**СТиВПП 2 - Валидный продукт**

**Исходный код программы и тестов**

**ssd\_product.py**

|  |
| --- |
| class SSDProduct:  ALLOWED\_INTERFACES = ['SATA', 'NVMe', 'SAS']   def \_\_init\_\_(self, data: dict):  *# Проверка обязательных полей и типов* if 'brand' not in data or not isinstance(data['brand'], str):  raise ValueError("Неверное значение для 'brand'")  self.brand = data['brand']   if 'capacity' not in data or not isinstance(data['capacity'], (int, float)):  raise ValueError("Неверное значение для 'capacity'")  self.capacity = data['capacity']   if 'nvme' not in data or not isinstance(data['nvme'], bool):  raise ValueError("Неверное значение для 'nvme'")  self.nvme = data['nvme']   if 'read\_speed' not in data or not isinstance(data['read\_speed'], (int, float)):  raise ValueError("Неверное значение для 'read\_speed'")  self.read\_speed = data['read\_speed']   *# write\_speed – необязательное поле* if 'write\_speed' in data:  if not isinstance(data['write\_speed'], (int, float)):  raise ValueError("Неверное значение для 'write\_speed'")  self.write\_speed = data['write\_speed']  else:  self.write\_speed = None   if 'interface' not in data or not isinstance(data['interface'], str) or data[  'interface'] not in self.ALLOWED\_INTERFACES:  raise ValueError("Неверное значение для 'interface'")  self.interface = data['interface']   @property  def performance\_index(self):  *"""Индекс производительности = (read\_speed + write\_speed) / capacity.  Если write\_speed отсутствует, используется только read\_speed."""* total\_speed = self.read\_speed + (self.write\_speed if self.write\_speed is not None else 0)  return total\_speed / self.capacity   def \_\_repr\_\_(self):  return (f"SSDProduct(brand={self.brand}, capacity={self.capacity}, nvme={self.nvme}, "  f"read\_speed={self.read\_speed}, write\_speed={self.write\_speed}, interface={self.interface})")  *# Демонстрация работы класса* if \_\_name\_\_ == '\_\_main\_\_':  api\_answer\_ssd = {  "brand": "Samsung",  "capacity": 1024,  "nvme": True,  "read\_speed": 3500,  "write\_speed": 3000,  "interface": "NVMe"  }  try:  product = SSDProduct(api\_answer\_ssd)  print("Продукт создан успешно:", product)  print("Индекс производительности:", product.performance\_index)  except ValueError as e:  print("Ошибка:", e) |

**test\_ssd\_product.py**

|  |
| --- |
| import unittest  from ssd\_product import SSDProduct  class TestSSDProduct(unittest.TestCase):  def test\_valid\_data(self):  data = {  "brand": "Kingston",  "capacity": 512,  "nvme": False,  "read\_speed": 550,  "interface": "SATA"  }  product = SSDProduct(data)  self.assertEqual(product.brand, "Kingston")  self.assertEqual(product.capacity, 512)  self.assertFalse(product.nvme)  self.assertEqual(product.read\_speed, 550)  self.assertIsNone(product.write\_speed)  self.assertEqual(product.interface, "SATA")   def test\_invalid\_brand(self):  data = {  "brand": 123, *# неверный тип* "capacity": 512,  "nvme": False,  "read\_speed": 550,  "interface": "SATA"  }  with self.assertRaises(ValueError):  SSDProduct(data)   def test\_invalid\_interface(self):  data = {  "brand": "Intel",  "capacity": 256,  "nvme": True,  "read\_speed": 2000,  "interface": "USB" *# недопустимое значение* }  with self.assertRaises(ValueError):  SSDProduct(data)  if \_\_name\_\_ == '\_\_main\_\_':  unittest.main() |

Товара «SSD-диск»:

* **brand** (строка, обязательное) — бренд SSD.
* **capacity** (число, обязательное) — объём в ГБ.
* **nvme** (логический тип, обязательное) — поддерживает ли NVMe.
* **read\_speed** (число, обязательное) — скорость чтения в МБ/с.
* **write\_speed** (число, НЕобязательное) — скорость записи в МБ/с.
* **interface** (выбор, обязательное) — один из: "SATA", "NVMe", "SAS".

**Проверяемые случаи**

* Корректные данные: тест test\_valid\_data проверяет создание объекта с минимальным набором обязательных полей.
* Неверный тип для brand: тест test\_invalid\_brand гарантирует, что при неправильном типе поля вызывается исключение.
* Неверное значение для interface: тест test\_invalid\_interface проверяет, что значение, не входящее в список разрешённых, приводит к ошибке.

**Отчёт по покрытию**

Тесты охватывают основные сценарии: корректное создание объекта и генерацию ошибок для неверных данных. Такой набор тестов обеспечивает базовую валидацию входных данных и корректную работу вычисляемого свойства.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.