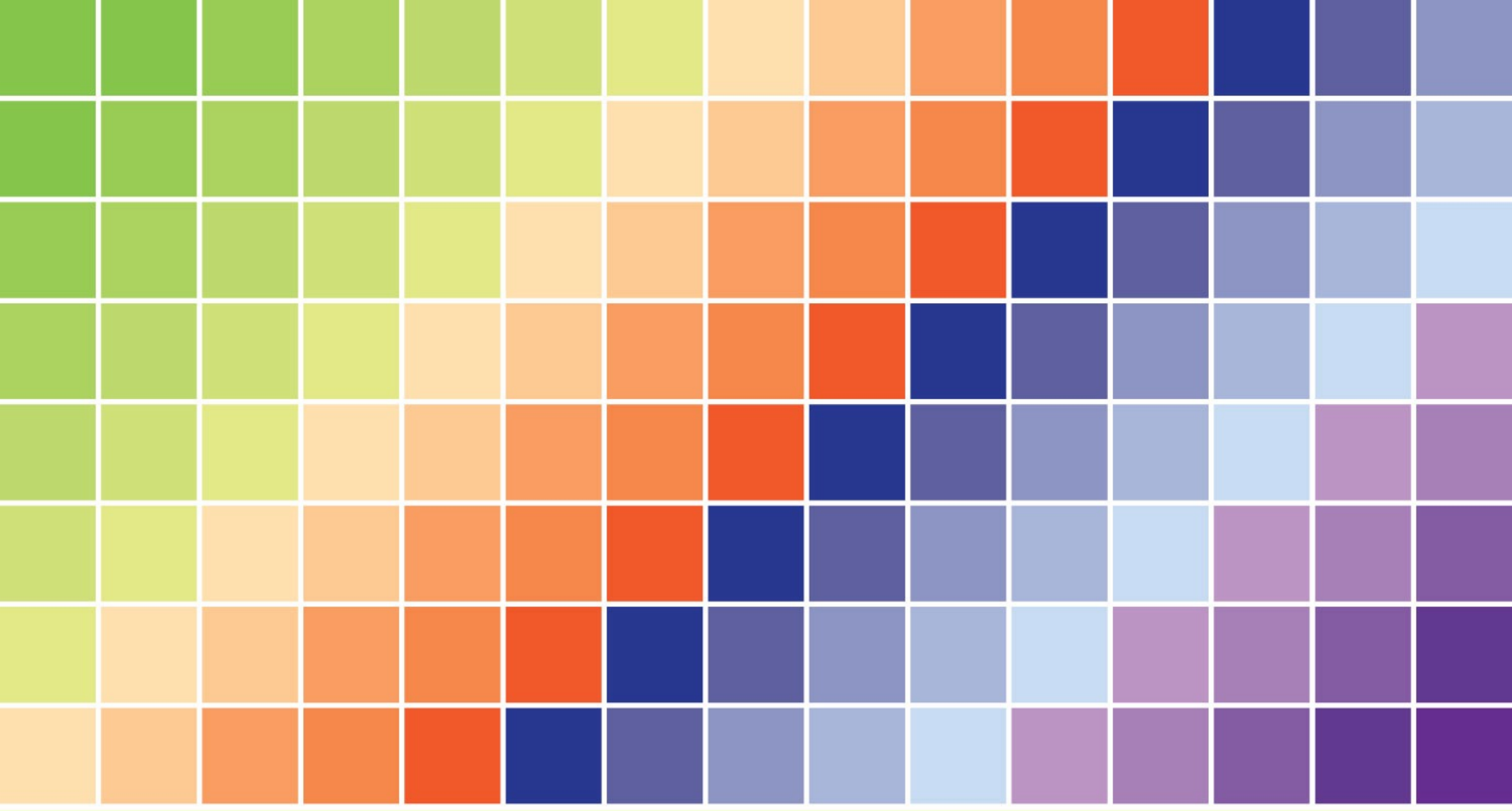


School of Computer Science

СПИСКИ. ЦИКЛ FOR

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы

ВОПРОС ПО ПРОШЛОМУ ЗАНЯТИЮ

Схема оператора цикла:

```
while <логическое выражение>:
    Код, выполняемый при True
    для логического
    выражения – тело цикла
else:
    Блок кода, выполняемый при
    некоторых условиях
```

При каких условиях выполнится блок кода после else: в этом операторе?

ВОПРОС ПО ПРОШЛОМУ ЗАНЯТИЮ

Рассмотрите этот код:

```
count = 0
# Начало цикла
while True:
    count += 1
    if count == 5:
        continue
    elif count > 10:
        break
else:
    # Блок else цикла
    count = 28
# Конец цикла
print(count)
```

Куда передает управление:
- **continue**
- **break**

?

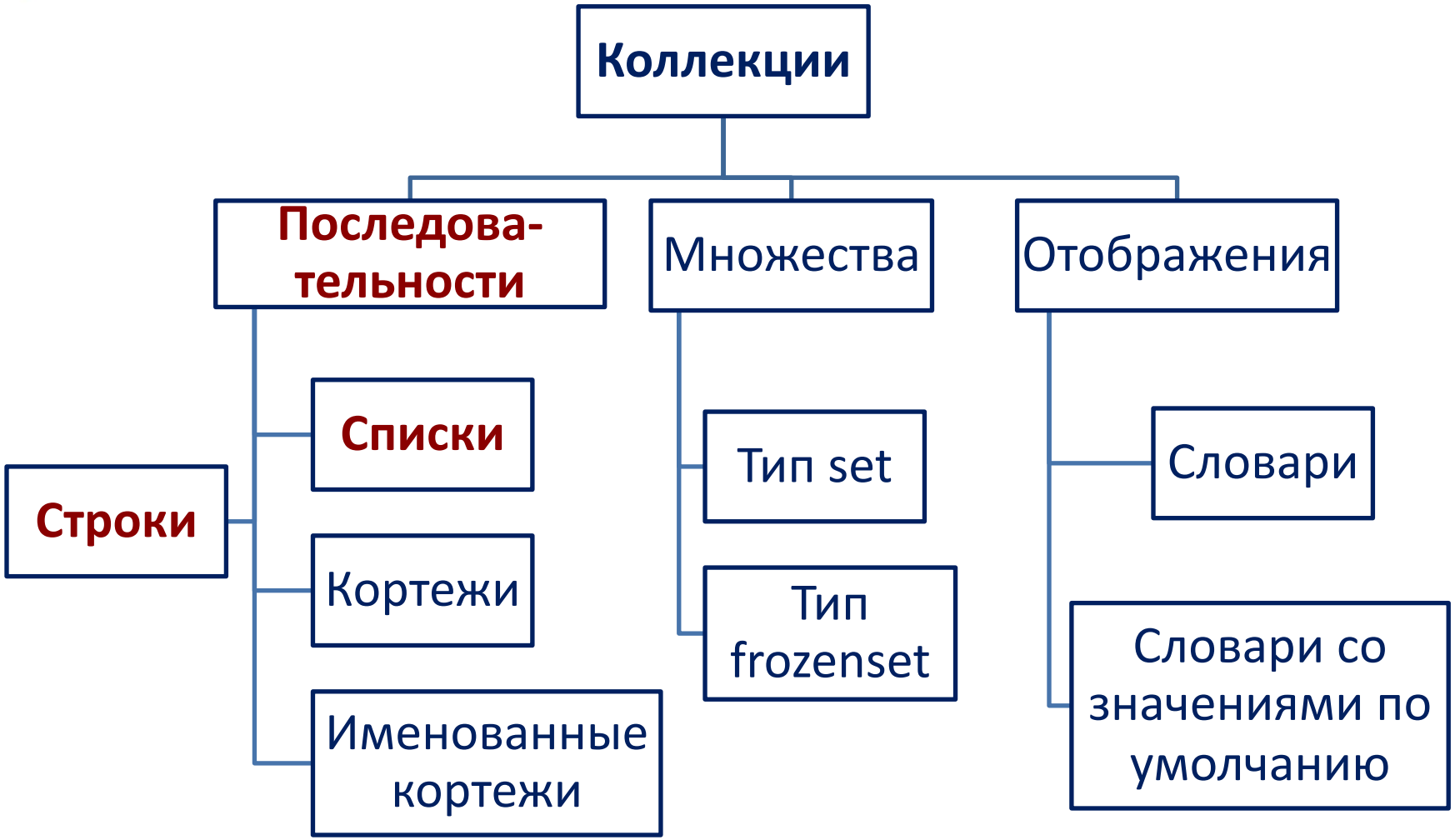
Что напечатает эта программа ?

ПРОВЕРОЧНЫЙ ВОПРОС

Сколько всего знаков * будет выведено после исполнения фрагмента программы?

```
1   i = 0
2   while i < 5:
3       print('*')
4       if i % 2 == 0:
5           print('**')
6       if i > 2:
7           print('***')
8       i = i + 1
9
```

ТИПЫ КОЛЛЕКЦИЙ В PYTHON



ПОСЛЕДОВАТЕЛЬНОСТИ

- Составной тип данных, поддерживающий следующие операции:
  - Конкатенация и тиражирование: + и *
  - Проверки на вхождение in и not in
  - Функция определения размера len(object)
  - Индексация: object[index]
  - Извлечение срезов: object[start:stop:step]
  - Итерации, гарантирующие строгую последовательность элементов

- Примеры последовательностей:
  - str, list, tuple

СПИСКИ

- Объявление списка:
  - list()
  - []
  - [*элементы ч/з запятую*]

- Примеры:

```
- lst1 = list()   # пустой список
- lst2 = []       # тоже пустой список
- lst3 = [-17.5, 'text', 83]
- lst4 = [['Monday', -18], # список
          ['Tuesday', +5], # с вложен-
           'Unknown' ] # ными списками
```

МЕТОДЫ СПИСКОВ

Вызов	Описание
L.append(x)	Добавляет элемент x в конец списка L. Возвращает None
L.count(x)	Возвращает число вхождений элемента x в список L
L.extend(m) **L += m**	Добавляет в конец списка L все элементы итерабельного объекта m; то же что и ‘+=’. Возвращает None
L.index(x [, start, end])	Возвращает индекс самого первого (слева) вхождения элемента x в список L (или в срез start:end списка L) или возбуждает исключение ValueError

МЕТОДЫ СПИСКОВ

Вызов	Описание
L.insert(i,x)	Вставляет элемент x в список L в позицию int i. Возвращает None
L.pop()	Удаляет самый последний элемент из списка L и возвращает его же
L.pop(i) **del L[i]**	Удаляет из списка L элемент с индексом int i и возвращает его же. del L[i] удаляет i-ый элемент, не возвращая его
L.remove(x)	Удаляет первый найденный элемент x из списка L или возбуждает исключение ValueError, если элемент x не будет найден. Возвращает None

МЕТОДЫ СПИСКОВ

Вызов	Описание
L.clear()	Полностью очищает список. Возвращает None
L.copy()	Возвращает поверхностную копию списка
L.reverse()	Переставляет в памяти элементы списка в обратном порядке
L.sort()	Сортирует список в памяти. По умолчанию, в порядке возрастания элементов. Возвращает None. Элементы списка должны быть однотипными, иначе – TypeError

СПИСКИ И СТРОКИ

- Преобразование из строки в список:
  - **str.split(sep=None, maxsplit=-1) → list**
  - Возвращает список слов в строке, используя sep как разделитель слов
  - Пример: '1,2,3'.split(',') → ['1', '2', '3']

- Слияние списка в строку:
  - **str.join(the_string_list) → str**
  - Возвращает строку, в которую сливаются все элементы списка через заданный разделитель
  - Пример: '.'.join(['ab', 'pq', 'rs']) → 'ab.pq.rs'

- Смотрите примеры по ссылке http://pythontutor.ru/lessons/lists/#section_2 в главе «2. Методы split и join»

ПРИМЕРЫ НА СПИСКИ

- Пример работы со списком смотрите в shop_list.py

- Пример генерации пароля через случайный выбор символов в списке смотрите в awful_password.py

- Другие примеры:
- https://pythonru.com/primery/python-spiski-primery

ИЗМЕНЯЕМЫЕ И НЕИЗМЕНЯЕМЫЕ ПОСЛЕДОВАТЕЛЬНОСТИ

- Строки – **неизменяемые** (immutable):
  - Нельзя присваивать элемент строки
    string[n] = ‘X’
  - Индексация допустима только справа от оператора присваивания

- Списки – **изменяемые** (mutable):
  - Можно присваивать (изменять) существующий элемент списка

- Рассмотрим визуальные примеры

ЦИКЛ FOR

```
for <переменная> in
    <итерабельный объект>:
    код, выполняемый для
    каждого элемента
    итерабельного объекта…
[else:
    код, выполняемый, если блок
    for нормально завершился
    или не выполнялся вовсе
]
```

Отступы обяза-тельны!

Допустимы **break** и **continue** в теле цикла, а также **else** в конце. Смысл тот же, что и для **while**

ЦИКЛ FOR. ИТЕРАЦИИ

```
>>>
>>> # итерации по строке, т.е. перебор всех символов строки
>>> word = "Python"
>>> for letter in word:
        print(letter)


P
y
t
h
o
n
>>>
>>> # итерации по списку, т.е. перебор всех символов строки
>>> day_schedule = ["Wake-up", "Coffee", "Work", "Lunch", "Work", "Home"]
>>> for item in day_schedule:
        print(item)


Wake-up
Coffee
Work
Lunch
Work
Home
>>>
```

RANGE. СПОСОБ ГЕНЕРАЦИИ ИНДЕКСОВ ДЛЯ ПОСЛЕДОВАТЕЛЬНОСТИ

- range() генерирует индексы, с помощью которых можно адресоваться к элементам последовательности

```
range([start,] stop [, step])
→ <итерируемый объект>
```

Возвращает последовательность чисел от start (включая) до stop (исключая) с шагом step

ИСПОЛЬЗОВАНИЕ RANGE ДЛЯ ЦИКЛА FOR

- range() генерирует индексы, с помощью которых можно адресоваться к элементам последовательности:

```
for index in range([start,] stop
[, step]):
    <тело цикла, использующее
     переменную index>
```

range() *возвращает последовательность чисел от* ***start*** *до* ***stop*** *с шагом* ***step***

RANGE. ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ

```
range([start,] stop [, step])
```

– Возвращает целочисленный итератор:

1. С одним аргументом (stop) итератор представляет последовательность целых чисел от 0 до (stop-1)
2. С двумя аргументами (start, stop) – последовательность целых чисел от start до (stop-1) с шагом 1
3. С тремя аргументами – цепочку целых чисел от start до (stop-1) с шагом step

RANGE() ПОХОЖ НА СПИСОК

- range() возвращает последовательность чисел от start до stop с шагом step
- Эта последовательность легко превращается в список:

```
>>> lst = [2, 4, 6, 8]
>>> rng = range(2, 10, 2)
>>> rng
range(2, 10, 2)
>>> lst == rng
False
>>> lst == list(rng)
True
>>> list(rng)
[2, 4, 6, 8]
```

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. ЦИКЛ FOR

- Пример считалок с функцией range() смотрите в for_counter.py
- Пример последовательной обработки строки в цикле for смотрите в for_string.py
- Пример работы со списком в цикле for смотрите в for_list.py:
  - Почему не удались первые 2 попытки решить задачу?
  - Чем отличается перебор элементов от перебора индексов последовательности?
- Пример работы со списком, содержащим вложенные списки в циклах for смотрите в for_nested_list.py

ПРАКТИЧЕСКОЕ ЗАДАНИЕ №2. СРАВНЕНИЕ WHILE И FOR

- Нужно посчитать сумму нечетных чисел от 1729 до 13503 включая границы
- Сделайте это с помощью цикла while и сообщите сумму
- Решите ту же задачу с помощью цикла for и проверьте результат

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. ПРОГРАММА «УЖАСНАЯ ПОЭЗИЯ»

- Сам факт написания стихов компьютером дискутируется с 1950-ых годов:
  - https://www.ted.com/talks/oscar_schwartz_can_a_computer_write_poetry?language=ru
  - https://www.techinsider.ru/technologies/237094-robot-pisatel-chut-ne-vyigral-literaturnuyu-premiyu
- Современные подходы связаны с генеративным искусственным интеллектом:
  - ChatGPT (Open AI)
  - GigaChat (Сбер)
  - YandexGPT2 (Яндекс)
- **Наш пример на списки и цикл for для бредовой компьютерной поэзии – смотрите в awful_poetry.py**

СТРУКТУРА АНГЛИЙСКОГО ПРЕДЛОЖЕНИЯ

[particle] [adjective] noun verb [adverb]

- **particle** – частица, то есть артикль (a/the) или притяжательное местоимение (her, his, my, our, their, …)
- **adjective** – прилагательное
- **noun** – существительное
- **verb** – глагол в форме Past Simple
- **adverb** – наречие

ДОМАШНЕЕ ЗАДАНИЕ. «УЖАСНАЯ ПОЭЗИЯ» V.2.0

- Исключать повторение одних и тех же частиц, существительных, глаголов и наречий в рамках каждого шестистрочия (строфы)
- Случайные прилагательные вставлять перед существительным в 2/3 случаев, когда не используются наречия
- Задание со "звездочкой": оптимизируйте полученный код с применением собственной функции

СПАСИБО ЗА ВНИМАНИЕ !

ВОПРОСЫ ?

School of Computer Science