# Pydantic

from pydantic import BaseModel - базовая модель

Валидация данных — это процесс проверки данных различных типов по критериям корректности и полезности для конкретного применения.

Сериализация — это процесс преобразования объекта в поток байтов для сохранения или передачи в память, базу данных или файл.

class BigClass(BaseModel):

примеры базовых валидаций

fio: str = ‘Danila’

*всё, что после равно – это стандартное значение*

age: int

weight: float

accept: bool

более сложные:

from typing import Dict, List, Tuple, Set

bazam: Dict[]

list: List[]

tuple: Tuple[]

set: set[]

very\_hard: Dict[str, List[Tuple[int, bool]]]

Валидация электронной почты

from pydantic import EmailStr

email: EmailStr

Валидация даты:

from datetime import datetime

date\_event: datetime

*Literal нужен для списка того, что может использоваться в классе*

from typing import Literal

work: Literal['медицина', 'историк', 'астрофизик']

from typing import Optional

description: Optional[str] = None – Если нужно выбрать что-то, либо none

*Annotated вообще он передаёт мета инфомацию, но в FastAPi он проверяет то, что ему передаётся, первым элементом подаётся тип данных, вторым условие*

*Используется для условий*

from typing import Annotated

weight: Annotated[float, Gt(0)]

'Gt',

'Ge',

'Lt',

'Le',

'MinLen',

'MaxLen',

# Список всего, что можно использовать в Annotated

'BaseMetadata', 'GroupedMetadata','Gt', 'Ge', 'Lt', 'Le', 'Interval', 'MultipleOf', 'MinLen', 'MaxLen', 'Len', 'Timezone', 'Predicate', 'LowerCase', 'UpperCase', 'IsDigits', 'IsFinite', 'IsNotFinite', 'IsNan', 'IsNotNan', 'IsInfinite', 'IsNotInfinite', 'doc', 'DocInfo', '\_\_version\_\_',

# Как писать документацию

class Foo:

"""A class docstring.

Attributes:

bar: A description of bar. Defaults to "bar".

"""

bar: str = 'bar'

def bar(self, baz: int) -> str:

"""A function docstring.

Args:

baz: A description of `baz`.

Returns:

A description of the return value.

"""

return 'bar'

from pydantic import Field

Field используется по типу annotated, можно создавать условия, стандартные значения и т.д.

year\_zvaniya: int = Field(gt=0)

Field(pattern=r'^\d\*$') – для использования регулярных выражений

from decimal import Decimal

Field(max\_digits=5, decimal\_places=2)

Проверяет десятичное число max\_digits=максимальное количество знаков до и после запятой, decimal\_places=максимальное количество после запятой