
day, month, year = 24, 03, 2023
```

Вывод программы:

```
"March 24, 2023"
```

Задание № 4.5. Дана строка со словами, разделенными пробелами. Получить строку с этими же словами, но в отсортированном порядке.

Пример:

```
initial_string = "program file text code number line"
```

Вывод результирующей строки:

```
code file line number program text
```

Задание № 4.6. Дан список из слов `"yes"` и `"no"`. Получить новый список с логическими значениями, заменяя `"yes"` на `True` и `"no"` на `False`.

Пример:

```
answers = ["yes", "yes", "no", "yes", "yes", "no", "yes", "no"]
```

Результирующий список:

```
[True, True, False, True, True, False, True, False]
```

Задание № 4.7. Дан список из слов `"yes"` и `"no"`. Определить долю слов `"yes"`.

Задание № 4.8. Дана строка со словами, разделенными пробелами, и целое положительное число `min_len`. Получить список со всеми словами из строки, длина которых не меньше `min_len`.

Пример:

```
string_with_words = "многообразие смысл феномен массив данные объём"
min_len = 7
```

Результирующий список:

```
["многообразие", "феномен"]
```

Задание № 4.9. Дана строка со словами, разделенными пробелами. Получить список, содержащий все слова из строки без повторений.

Пример:

```
string = "the sixth sick sheikh's sixth sheep is sick"
```

Результирующий список:

```
["the", "sixth", "sick", "sheikh's", "sheep", "is"]
```

Задание № 4.10. Дан список со словами и два слова `old` и `new`. Заменить в этом списке все слова `old` на `new`.

Пример:

```
code = ["one", "none", "none", "one", "none", "none", "none"]
old = "none"
new = "zero"
```

Результирующий список:

```
["one", "zero", "zero", "one", "zero", "zero", "zero"]
```

Задание № 4.11. Дано два списка. Получить третий список, который содержит все элементы второго списка, не входящие в первый.

Пример:

```
fruits = ["apple", "orange", "banana"]
breakfast = ["bread", "orange", "banana", "butter", "egg"]
```

Результирующий список:

```
["bread", "butter", "egg"]
```

Задание № 4.12. Дан список с простыми арифметическими выражениями вида `"x op y"`, где `x`, `y` — произвольные числа, а знак `op` может быть одним из символов: `+`, `-`, `*`, `/`. Вычислить значения всех выражений и вывести их на экран в формате: `x op y = res`.

Пример:

```
expressions = ["2 + 2", "9 * 7", "123.50 - 77.45", "32768 / 128"]
```

Вывод программы:

```
2.0 + 2.0 = 4.0
9.0 * 7.0 = 63.0
123.5 - 77.45 = 46.05
32768.0 / 128.0 = 256.0
```

Задание № 4.13. Дан список фамилий и инициалов. Фамилии и инициалы разделены пробелом, инициалы написаны без пробелов. Получить новый список, в котором в строках инициалы поставлены перед фамилией.

Пример:

```
poets = [
    "Пушкин А.С.",
    "Лермонтов М.Ю.",
    "Есенин С.А.",
    "Ахматова А.А.",
    "Маяковский В.В."
]
```

Результирующий список:

```
["А.С. Пушкин", "М.Ю. Лермонтов", "С.А. Есенин", "А.А. Ахматова",
"В.В. Маяковский"]
```

Задание № 4.14. Дан список имен файлов, расширение и подстрока. Получить новый список имен файлов, в котором к началу каждого имени файла заданного расширения добавлена данная подстрока.

Пример:

```
extension = ".pdf"
list_of_files = [
    "temp.txt",
    "lab_4.pdf",
    "book.pdf",
    "project.doc",
```

```
    "lecture.pdf",
    "data.dat"
]
substring = "mine_"
```

Результирующий список:

```
["temp.txt", "mine_lab_4.pdf", "mine_book.pdf", "project.doc",
"mine_lecture.pdf", "data.dat"]
```

Задание № 4.15. Дан список имен файлов, расширение и подстрока. Получить новый список имен файлов, в котором к концу каждого имени файла заданного расширения добавлена данная подстрока.

Пример:

```
extension = ".pdf"
list_of_files = [
    "temp.txt",
    "lab_4.pdf",
    "book.pdf",
    "project.doc",
    "lecture.pdf",
    "data.dat"
]
substring = "_copy"
```

Вывод результирующего списка:

```
["temp.txt", "lab_4_copy.pdf", "book_copy.pdf", "project.doc",
"lecture_copy.pdf", "data.dat"]
```

Полезные ссылки

Официальный сайт по языку Python:

https://www.python.org/

Официальная документация по языку Python 3:

https://docs.python.org/3/

Онлайн-интерпретаторы языка Python:

https://www.onlinegdb.com/online_python_compiler

https://www.online-python.com/

https://www.programiz.com/python-programming/online-compiler/