Table of contents

1

OOP

-
-
-
-
-

1.1

OOP

- - Car

1.1.1 OOP

-
-
-
-

1.2

1.2.1

-
-
-

1.2.2

-
-
-

```python
#
class Car:
    def __init__(self, make, model):
        self.make = make
        self.model = model

    def start(self):
        return f"{self.make} {self.model}    "

#
car1 = Car("  ", "  ")
car2 = Car("  ", "   ")
```

1.3

```python

class Dog:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def bark(self):
        return f"{self.name}   "

    def info(self):
        return f"{self.name} {self.breed} "
```

```
#
my_dog = Dog(" ", "      ")
print(my_dog.bark())  #
print(my_dog.info())  #
```

1.4 __init__

- - - self

```
class Person:
    def __init__(self, name, age, email):
        self.name = name        #
        self.age = age          #
        self.email = email      #
        self.friends = []       #

#
person1 = Person(" ", 25, "taro@email.com")
person2 = Person(" ", 30, "hanako@email.com")

print(person1.name)  #
print(person2.age)   # 30
```

1.5

1.5.1

```
class BankAccount:
    def __init__(self, account_number, balance=0):
        self.account_number = account_number
        self.balance = balance
        self.transaction_history = []
```

1.5.2

```
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            self.transaction_history.append(f"{amount}   ")
            return True
        return False

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            self.transaction_history.append(f"{amount}   ")
            return True
        return False
```

1.6 BankAccount

```
#
account = BankAccount("123456789", 100000)

#
account.deposit(50000)
account.withdraw(20000)

#
print(f" : {account.account_number}")
print(f" : {account.balance} ")
print(" :", account.transaction_history)
```

 :

```
 : 123456789
 : 130000
 : ['50000   ', '20000   ']
```

1.7

```python
class Dog:
    species = "Canis lupus"  #
    total_dogs = 0           #

    def __init__(self, name, breed):
        self.name = name     #
        self.breed = breed   #
        Dog.total_dogs += 1  #

    @classmethod
    def get_total_dogs(cls):
        return cls.total_dogs

#
dog1 = Dog(" ", "     ")
dog2 = Dog("  ", "     ")

print(Dog.species)           # Canis lupus
print(Dog.get_total_dogs())  # 2
```

1.8

```python
class Counter:
    count = 0  #

    def __init__(self, name):
        self.name = name        #
        Counter.count += 1      #
        self.instance_count = 1  #

counter1 = Counter("1 ")
counter2 = Counter("2 ")

print(f"   : {Counter.count}")      # 2
print(f"  1  : {counter1.name}")     # 1
print(f"  2  : {counter2.name}")     # 2
```

1.9

```
class Book:
    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages

    def __str__(self):
        return f" {self.title}   : {self.author}"

    def __repr__(self):
        return f"Book('{self.title}', '{self.author}', {self.pages})"

book = Book("1984", "      ", 328)
print(str(book))   #  1984   :
print(repr(book))  # Book('1984', '      ', 328)
print(book)        # __str__
```

1.10

```
class Circle:
    def __init__(self, radius):
        self._radius = radius  # "    "

    @property
    def radius(self):
        return self._radius

    @radius.setter
    def radius(self, value):
        if value < 0:
            raise ValueError("       ")
        self._radius = value

    @property
    def area(self):
        return 3.14159 * self._radius ** 2

circle = Circle(5)
print(circle.area)     # 78.53975
circle.radius = 10     #
print(circle.area)     # 314.159
```

1.11

Python ” ”

```
class BankAccount:
    def __init__(self, balance):
        self.public_attr = "     "
        self._protected_attr = "       "
        self.__private_attr = "          "

    def get_private(self):
        return self.__private_attr

account = BankAccount(100000)
print(account.public_attr)      # OK
print(account._protected_attr)  #
# print(account.__private_attr) # AttributeError
print(account.get_private())    #
```

Python

1.12

1.12.1

```
class Person:
    population = 0

    def __init__(self, name):
        self.name = name
        Person.population += 1

    @classmethod
    def get_population(cls):
        return cls.population

    @classmethod
    def create_anonymous(cls):
        return cls(" ")

print(Person.get_population())  # 0
person1 = Person(" ")
anonymous = Person.create_anonymous()
print(Person.get_population())  # 2
```

1.12.2

self cls

```
class MathUtils:
    @staticmethod
    def add(a, b):
        return a + b

    @staticmethod
    def is_even(number):
        return number % 2 == 0

    @staticmethod
    def factorial(n):
        if n <= 1:
            return 1
        return n * MathUtils.factorial(n - 1)

#
print(MathUtils.add(5, 3))      # 8
print(MathUtils.is_even(4))     # True
print(MathUtils.factorial(5))   # 120
```

1.13

```python
class Student:
    total_students = 0

    def __init__(self, name, student_id, email):
        self.name = name
        self.student_id = student_id
        self.email = email
        self.grades = {}
        self.enrolled_courses = []
        Student.total_students += 1

    def enroll_course(self, course):
        if course not in self.enrolled_courses:
            self.enrolled_courses.append(course)
            self.grades[course] = []

    def add_grade(self, course, grade):
        if course in self.grades:
            self.grades[course].append(grade)

    def get_average(self, course):
        if course in self.grades and self.grades[course]:
            return sum(self.grades[course]) / len(self.grades[course])
        return 0

    def __str__(self):
        return f" : {self.name} (ID: {self.student_id})"
```

1.14 Student

```python
#
alice = Student("  ", "S001", "tanaka@school.edu")
bob = Student("  ", "S002", "sato@school.edu")

#
alice.enroll_course(" ")
alice.enroll_course(" ")
```

```
bob.enroll_course(" ")

#
alice.add_grade(" ", 85)
alice.add_grade(" ", 92)
alice.add_grade(" ", 78)

#
print(alice)                              #   :    (ID: S001)
print(f"   : {alice.get_average(' ')}")  # 88.5
print(f"  : {Student.total_students}")    # 2
```

1.14.1 1 Rectangle

```
class Rectangle:
    def __init__(self, width, height):
        #
        pass

    def area(self):
        #
        pass

    def perimeter(self):
        #
        pass

    def __str__(self):
        #
        pass
```

1.14.2 2

```
class LibraryBook:
    def __init__(self, title, author, isbn):
        #
        pass

    def check_out(self):
        #
        pass

    def return_book(self):
        #
        pass
```

1.14.1 Rectangle

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

    def perimeter(self):
        return 2 * (self.width + self.height)

    def __str__(self):
        return f"Rectangle({self.width}x{self.height})"
```

1.14.2

```
class LibraryBook:
    def __init__(self, title, author, isbn):
        self.title = title
        self.author = author
        self.isbn = isbn
        self.is_checked_out = False
```

```python
    def check_out(self):
        if not self.is_checked_out:
            self.is_checked_out = True
            return True
        return False

    def return_book(self):
        if self.is_checked_out:
            self.is_checked_out = False
            return True
        return False
```

1.15

1. : BankAccount BA
2. :
3. `__init__` :
4. : `__str__` `__repr__`
5. :
6. :
7. :

1.16

1.16.1 1. `self`

```python
class Counter:
    def __init__(self, start=0):
        count = start   #    self.count = start

    def increment(self):
        count += 1      #    self.count += 1
```

1.16.2 2.

```python
#
class Student:
    def __init__(self, name, courses=[]):
        self.courses = courses  #

#
class Student:
    def __init__(self, name, courses=None):
        self.courses = courses if courses is not None else []
```

1.17

1.17.1

-
-
-
-

1.17.2

- → namedtuple
- →
- →

1.18

-
-
- `__init__`
- `self`
-
-
-

1.19

1.
2. `__init__`
3.
4.
5. 3