Функциональное программирование

Берленко Т.А. СПбГЭТУ “ЛЭТИ”, ФКТИ, МОЭВМ

Повторение: переопределение из ООП

- Представьте, что вы пишите свой собственный класс для определенной задачи (не важно, какой именно).
- Что надо сделать, чтобы объекты этого класса *obj* можно было бы передавать в функцию *print(obj)*?
- Что надо сделать, чтобы можно было использовать функцию *len(obj)* с объектами вашего класса?

Повторение: полиморфизм в исключениях

```
L_base = BaseException.__subclasses__()  # получили всех наследников класса BaseException
print(L_base)
print(len(L_base))
L_obj = object.__subclasses__()  # получили всех наследников класса object
print(L_obj)
print(len(L_obj))

for e in L_base:
    print(e)  # вне зависимости от особенностей класса -- полиморфзм -- выводим на экран
    print(e.__subclasses__())
```

Дополнение про исключения: иерархия классов

Что будет выведено на экран?:

```
try:
    c = 5 / 0
except Exception:
    print('Exception occurred')
except ArithmeticError:
    print('ArithmeticError occurred')
except ZeroDivisionError:
    print('ZeroDivisionError occurred')
```

Функциональное программирование

- Функциональное программирование — раздел дискретной математики и парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в математическом понимании последних (в отличие от функций как подпрограмм в процедурном программировании).
- Относится к декларативной парадигме.
- Не предполагается явное хранение состояния программы.

Функции в функциональном программировании

- **Функция высшего порядка**

  Функция, которая может принимать на вход и/или возвращать другую функцию.

- **Чистая функция**

  Функция зависит только от своих параметров и не взаимодействует с внешними данными. Это значит, что для одних и тех же данных гарантировано получится один и тот же результат (также говорят, что функция детерминирована и не имеет *побочных эффектов*).

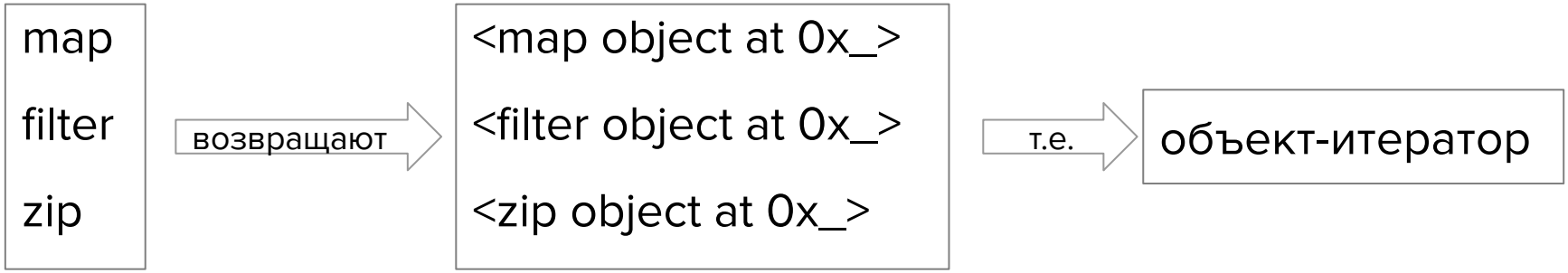
Функциональное программирование

- Данные неизменяемые
- Программа - совокупность чистых функций
- Отсутствие циклов
- Использование функций высшего порядка
- Функция может быть сохранена в переменную
- Функция не зависит от имени, по которому мы к ней обращаемся.

Примеры ЯП

- Лисп
- F#
- Haskell

Рассматриваемые функции в Python



- Что такое объект-итератор (или просто -- итератор)?
- Что такое итерируемый объект?

Итератор и итерируемый объект

Свойство	Итерируемый объект (**iterable** или iterable object)	Итератор (**iterator** или iterator object)
1) функция iter()	возвращает новый итератор	возвращает текущий итератор
2) функция next()	не работает!	переход к новому элементу
3) обход в цикле	элементы доступны вне цикла	элементы извлекаются и больше не доступны
4) взаимный переход	можно получить итератор	можно создать итерируемый объект

Функция map()

map(<function>, <iterable_1> [, <iterable_2>, ... ,<iterable_N-1>])

- применяет функцию function к элементам итерируемого объекта iterable
- формирует итератор из измененных элементов объекта iterable
- function должна возвращать значение
- количество объектов iterable равно количеству аргументов в функции function

Функция filter()

filter(<function>, <iterable>)

- применяет функцию function к элементам итерируемого объекта iterable
- формирует итератор из тех элементов объекта iterable, для которых function вернула True
- function должна результат, приводимый к True и False

Функция zip()

zip(*iterables)

- формирует итератор из пар (кортежей) нулевых, первых и т.д. элементов объектов iterables
- полезно в циклах, когда есть необходимо итерироваться по нескольким элементам сразу
- удобна для создания словарей

lambda-выражения

lambda argument1, argument2,..., argumentN : expression

- анонимные (т.е. не имеющие имени) функции, описываемые сразу в том месте, где эту функцию необходимо сразу вызвать
- argument1, argument2,..., argumentN - аргументы (как входные аргументы в случае определения функции через def)
- expression - выражение, в котором участвуют аргументы

Большой пример

ООП + функциональное программирование

- инкапсуляция, наследование, полиморфизм
- исключения
- функциональное (lambda, map, filter и пр.)

Источники и очень полезные ссылки

- Иерархия исключений: https://pythonworld.ru/tipy-dannyx-v-python/isklyucheniya-v-python-konstrukciya-try-except-dlya-obrabotki-isklyuchenij.html
- Функциональное программирование: https://ru.wikipedia.org/wiki/Функциональное_программирование
- Функция filter: https://docs.python.org/3/library/functions.html#filter
- Функция map: https://docs.python.org/3/library/functions.html#map
- Функция zip: https://docs.python.org/3/library/functions.html#zip
- Функция next(): https://docs.python.org/3/library/functions.html#next
- Функция iter(): https://docs.python.org/3/library/functions.html#iter
- Ключевое слово lambda: https://docs.python.org/3/reference/expressions.html#lambda
- lambda-выражения: https://docs.python.org/3/tutorial/controlflow.html#lambda-expressions