Понятие итератора

Основы Python

План урока

- Понятие итератора. Зачем они нужны?
- Iterable объекты. Методы iter(), next()
- Краткий обзор исключений в python. StopIteration
- Класс range
- Примеры задач

ИТЕРАТОР

(от англ. iterator - перечислитель)

Интерфейс, предоставляющий доступ к элементам коллекции или последовательности и навигацию по ним

Примеры коллекций

№	Фамилия
1	Петров
2	Сидоров
3	Иванов

Фамилия в таблице

Черные клетки шахматной доски

Обход коллекции абстрагированно

Этапы

- Выбираем коллекцию
- Идем к следующему элементу (первому)
- Идем к следующему элементу
- ...
- Идем к следующему элементу
- Конец коллекции

Обход коллекции в python

Этапы обхода collection

- it = iter(collection) # создаем итератор
- elem = next(it) # идем к следующему элементу (первому)
- elem = next(it) # идем к следующему элементу
- ...
- elem = next(it) # идем к следующему элементу
- Exception: StopIteration # исключение

А что делать с этим?

ИСКЛЮЧЕНИЕ

(исключительная ситуация)

Ошибка во время выполнения программы, которая может быть обработана без аварийного завершения программы

Обработка исключений

Конструкция блока try .. except

ooo | код

```
try:
    <code block>
except <Exception class 1>:
    <code block 1>
except <Exception class 2>:
    <code block 2>
finally:
    <code block finally>
```

Код в котором потенциально возможна ошибка

Обработка исключений

Конструкция блока try .. except

ooo | код

```
try:
    <code block>
except <Exception class 1>:
    <code block 1>
except <Exception class 2>:
    <code block 2>
finally:
    <code block finally>
```

Перехватчики ошибок

Обработка исключений

Конструкция блока try .. except

○○○	код

```
try:
    <code block>
except <Exception class 1>:
    <code block 1>
except <Exception class 2>:
    <code block 2>
finally:
    <code block finally>
```

Класс ошибки

Обработка исключений

Конструкция блока try .. except

ooo | код

```
try:
    <code block>
except <Exception class 1>:
    <code block 1>
except <Exception class 2>:
    <code block 2>
finally:
    <code block finally>
```

Блок кода выполняемый
в случае обнаружения ошибки

Обработка исключений

Конструкция блока try .. except

○○○	код

```
try:
    <code block>
except <Exception class 1>:
    <code block 1>
except <Exception class 2>:
    <code block 2>
finally:
    <code block finally>
```

Блок **try** выполняется, если нет ошибок

Обработка исключений

Конструкция блока try .. except

ooo | код

```
try:
    <code block>
except <Exception class 1>:
    <code block 1>
except <Exception class 2>:
    <code block 2>
finally:
    <code block finally>
```

Выполняется всегда

Обработка исключений

Пример. Деление на ноль

ooo | код

```
try:
    x = 5 / 0
    print(x)
except ZeroDivisionError:
    print("zero division")
```

Если 0 заменить на 1, например, то блок кода **try** выполнится успешно и исключение сгенерировано не будет

Функция range

Возвращает объект класса **range**

Объект класса **range**
это арифметическая прогрессия

Функция range

Возвращает объект класса **range**

Объект класса **range**
это арифметическая прогрессия

○○○	код

```
range(start, end, step)
```

Первый элемент последовательности

Функция range

Возвращает объект класса **range**

Объект класса **range**
это арифметическая прогрессия

○○○	код

```
range(start, end, step)
```

Шаг
прогрессии

Функция range

Возвращает объект класса **range**

Объект класса **range**
это арифметическая прогрессия

○○○	КОД

```
range(start, end, step)
```

Если **step** больше нуля,
то последовательность состоит из всех элементов от **start** до **end** с шагом **step**

End в последовательность не входит

Функция range

Возвращает объект класса **range**

Объект класса **range**
это арифметическая прогрессия

○○○	КОД

```
range(start, end, step)
```

Если **step** меньше нуля, то все элементы последовательности строго больше **end**

End в последовательность не входит

Функция range

ooo	код

```
range(start, end) # step = 1
```

ooo	код

```
range(end) # start = 0, step = 1
```

ooo	код

```
it = iter(range(2, 11, 3))
```

start

end

step

Функция range

ooo | код

```
it = iter(range(2, 11, 3))
print(next(it))
```

Печатаем первый элемент последовательности

ooo | вывод

```
2
```

Функция range

ooo | код

```
it = iter(range(2, 11, 3))
print(next(it))
print(next(it))
```

Печатаем второй элемент последовательности

ooo | вывод

```
2
5
```

Функция range

ooo | код

```
it = iter(range(2, 11, 3))
print(next(it))
print(next(it))
print(next(it))
```

Печатаем третий элемент последовательности

ooo | вывод

```
2
5
8
```

Функция range

ooo | код

```
it = iter(range(2, 11, 3))
print(next(it))
print(next(it))
print(next(it))
print(next(it))
```

Получаем исключение **StopIteration**

ooo | вывод

```
2
5
8
```

Функция range

ooo | код

```
it = iter(range(2, 11, 3))
while True:
    try:
        print(next(it))
    except StopIteration:
        break
```

Блок **try** перехватывает исключение **StopIteration**

ooo | вывод

```
2
5
8
```