Объектно-ориентированное программирование

Что это

Python имеет множество встроенных типов, например, int, str и так далее, которые мы можем использовать в программе. Но также Python позволяет определять собственные типы с помощью классов. Класс представляет некоторую сущность. Конкретным воплощением класса является объект.

Что это

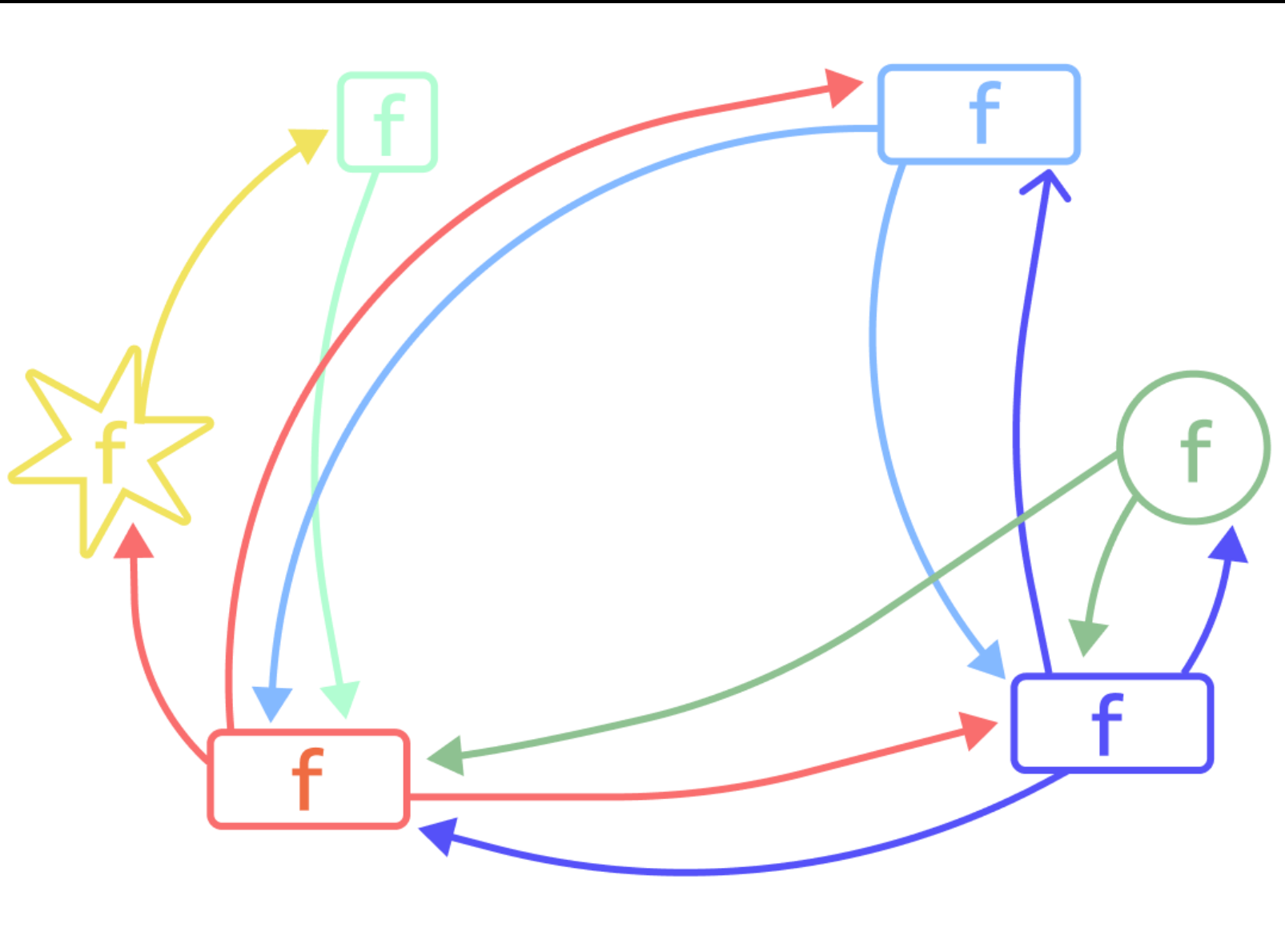
Объектно-ориентированное программирование (ООП) — это подход, при котором программа рассматривается как набор объектов, взаимодействующих друг с другом. У каждого есть свойства и поведение.

Что это

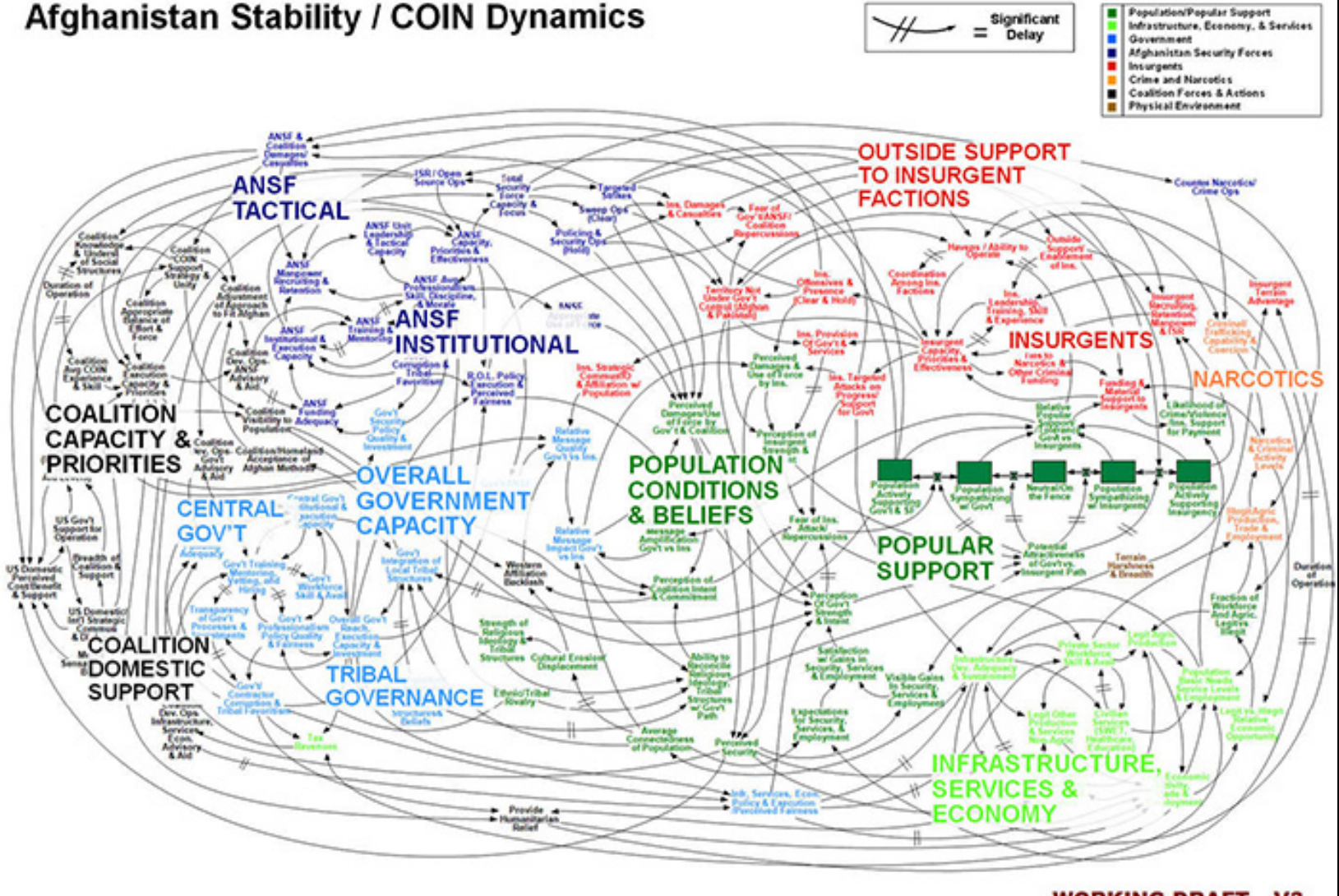
Людям проще воспринимать окружающий мир как объекты, которые поддаются определенной классификации (например, разделение на живую и неживую природу).

Что было раньше

До ООП в разработке использовался другой подход — процедурный. Программа представляется в нем как набор процедур и функций — подпрограмм, которые выполняют определенный блок кода с нужными входящими данными.


f
f
f
f
f
f

Afghanistan Stability / COIN Dynamics



Зачем нужно ООП

- структурировать информацию и не допускать путаницы;
- точно определять взаимодействие одних элементов с другими;
- повышать управляемость программы;
- быстрее масштабировать код под различные задачи;

Зачем нужно ООП

- лучше понимать написанное;
- эффективнее поддерживать готовые программы;
- внедрять изменения без необходимости переписывать весь код.

Где есть

Возможности ООП поддерживает большинство популярных языков программирования, включая C++, JavaScript, PHP, Python и другие.

Структура ООП

Чтобы сделать код проще, программу разбивают на независимые блоки — объекты. В реальной жизни это может быть стол, чашка, человек, книга, здание и многое другое. В программировании объекты — это структуры данных: пользователь, кнопка, сообщение.

Что такое класс

Класс — это «шаблон» для объекта, который описывает его свойства. Несколько похожих между собой объектов, например профили разных пользователей, будут иметь одинаковую структуру, а значит, принадлежать к одному классу. Каждый объект — это экземпляр какого-нибудь класса.

Аналогии

Понятие «программист» — это класс.

Конкретный разработчик по имени Иван — это объект, принадлежащий к классу «программист» (экземпляр класса).

Зарплата, рабочие обязанности, изученные технологии и должность в компании — это свойства, которые есть у всех объектов класса «программист» и могут различаться у разных объектов.

Атрибуты и методы

Объект — это набор переменных и функций, как в традиционном функциональном программировании. Переменные и функции и есть его свойства.

Атрибуты — это переменные, конкретные характеристики объекта, такие как цвет поля или имя пользователя.

Атрибуты и методы

Методы — это функции, которые описаны внутри объекта или класса. Они относятся к определенному объекту и позволяют взаимодействовать с ними или другими частями кода.

Класс:
программист

Объект:
разработчик Иван

Атрибуты:
зарплата, обязанности

Методы:
написание кода

Python

В языке Python класс определяется с помощью ключевого слова class:

```
class название_класса:
    атрибуты_класса
    методы_класса
```

Конструкторы

Для создания объекта класса используется конструктор.

```
class Person:
    # конструктор
    def __init__(self):
        print("Создание объекта Person")

tom = Person()      # Создание объекта Person
```

Атрибуты объекта

Атрибуты хранят состояние объекта. Для определения и установки атрибутов внутри класса можно применять слово self.

```
class Person:
    def __init__(self, name, age):
        self.name = name    # имя человека
        self.age = age      # возраст человека

tom = Person("Tom", 22)
```

Значения атрибутов

```
class Person:
    def __init__(self, name, age):
        self.name = name       # имя человека
        self.age = age         # возраст человека

tom = Person("Tom", 22)

tom.company = "Microsoft"
print(tom.company)  # Microsoft
```

Методы классов

Методы класса фактически представляют функции, которые определенны внутри класса и которые определяют его поведение.

```
class Person:      # определение класса Person
    def say_hello(self):
        print("Hello")

tom = Person()
tom.say_hello()    # Hello
```

Методы классов

Если метод должен принимать другие параметры, то они определяются после параметра self, и при вызове подобного метода для них необходимо передать значения:

```
class Person:      # определение класса Person
    def say(self, message):    # метод
        print(message)

tom = Person()
tom.say("Hello world")    # Hello world
```

```
class Person:
    def __init__(self, name, age):
        self.name = name       # имя человека
        self.age = age         # возраст человека

    def display_info(self):
        print(f"Name: {self.name}  Age: {self.age}")

tom = Person("Tom", 22)
tom.display_info()     # Name: Tom  Age: 22

bob = Person("Bob", 43)
bob.display_info()     # Name: Bob  Age: 43
```

Задание

- Создать класс «Кот» с атрибутами имя, возраст, окрас.
- Добавить метод выводящий «мяу» на консоль
- Добавить метод выводящий сообщение «{имя_кота мяукнул}»

Модуль OS и работа с файловой системой

Что это

Ряд возможностей по работе с каталогами и файлами предоставляет встроенный модуль os. Основные функции:

- mkdir(): создает новую папку
- rmdir(): удаляет папку
- rename(): переименовывает файл
- remove(): удаляет файл

Создание и удаление папки

Для создания папки применяется функция mkdir(), в которую передается путь к создаваемой папке:

```
import os
# путь относительно текущего скрипта
os.mkdir("hello")
# абсолютный путь
os.mkdir("c://somedir")
os.mkdir("c://somedir/hello")
```

Создание и удаление папки

Для удаления папки используется функция rmdir(), в которую передается путь к удаляемой папке:

```
import os
# путь относительно текущего скрипта
os.rmdir("hello")
# абсолютный путь
os.rmdir("c://somedir/hello")
```

Переименование файла

Для переименования вызывается функция rename(source, target), первый параметр которой - путь к исходному файлу, а второй - новое имя файла.

```
import os
os.rename("C://SomeDir/somefile.txt", "C://SomeDir/hello.txt")
```

Удаление файла

Для удаления вызывается функция remove(), в которую передается путь к файлу:

```
import os

os.remove("C://SomeDir/hello.txt")
```

Существование файла

Уже до открытия файла проверить, существует ли он или нет с помощью метода os.path.exists(path).

```
filename = input("Введите путь к файлу: ")
if os.path.exists(filename):
    print("Указанный файл существует")
else:
    print("Файл не существует")
```