NAJOT TA'LIM

Bootcamp Foundation

4-OY. PreFoundation

9

MAVZU

OOP(Object Oriented Programming). Class and objects.

Mallayev Oybek Usmankulovich

Najot ta'lim

Dars rejasi

OYD va Obyekt nima?

Class nima?

OYD tamoillari.

Obyektlar munosabatlari

XULOSA

Obyektga yo‘naltirilgan dasturlash

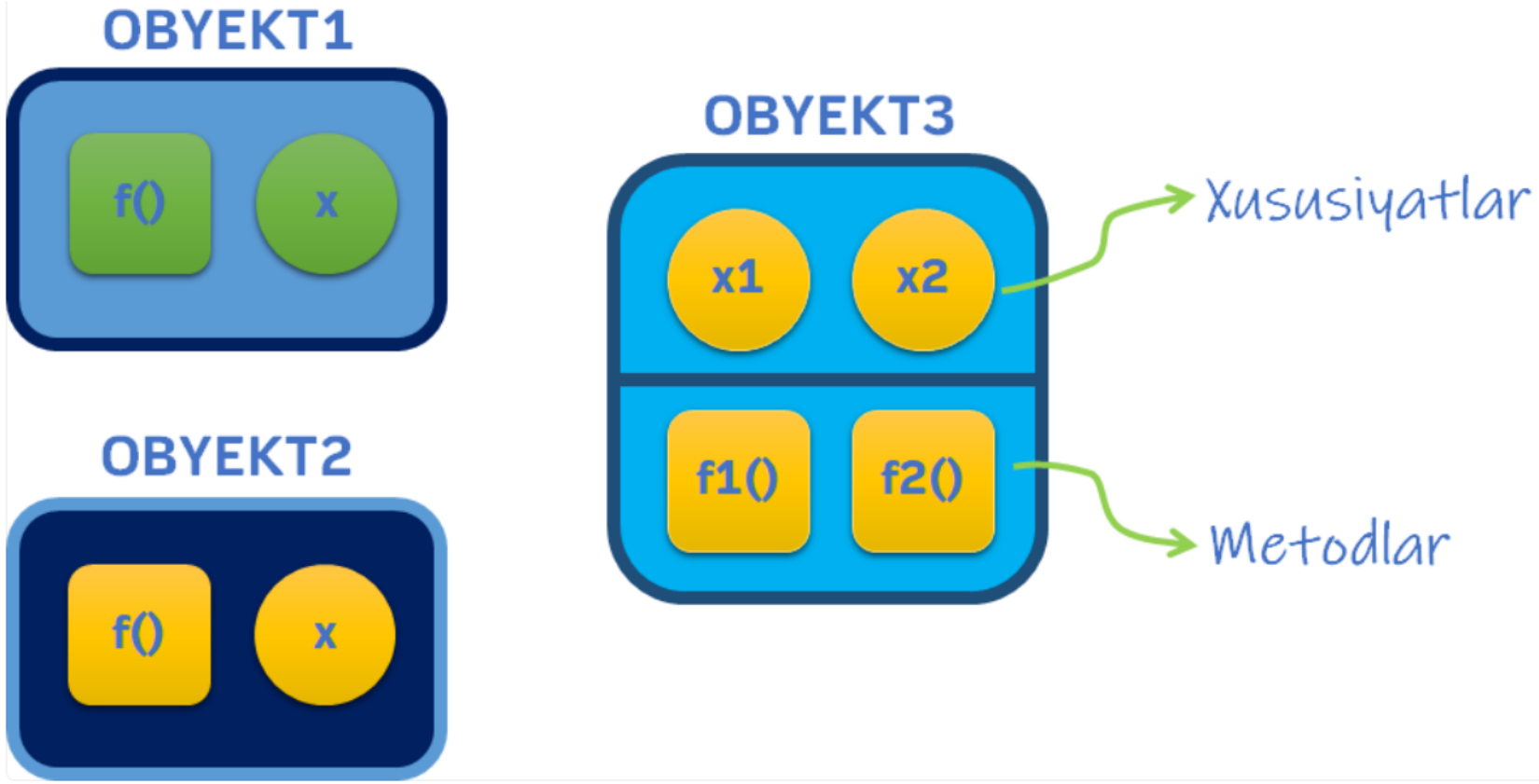
(OYD) – bu dasturlashga yangi bir yondashuvdir. Ma’lumotlarning turlari yildan yilga o’zgarib va ko’payib borayotgani va ularni boshqarishning yangi texnologiyalarini yaratish OYD ning asosiy maqsadi hisoblanadi.

Hozirda barcha zamonaviy dasturlash tillari OYD tamoyillari asosida ishlamoqda.

OYD asosida yangidan yangi obyektlar yaratiladi. Ushbu obyektlar hayotning muhim jabhalariga tadbiq etib borilada.

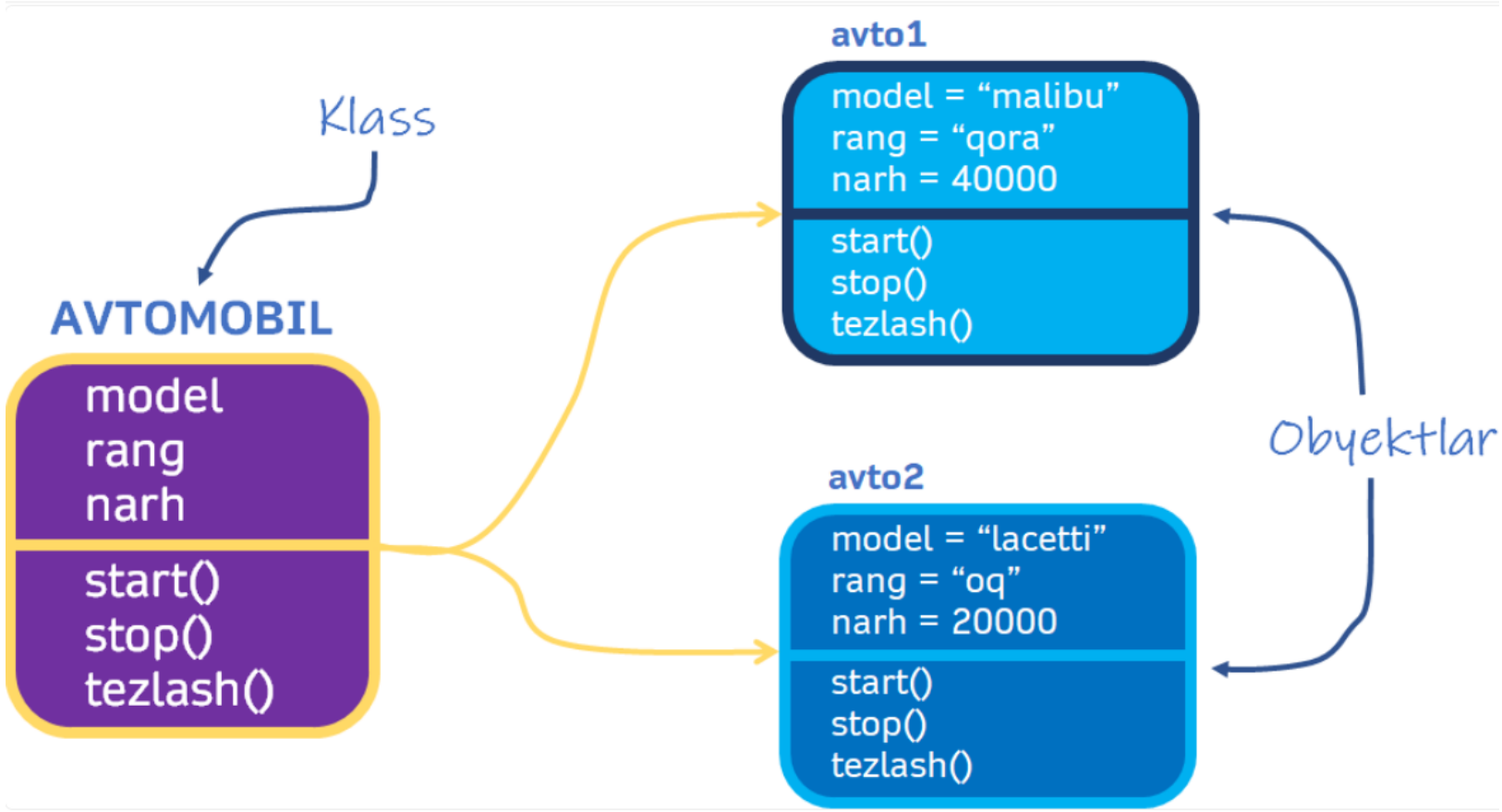
OBYEKT NIMA?

Object oriented dasturlashda o’zaro bo’gliq bo’lgan o’zgaruvchilar va funksiyalar bitta konteynerga jamlanadi va bunday konteynerlar obyekt deb ataladi. Bir obyektga tegishli o’zgaruvchilar uning xususiyatlari, unga tegishli funksiyalar esa metodlari deb ataladi.



KLASS NIMA?

Klass bu obyekt yaratish uchun **shablon** yoki **qolipdir**. Bitta klassdan biz istalgancha nusxa olishimiz va yangi obyektlar yaratishimiz mumkin. Demak obyekt bu biror klassning xususiy ko'rinishi. Odatda klasslarning nomi o'zgarmas, undan yaratilgan obyektlar esa istalgancha nomlanishi mumkin.



OOP TAMOYILLARI

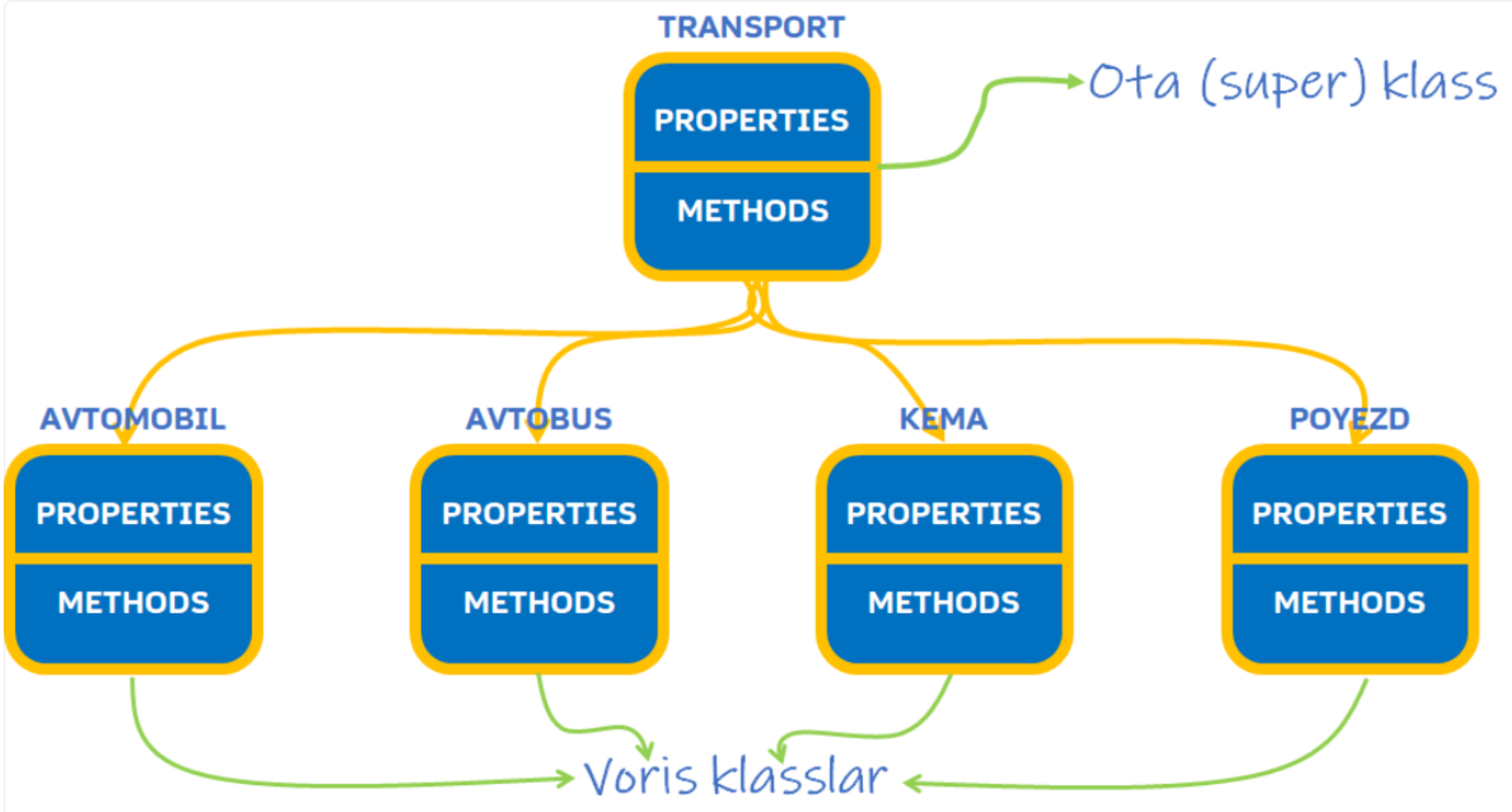
INKAPSULYATSIYA

Biz object oriented dasturlash haqida gapira turib, ma’lum bir obyektga tegishli bo’lgan xususiyatlar va metodlarni bitta konteynerga joylaymiz dedik. Bu jarayon inkapsulyatsiya (ya’ni kapsulaga solish) deb ataladi. Inkapsulyatsiya bizga klasslar yaratishga va keyinchalik bu klasslardan boshqa obyektlarni yaratishga yordam beradi.

ABSTRAKTSIYA

Abstraktsiya yordamida biz kodimizning ichki tuzilishini yashiramiz. Ya’ni, tashqaridan qaraganda obyektimiz 2 ta parameter va 2 ta metoddan iborat bo’lishi mumkin, lekin obyekt to’g’ri ishlashi uchun uning ichida o’nlab boshqa o’zgaruvchilar va funksiyalar yashirin bo’ladi. Klassdan foydalanishda esa uning ichki tuzilishi va qanday ishlashini bilish talab qilinmaydi. Bu o’zimizga ham boshqa dasturchilarga ham bu klassdan foydalanishda qulayliklar yaratadi.

VORISLIK


TRANSPORT
PROPERTIES
METHODS
Ota (super) klass
AVTOMOBIL
AVTOBUS
KEMA
POYEZD
PROPERTIES
METHODS
PROPERTIES
METHODS
PROPERTIES
METHODS
PROPERTIES
METHODS
Voris klasslar

POLIMORFIZM

Voris klass super klassdan o'zlashtirilgan metodning nomini saqlagan holda, uning ishlashini o'zgartirishiga **polimorfizm** deyiladi.

Keling bir misol ko'raylik. Biz kompyuter o'yini yaratish jarayonida o'yin Qahramon uchun super klass yaratamiz. Qahramon bir nechta xususiyatlarga va metodlarga ega. Jumladan `attack()` ya'ni xujum qilish metodi, qahramonni xujum qilishga undaydi. Endi biz bu superklassdan boshqa voris klasslarni yaratamiz.

PYTHONDAGI KLASSLAR

Klass tushunchasi siz uchun yangi bo'lishi mumkin, lekin biz shu vaqtgacha ulardan doimiy ravishda foydalanib keldik.

Keling `x` o'zgaruvchi yaratamiz, unga biror qiymat yuklaymiz va `type()` funksiyasi yordamida uning turini kuramiz:

```
x = 10
print(type(x))
```

Natija: `<class 'int'>`

```
matn = "salom"
print(type(matn))
```

Natija: `<class 'str'>`

METODLAR

Har bir obyekt uning ustida bajarish mumkin bo'lgan funksiyalar bilan keladi. Bu funksiyalar obyekt ichida yashirin bo'ladi, va biz ularga nuqta va funksiya nomi orqali murojat qilishimiz mumkin. Bunday funksiyalar shu klass (yoki obyektga) tegishli **metodlar** deyiladi.

Biz ba'zi metodlar bilan avvalgi darslarimizda tanishdik. Bir klassga tegishli metodlar, boshqa klassdagi obyketlar uchun mavjud bo'lmasligi tabiiy. Misol uchun matnlar uchun mavjud metodlarni, butun yoki o'nli sonlarga qo'llab bo'lmaydi.

```
matn = "salom"
print(matn.upper())
```

Natija: `SALOM`

```
son = 20
print(son.lower())
```

KLASS YARATISH

```
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self,ism,familiya,tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil
```

KLASSDAN OBYEKT YARATISH

```
talaba1 = Talaba("Alijon","Valiyev",2000)
```

OBYKETNING XUSUSIYATLARINI KO'RISH

Obyektning xususiyatlarini ko'rish uchun nuqta orqali murojat qilishimiz mumkin.

```
print(talaba1.ism)
```

Natija: `Alijon`

```
print(talaba1.familiya)
```

Natija: `Valiyev`

KLASSDAN BIR NECHTA OBYEKTLAR YARATISH

Yuqoridagi klassdan biz istalgancha obyektlar yaratishimiz mumkin:

```
talaba2 = Talaba("Olim","Olimov",1995)
talaba3 = Talaba("Husan","Akbarov",2004)
talaba4 = Talaba("Hasan","Akbarov",2004)
```

Bunda har bir obyekt o'zining alohida xususiyatlariga ega bo'ladi.

```
print(talaba2.ism)
print(talaba4.familiya)
```

Natija:

- `Olim`
- `Akbarov`

KLASSGA METODLAR QO'SHISH

```python
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self,ism,familiya,tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil

    def tanishtir(self):
        print(f"Ismim {self.ism} {self.familiya}. {self.tyil} yilda tu'gilganman")
```

OBYEKTNING METODLARIGA MUROJAAT

Obyekt ichidagi funksiyaga ya'ni obyektning metodiga murojat qilamiz:

```
talaba4 = Talaba("Hasan","Akbarov",2004)
talaba4.tanishtir()
```

Natija: `Ismim Hasan Akbarov. 2004 yilda tu'gilganman`

ARGUMENT QABUL QILUVCHI METODLAR

```python
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self,ism,familiya,tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil

    def get_name(self):
        """Talabaning ismini qaytaradi"""
        return self.ism

    def get_lastname(self):
        """Talabaning familiyasini qaytaradi"""
        return self.familiya
```

```python
def get_fullname(self):
    """Talabaning ism-familiyasini qaytaradi"""
    return f"{self.ism} {self.familiya}"

def get_age(self,yil):
    """Talabaning yoshini qaytaradi"""
    return yil-self.tyil

def tanishtir(self):
    print(f"Ismim {self.ism} {self.familiya}. {
                {self.tyil} yilda tu'gilganman")
```

XUSUSIYATLARGA STANDART QIYMAT BERISH

```python
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self,ism,familiya,tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil
        self.bosqich = 1

    def get_info(self):
        return f"{self.ism} {self.familiya}. {self.bosqich}-bosqich talabasi "
```

Endi, `Talaba` klassidan yangi obyekt yaratganimizda har bir yangi talabaning kursi `1` ga teng bo'ladi.

```python
talaba1 = Talaba("Alijon","Valiyev",2000)
print(talaba1.get_info())
```

Natija: `Alijon Valiyev. 1-bosqich talabasi`

STANDART QIYMATNI O'ZGARTIRISH

```
talaba1.bosqich= 2
print(talaba1.bosqich)
```

Natija: `2`

```
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self,ism,familiya,tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil
        self.bosqich = 1

    def get_info(self):
        """Talaba haqida ma'lumot"""
        return f"{self.ism} {self.familiya}. {self.bosqich}-bosqich talabasi "

    def set_bosqich(self,bosqich):
        """Talabaning kursini yangilovchi metod"""
        self.bosqich = bosqich
```

Metodga murojat qilamiz:

```
talaba1.set_bosqich(3)
print(talaba1.get_info())
```

Natija: `Alijon Valiyev. 3-bosqich talabasi`

```python
class Talaba:
    """Talaba nomli klass yaratamiz"""
    def __init__(self,ism,familiya,tyil):
        """Talabaning xususiyatlari"""
        self.ism = ism
        self.familiya = familiya
        self.tyil = tyil
        self.bosqich = 1

    def get_info(self):
        """Talaba haqida ma'lumot"""
        return f"{self.ism} {self.familiya}. {self.bosqich}-bosqich talabasi "

    def set_bosqich(self,bosqich):
        """Talabaning kursini yangilovchi metod"""
        self.bosqich = bosqich

    def update_bosqich(self):
        """Talabanining bosqichini 1taga ko'paytirish"""
        self.bosqich += 1
```

```python
talaba1 = Talaba("Alijon","Valiyev",2000)
print(talaba1.get_info())

talaba1.update_bosqich() # 1 bosqichga oshiramiz
print(talaba1.get_info())
```

Natija: `Alijon Valiyev. 1-bosqich talabasi`

OBYEKTLAR O'RTASIDA MUNOSABAT

```python
class Fan():
    def __init__(self,nomi):
        self.nomi = nomi
        self.talabalar_soni = 0
        self.talabalar = []

    def add_student(self,talaba):
        """Fanga talabalar qo'shish"""
        self.talabalar.append(talaba)
        self.talabalar_soni += 1

    def get_students(self):
        return [talaba.get_info() for talaba in self.talabalar]
```

```python
matematika = Fan("Oliy Matematika")
talaba1 = Talaba("Alijon","Valiyev",2000)
talaba2 = Talaba("Hasan","Alimov",2001)
talaba3 = Talaba("Akrom","Boriyev",2001)
```

Talabalarni yangi fanimizga qo'shamiz:

```python
matematika.add_student(talaba1)
matematika.add_student(talaba2)
matematika.add_student(talaba3)
```

```python
print(matematika.talabalar_soni)
```

Natija: 3

NUQTA YOKI METOD?

Pythondagi obyketlarning o'ziga xos xususiyatlaridan biri, obyektning xususiyatiga nuqta orqali murojat qilish mumkin. Misol uchun avval yaratagn `talaba1` obyektining ismini bilish uchun `talaba1.ism` deb yozish kifoya.

Bu o'ziga yarasha qulay bo'lsada, bu usuldan foydalanmagan afzal. Sababi, vaqt o'tib klassingiz takomillashishi, uning ba'zi xususiyatlari o'zgarishi, o'chirilishi yoki almashtirilishi mumkin. Shunday holatlarda nuqta orqali murojat qilish siz kutgan natijani bermasligi va dastur xato ishlashiga olib kelishi mumkin. Bunday holatlarning oldini olish uchun esa, obyektning xususiyatlarini metod orqali olishni odat qilish tavsiya qilinadi. Huddi shu kabi, obyektning xususiyatlarini yangilash uchun ham alohida metodlar yozga afzal.

OBYEKTNING XUSUSIYATLARI VA METODLARINI KO'RISH

dir() FUNKSIYASI

dir() funksiyasi yordamida istalgan obyekt yoki klassning xususiyatlari va metodlarini ko'rib olishimiz mumkin:

```
>>> dir(Talaba)
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'get_age',
 'get_fullname',
 'get_info',
 'get_lastname',
 'get_name',
 'set_bosqich',
 'update_bosqich']
```

dunder metodlar

Lekin bunda har bir klass bilan keluvchi maxsus **dunder metodlar** ham chiqib keldi. Dunder metodlar ikki pastki chiziq (`__`) bilan boshlanadi va maxsus holatlar uchun saqlab qo'yilgan. Biz hozircha faqat `__init__` metodi bilan tanishdik, qolganlari bilan keyingi darslarimizda yana ko'rishamiz. Dunder metodlardan keyin esa biz murojat qilishimiz mumkin bo'lgan metodlar ro'yxati kelgan.

> Dunder — double underscore (ikki pastki chiziq) so'zlarining qisqartmasi.

Keling dunder metodlarni tashlab, bizga kerak metodlarni qaytaruvchi sodda funksiya yozamiz:

```
def see_methods(klass):
    return [method for method in dir(klass) if method.startswith('__') is False]

print(see_methods(Talaba))
```

Natija:

```
['get_age', 'get_fullname', 'get_info', 'get_lastname', 'get_name', 'set_bosqich',
'update_bosqich']
```

__dict__ METODI

Yuqorida zikr qilingan dunder metodlardan biri bu `__dict__` metodi bo'lib, bu metod **obyketning** xususiyatlarini lug'at ko'rinishida qaytaradi:

```
print(talaba1.__dict__)
```

Natija: `{'ism': 'Alijon', 'familiya': 'Valiyev', 'tyil': 2000, 'bosqich': 1}`

Natijadan faqatgina kalitlarni ajratib olsak, obyektning xususiyatlari chiqadi:

```
print(talaba1.__dict__.keys())
```

Natija: `dict_keys(['ism', 'familiya', 'tyil', 'bosqich'])`

Amaliy vazifalar

1-masala. Odam ismli class e'lon qiling va u bitta property qabul qilsin:

- ism
  hamda,
- salomlashish() nomli method i bo'lsin
  Ushbu class dan object hosil qilinsin, va salomlashish() chaqirilganda "Salom Falonchi" degan yozuv chiqsin
  Ma'lumotni foydalanuvchi kiritsin. Masalan:
  Input: Azamjon
  Output: Salom Azamjon

Amaliy vazifalar

2-masala. Odam nomli class e'lon qiling. Uning property lari:

- ism
  va methodlari:
- kuylash()
- eshitish()
- gapirish()
  Yuqoridagilardan foydalanib 2 ta obyekt hosil qiling.
  1- obyekt kuylaganida 2-obyekt uni eshitgandan keyin gapirish methodi ishga tushsin (kuylayotdan obyektga qandaydir feedback bersin)

Amaliy vazifalar

3-masala. Odam classi e'lon qiling. Uning property si:

- ism

  va, method lari:

- yugurish()
- yiqilish()

Classdan foydalanib obyekt hosil qiling. Obyekt yugurishni boshlaganda dastur 5 sekund kutib tursin va keyin yiqilish methodi ishga tushsin.

AMALIYOT

- `Avto` degan yangi klass yarating. Unga avtomobillarga doir bo'lgan bir nechta xususiyatlar (model, rang, korobka, narh va hokazo) qo'shing. Ayrim xususiyatlarga standart qiymat bering (masalan, `kilometer=0`)
- `Avto` ga oid obyektning xususiyatlarini qaytaradigan metodlar yozing
  - `get_info()` metodi avti haqida to'liq ma'lumotni matn ko'rinishida qaytarsin
- `Avto` ga oid obyektning xususiyatlarini yangilaydigan metodlar yozing.
  - `update_km()` metodi son qabul qilib olib, avtomobilning yurgan kilometrajini yangilab borsin
- Yangi, `Avtosalon` degan klass yarating va kerakli xususiyatlar bilan to'ldiring (salon nomi, manzili, sotuvdagi avtomobillar va hokazo)
- Avtosalonga yangi avtomobillar qo'shish uchun metod yozing
- Avtosalondagi avtomobillar haqida ma'lumot qaytaruvchi metod yozing
- Yuqoridagi obyektlar va ularga tegishli metodlarni tekshirib ko'ring
- `dir()` funksyasi va `__dict__` metodi yordamida o'zingiz yozgan va Pythondagi turli klass va obyektlarning xususiyatlari va metodlarini toping (`dir(str)` , `dir(int)` va hokazo)

Xulosa

Afzalliklari

- Parallel dasturlash – bir loyihaning turli qismlari bir vaqtda yaratilishi mumkin
- Vorislik tamoyili klasslardan qayta foydanalish imkonini beradi
- Polimorfizm tamoyili klasslarni moslashuvchan qiladi
- Klasslardan boshqa dastur va loyihalarda qayta-qayta foydalanish mumkin

Kamchiliklari

- Dasturlashga yangi qadam qo’yganlar uchun biroz tushunarsiz
- Har doim ham samarali emas
- Ba’zida dasturimizni haddan tashqari murakkablashtirib yuborishi mumkin

NAJOT TA’LIM

Bootcamp Foundation

4-OY. PreFoundation

E’tiboringiz uchun rahmat

Mallayev Oybek Usmankulovich

Najot ta’lim