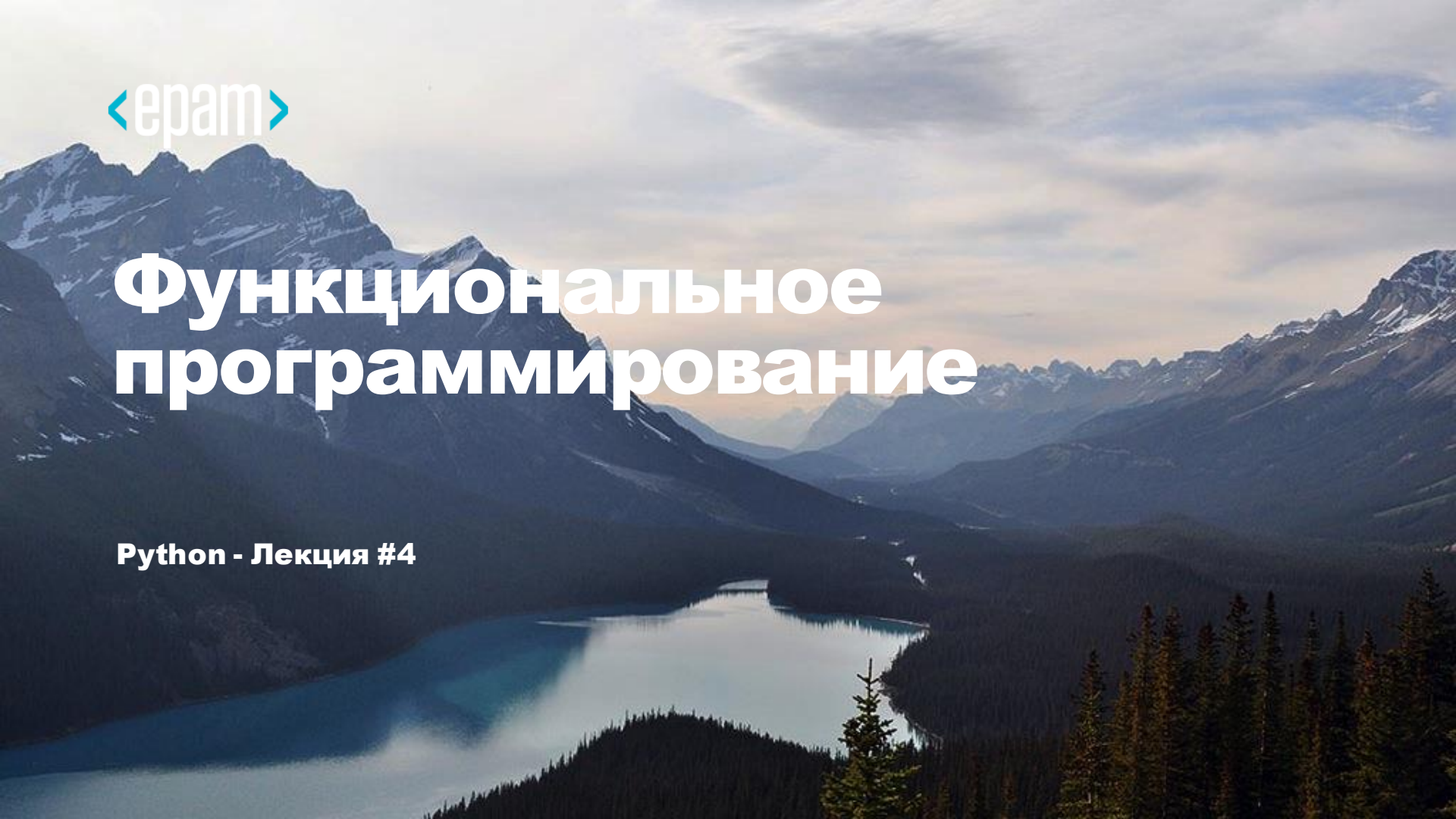

Функциональное программирование

Python - Лекция #4

Lambda

```
def func(x, y):
    return (x**2 + y**2) ** 0.5

func = lambda x, y: (x**2 + y**2) ** 0.5

(lambda x, y: (x**2 + y**2) ** 0.5)(10, 2)
```

Объекты первого класса (употребляются на равне с другими конструкциями)

Функциональное программирование

- Все есть функция
- Не храним состояние
- Лямбда исчисление
  - Аппликация
  - Абстракция
  - α-эквивалентность
  - β-редукция
  - η-преобразование
  - Каррирование
  - И другие баззворды

Концепции

- Функции высших порядков
- Чистые функции
  - Мемоизированные функции
  - Если результат не используется - можно удалить
  - Порядок вызовов не важен
- Рекурсии
- Любой цикл - рекурсия
- Нестрогие вычисления **(НЕ РАБОТАЕТ В Python)**

```
print(len([2+1, 3*2, 1/0, 5-4]))
```

Пример функционального программирования

```
# императивный стиль
target = []  # создать пустой список
for item in source_list:  # для каждого элемента исходного списка
    trans1 = G(item)  # применить функцию G()
    trans2 = F(trans1)  # применить функцию F()
    target.append(trans2)  # добавить преобразованный элемент в список


# функциональный стиль
compose2 = lambda A, B: lambda x: B(A(x))
target = map(compose2(G, F), source_list)
```

Функциональное программирование

Преимущества

- Повышение надёжности кода – код четко структурирован и без side-эффектов
- Удобство организации модульного тестирования – функции без side-эффектов
- Возможности оптимизации при компиляции
- Возможности параллелизма – за счет отсутствия side-эффектов

Недостатки

- Высокие требования к сборщику мусора
- Проблемы с операциями ввода-вывода (нестрогая модель вычисления)
- Порог входа

Списковые включения (абстракции) / list comprehensions

- Способ компактного описания списков
- `[n for n in range(1, 10000)]`
- `[n * 2 for n in range(1, 10000)]`
- `[(n * (2 + n)) ** 4 for n in range(1, 10000)]`
- `[n * 2 for n in range(1, 10000) if n % 2 == 0]`

- Может создавать dict, set, list, tuple
- **Dict** – `{n: 2 for n in range(1, 3)}`
- **Dict** – `{n + 1: n ** 2 for n in range(1, 3)}`
- **Set** – `{n ** 2 for n in range(1, 3)}`
- **List** – `[n for n in range(1, 10000)]`
- **Tuple** – `tuple(n for n in range(1, 10000))`
- **Generator** – `(n for n in range(1, 10000))`
- **Error** – `n for n in range(1, 10000)`

Базовые

Функция	Пример
filter	Фильтрация
map	Применение функции
functools.reduce	Reduce
zip	Сборка списка из элементов подсписков
enumerate	Добавление индекса
sorted	Сортировка
any	Проверка на корректность как минимум одного значения
all	Проверка на корректность всех значений

operator

Тип	Пример
Математич. операции	`sub, add, truediv`
Сравнение	`lt, le, eq, truth`
Доступ	`attrgetter, methodcaller, itemgetter`

functools

Функция	Пример
partial	Каррирование
partialmethod	То же самое, но для классов

itertools

Функция	Пример
cycle	То же самое, но для классов
repeat	Повторение элемента n раз (по умолчанию – inf)
islice	Создать срез итерируемого объекта
chain	Конкатенация итераторов

Task #1

```
names = ['Alexey', 'Ivan', 'Petr']

for i in range(len(names)):
    names[i] = hash(names[i])

print(names)
```

Task #2

```
sentences = ['test string',
             'with two test words: test and test',
             'and some without ** string']

count = 0
for sentence in sentences:
    count += sentence.count('test')
```

Task #3

У вас есть данные формата
[{name: ‘Alexey’, rate: 2, course: ‘Python’}, …]

Выведите топ студентов по каждому из предметов

HW #1

Изучить функции описанные в этой презентации

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