Урок №5

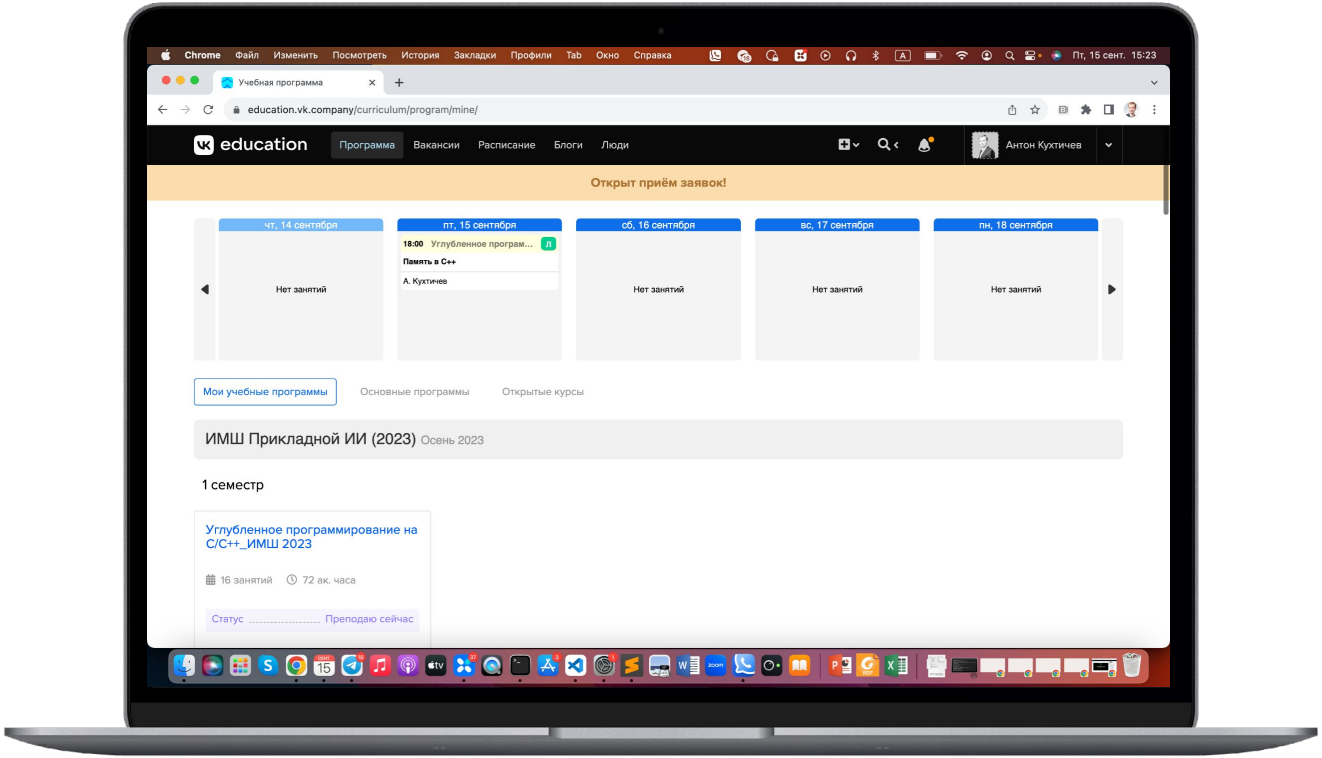
Стандартная библиотека

26 октября 2023 года

Антон Кухтичев

Напоминание отметиться на портале

и оставить отзыв
после лекции



Содержание занятия

- Квиз
- Числа, строки
- collections
- functools
- itertools
- Enum, dataclass

Квиз

Числа

float

```
>>> float("-inf"), float("inf"), float("nan")
(-inf, inf, nan)
>>> 0.1 + 0.2 == 0.3
False
>>> 0.1 + 0.2 <= 0.3
False
>>> 0.1 + 0.2
0.30000000000000004
>>> (0.1).as_integer_ratio()
(3602879701896397, 36028797018963968)
>>> format(0.1, ".25f")
'0.1000000000000000055511151'
>>> math.isclose(0.1 + 0.2, 0.3)
True
```

Decimal, Fraction

```
>>> from decimal import Decimal
>>> Decimal("0.1") + Decimal("0.2") == Decimal("0.3")
True
>>> Decimal(1) / Decimal(7)
Decimal('0.14285714285714285714285714285714')

>>> from fractions import Fraction
>>> Fraction(1, 10)
Fraction(1, 10)
>>> Fraction(1, 10) + Fraction(2, 10) == Fraction(3, 10)
True
```

Строки

str

- isalpha()
- isascii()
- isidentifier()
- isalnum()
- isdecimal()
- isdigit()
- isnumeric()

```
>>> s = "1¹²²³⁴⁴"
>>> s.isalnum()  # ???
>>> s.isdigit()  # ???
>>> s.isnumeric()  # ???
>>> s.isdecimal()  # ???
```

str

- startswith(prefix[, start[, end]])
- endswith(suffix[, start[, end]])
- capitalize()
- upper()
- isupper()
- lower()
- islower()
- title()
- istitle()
- swapcase()
- isprintable()
- isspace()

str

- count(sub[, start[, end]])
- find(sub[, start[, end]]), rfind(sub[, start[, end]])
- index(sub[, start[, end]]), rindex(sub[, start[, end]])
- replace(old, new[, count])
- center(width[, fillchar])
- encode(encoding='utf-8', errors='strict')
- expandtabs(tabsize=8)
- format(*args, **kwargs)
- ljust(width[, fillchar]) ,rjust(width[, fillchar])
- lstrip([chars]), strip([chars]), rstrip([chars])

str

- split(sep=None, maxsplit=-1), rsplit(sep=None, maxsplit=-1)
- splitlines(keepends=False)
- partition(sep), rpartition(sep)
- join(iterable)
- zfill(width)
- removeprefix(prefix, /)
- removesuffix(suffix, /)

collections

Специализированные контейнерные типы данных, предоставляющие альтернативы для встроенных `dict`, `list`, `set` и `tuple`

collections.namedtuple

namedtuple(typename, field_names, *, rename=False, defaults=None, module=None)

```
>>> Point = collections.namedtuple("Point", ["x", "y"])
>>> p = Point(11, y=22)  # p = (11, 22)
>>> p[0] + p[1]
33
>>> x, y = p
>>> x, y
(11, 22)
>>> p.x + p.y
33
>>> p._asdict()  # {'x': 1, 'y': 4}
```

collections.defaultdict

collections.defaultdict([default_factory[, ...]])

Словарь, у которого по умолчанию всегда вызывается функция **default_factory**.

```
>>> import collections
>>> defdict = collections.defaultdict(list)
>>> print(defdict)
defaultdict(<class 'list'>, {})
>>> for i in range(5):
...     defdict[i].append(i)
>>> print(defdict)
defaultdict(<class 'list'>, {0: [0], 1: [1], 2: [2], 3: [3], 4: [4]})
```

collections.OrderedDict

collections.OrderedDict([items])

Словарь, который помнит порядок, в котором ему были даны ключи.

```
>>> import collections
>>> d = collections.OrderedDict(
...     [("a", "A"), ("b", "B", ("c", "C")]
... )
>>> for k, v in d.items():
...     print(k, v)
'a', 'A'
'b', 'B'
'c', 'C'
>>> d.move_to_end("b")
```

collections.Counter

collections.Counter([iterable-or-mapping])

Словарь для подсчёта хешируемых объектов.

- elements()
- most_common([n])
- subtract([iterable-or-mapping])
- update([iterable-or-mapping])

```
>>> words = re.findall(r"\w+", open("hamlet.txt").read().lower())
>>> Counter(words).most_common(5)
[('the', 1143), ('and', 966), ('to', 762), ('of', 669), ('i', 631)]
```

collections.deque

`collections.deque([iterable[, maxlen]])`

Двусторонняя очередь из итерируемого объекта с максимальной длиной `maxlen`.

Добавление и удаление элементов в начало или конец выполняется за константное время.

- `append(x)`
- `appendleft(x)`
- `extend(iterable)`
- `extendleft(iterable)`
- `insert(i, x)`
- `pop()/popleft()`
- `remove(value)`

```
>>> d = deque("ghi")
>>> d.append("j")
>>> d.appendleft("f")
>>> d
deque(['f', 'g', 'h', 'i', 'j'])
>>> d.pop()
'j'
>>> d.popleft()
'f'
>>> d
deque(['g', 'h', 'i'])
```

functools

Для функций высшего порядка

functools

- cache(user_function)
- cached_property(func)
- lru_cache(user_function)
- lru_cache(maxsize=128, typed=False)

```
@functools.cache
def factorial(n):
    if n <= 1:
        return 1
    return n * factorial(n - 1)
```

singledispatch, singledispatchmethod(func)

```
@functools.singledispatch
def fun(arg, prefix="Hello"):
    print(f"{prefix} any type {arg}")

@fun.register(str)
def _(arg, prefix="Hi"):
    print(f"{prefix} str {arg}")

@fun.register
def _(arg: int, prefix="Hello"):
    print(f"{prefix} int {arg}")

>>> fun(123)  # Hello int 123
>>> fun.registry.keys()
# dict_keys([<class 'object'>, <class 'str'>, <class 'int'>])
```

functools

- partial(func, /, *args, **keywords)
- partialmethod(func, /, *args, **keywords)
- total_ordering
- reduce(function, iterable[, initializer])
- wraps(wrapped, assigned=WRAPPER_ASSIGNMENTS, updated=WRAPPER_UPDATES)
- update_wrapper(wrapper, wrapped, assigned=WA, updated=WU)

```
>>> basetwo = partial(int, base=2)
>>> basetwo("10010")
18
```

itertools

Можно бесконечно смотреть на
бесконечный цикл

Itertools: бесконечные итераторы

- `count(start=0, step=1)`
- `cycle(iterable)`
- `repeat(object[, times])`

Itertools

- accumulate(iterable[, func, *, initial=None])
- chain(*iterables)
- compress(data, selectors)
- dropwhile(predicate, iterable)
- takewhile(predicate, iterable)
- filterfalse(predicate, iterable)
- groupby(iterable, key=None)
- islice(iterable, start, stop[, step])
- pairwise(iterable)
- starmap(function, iterable)
- tee(iterable, n=2)
- zip_longest(*iterables, fillvalue=None)

Itertools: комбинаторика

- `product(*iterables, repeat=1)`
- `permutations(iterable, r=None)`
- `combinations(iterable, r)`
- `combinations_with_replacement(iterable, r)`

Разное

Enum

```python
from enum import Enum
class HttpStatus(enum.Enum):
    OK = 200
    NOT_FOUND = 404
    SERVER_ERROR = 500

print(HttpStatus.OK.value, HttpStatus.OK.name)  # (200, 'OK')

status = HttpStatus(200)

if status is HttpStatus.OK:
    print("OK")
```

Dataclasses

```
import dataclasses

@dataclasses.dataclass
class Point:
    x: int
    y: int
    vector: list[int] = dataclasses.field(default_factory=list)

p = Point(10, 20)
print(p.x, p.y, p.vector)  # (10, 20, [])

@dataclasses.dataclass(slots=True)
class PointSlots:
    x: int
    y: int

p = PointSlots(10, 20)
print(p)  # PointSlots(x=10, y=20)
```

Path

```
from pathlib import Path
p = Path(".")
pdf_path = p / "storage" / "slides.pdf"
abs_path = "/usr" / p / "storage" / "slides.pdf"

p.is_dir(), p.is_file()
p.is_absolute()
p.exists()
p.glob(pat)
p.open(), p.read_text(), p.write_text()
p.unlink()
```

heapq

```
heappush(heap, item)

heappop(heap)

heappushpop(heap, item)

heapify(x)

heapreplace(heap, item)

merge(*iterables, key=None, reverse=False)

nlargest(n, iterable, key=None)

nsmallest(n, iterable, key=None)
```

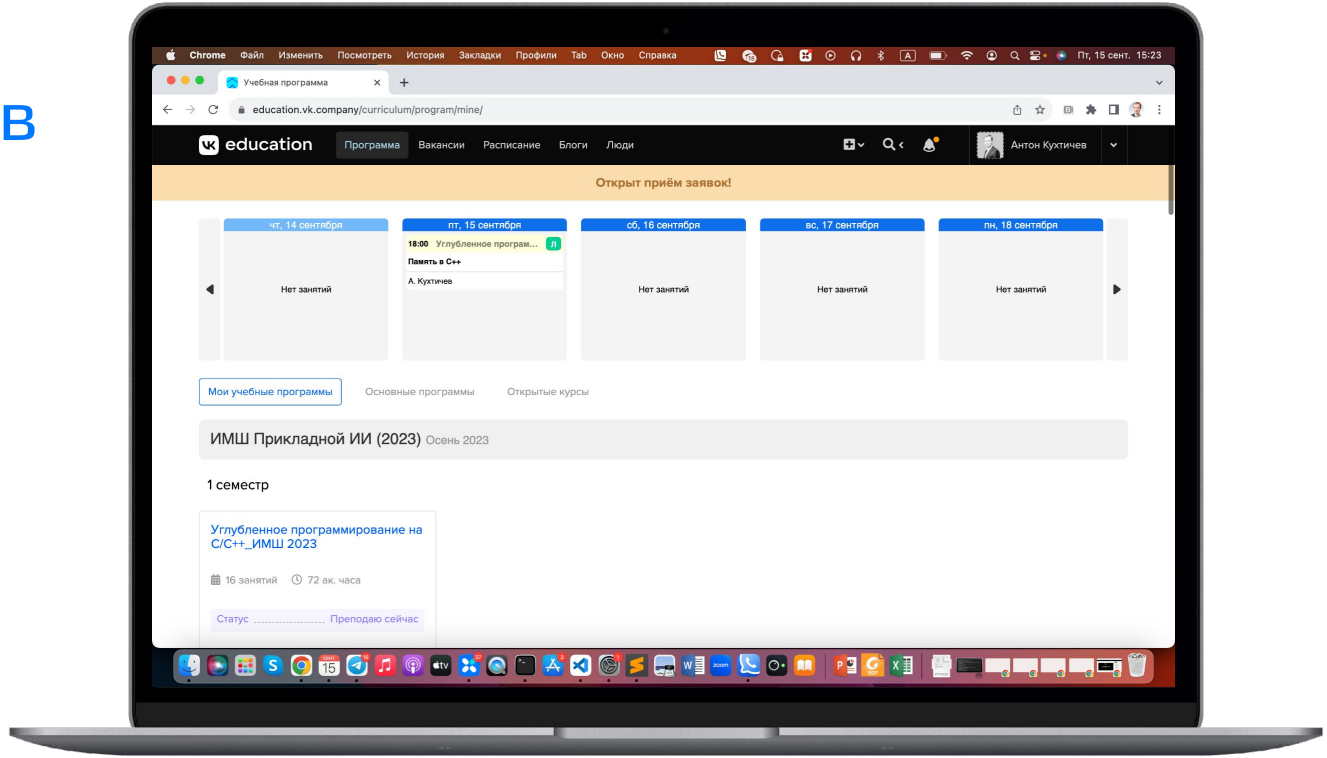
Домашнее задание

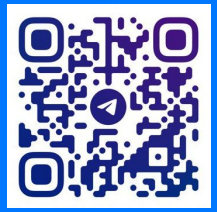
Домашнее задание #5

1. LRU cache без OrderedDict
2. Тесты
3. flake8 + pylint перед сдачей

Напоминание оставить отзыв

Это правда важно





Спасибо
за внимание!