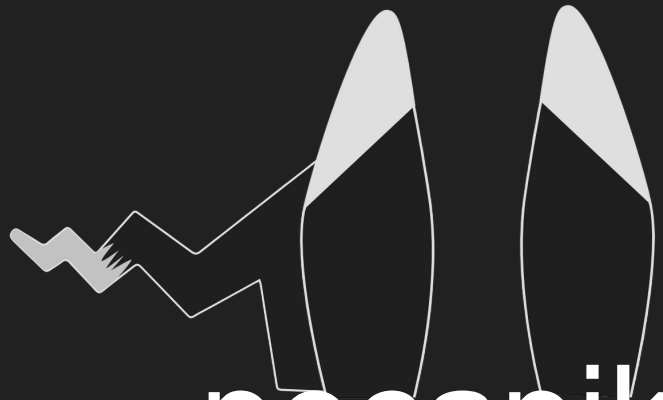
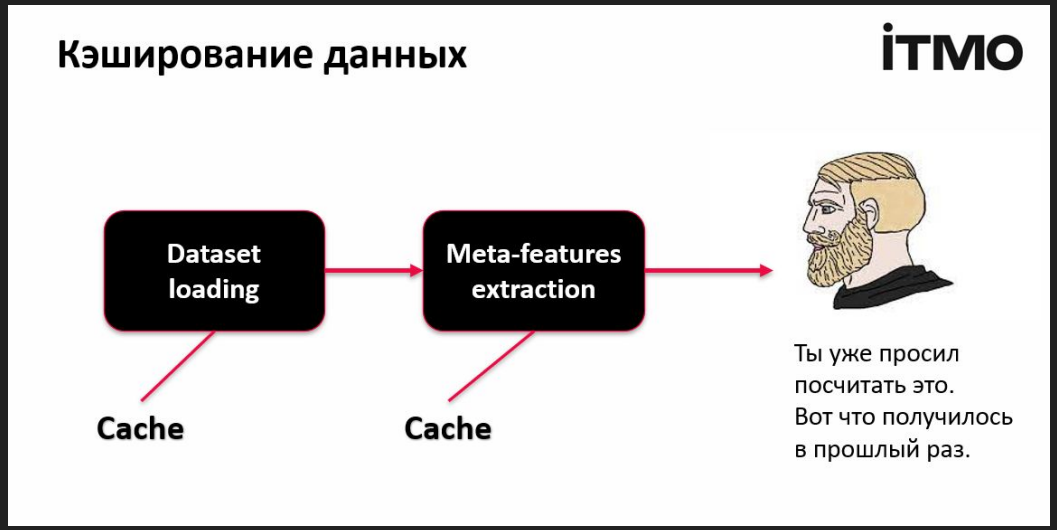


pecapiku

— persistent cache pickling utility.

С чего всё начиналось…



Слайд из презентации GAMLET - фреймворка для мета-обучения

```python
class CacheType(Enum):
    file = 'file'
    directory = 'directory'


@dataclass
class CacheProperties:
    type: Optional[CacheType] = None
    dir: Optional[Path] = None
    path_template: Optional[PathType] = None
```

Фрагмент исходного кода GAMLET

Готовые решения

Преимущества	functools.lru_cache	percache
Сохраняет кэш при перезапуске	-	+
Даёт доступ к файлу кэша	-	+
Работает с не хэшируемыми аргументами	-	+
Позволяет самостоятельно выбрать ключ кэширования, не передавая его в функцию	-	-
Позволяет задать уровень доступа к кэшу	-	-

Что получилось

Преимущества	functools.lru_cache	percache	pecapiku
Сохраняет кэш при перезапуске	-	+	+
Даёт доступ к файлу кэша	-	+	+
Работает с не хэшируемыми аргументами	-	+	+
Позволяет самостоятельно выбрать ключ кэширования, не передавая его в функцию	-	-	+
Позволяет задать уровень доступа к кэшу	-	-	+

Базовый синтаксис

```
@CacheDict()
def do_heavy_thing(time_: float):
    time.sleep(time_)
    return time_
```

Указываем файл для кэширования

```
@CacheDict(
    file_path='heavy_thing_cache.pkl',
)
def do_heavy_thing(time_: float):
    time.sleep(time_)
    return time_
```

Указываем свой ключ для кэширования

```
@CacheDict(
    inner_key='time_ + 1', # аргумент - ключ в кэше
)
def do_heavy_thing(time_: float, foo):
    print(foo)
    time.sleep(time_)
    return time_
```

Настраиваем доступ к кэшу

```
@CacheDict(
    access='rew', # read, execute, write
)
def do_heavy_thing(time_: float):
    time.sleep(time_)
    return time_
```

Режим перерасчёта кэша

```
@CacheDict(
    access='ew', # запрет на чтение кэша
)
def do_heavy_thing(time_: float):
    time.sleep(time_)
    return time_
```

Режим “только кэш”

```
@CacheDict(
    access='r', # запрет на исполнение кода
)
def do_heavy_thing(time_: float):
    time.sleep(time_)
    return time_
```

Отключаем доступ к кэшу

```
@CacheDict(
    access='e', # никакого кэша. только execute
)
def do_heavy_thing(time_: float):
    time.sleep(time_)
    return time_
```

Указываем внешний ключ для кэширования

```
for i in range(10):
    cached_do_heavy_thing = \
        CacheDict(
            do_heavy_thing,
            file_path='heavy_thing_cache.pkl',
            outer_key=i,
        )
    cached_do_heavy_thing(i**2)
```

Для чего НЕ предназначен pecapiku

- Для замены Data Version Control
- Ускорение кода с временем выполнения >> 1с

Для чего предназначен pecapiku

- Кэширование повторяющихся вычислений
- Экономия времени на отладку программы
- Ускорение повторного запуска долгих скриптов с минимальными изменениями
- Для чего хватит фантазии

Зоны роста

- Документация и релиз в PyPI
- Разные бэкенды для кэша
- Централизованное управление кэшем

Это конец. Или начало?