PYTHON DASTURLASH TILIDA FUNKSIYALAR BILAN ISHLASH.

Kodirkulov Bunyod Narzullo o‘g‘li
Sirdaryo viloyati Shirin shahar
Shirin energetika kolleji informatika fani o'qituvchisi

Annotatsiya: Ushbu maqola Python dasturlash tilida **funksiyalar bilan ishlash** masalasiga bag‘ishlangan. Funksiya tushunchasi, uning tuzilishi va dasturiy ta’minotni modularlashtirishdagi ahamiyati keng yoritiladi. Ushbu maqola dasturchilarga **kodni optimallashtirish, takrorlanishdan qochish** va **murakkab masalalarni soddalashtirilgan modullarga ajratib** ishlashni o‘rgatish maqsadini ko‘zlagan. Python tilida funksiyalar bilan ishlashning barcha asosiy jihatlari yoritilgan bo‘lib, bu bilimlar dasturiy ta’minotni yaratishda muhim ahamiyat kasb etadi.

Kalit so‘zlar: funksiya, return, lampda, scope, *args, **kwargs, default, rekursiv, local, global.

Python dasturlash tilida funksiyalar — dasturlarni modularlashtirish va kodni qayta ishlatish imkonini beradigan muhim elementlardir. Katta dasturlarni turli qismlarga ajratish, har bir qismni funksiya sifatida ifodalash orqali kodning o‘qiluvchanligi va samaradorligi oshadi. Ushbu maqolada funksiyalarning turlari, parametrlar bilan ishlash, lambda funksiyalar, rekursiya va boshqa tushunchalar haqida batafsil ma'lumot beriladi.

Funksiya — bu muayyan vazifani bajarish uchun yozilgan va bir necha marta ishlatilishi mumkin bo‘lgan kodlar to‘plamidir. Funksiyalar yordamida murakkab dasturlarni kapsulalash va kodni soddalashtirish imkoniyati yaratiladi.

Funksiya quyidagicha tuziladi:

```
def funksiya_nomi(parametrlar):
    """Funksiya haqidagi qisqacha ma'lumot"""
    # Funksiya tanasi
    return natija  # Optional (qaytariladigan qiymat)
```

Funksiya tuzishning afzalliklari:

- Kodni qayta ishlatish imkoniyati.
- Katta dasturlarni boshqarishni soddalashtirish.
- Dasturiy ta'minot tuzilishini modulli qilish.
- Xatolarni aniqlash va tuzatishni yengillashtirish.

Funksiya parametrlar qabul qilib, argumentlar orqali ma'lumot oladi. Bu orqali funksiya dinamik tarzda har xil qiymatlar bilan ishlashi mumkin.

- Parametrlar: Funksiya e’lonida ishlatiladigan o‘zgaruvchilar.

- Argumentlar: Funksiya chaqirilganda beriladigan haqiqiy qiymatlar.

Funksiya parametrlar bilan:

```
def toliq_ism(ism, familiya):
    return f"{ism} {familiya}"
print(toliq_ism("Ali", "Valiyev"))  # Natija: Ali Valiyev
```

Standart qiymatli parametrlar: Agar argument uzatilmasa, default qiymatlar ishlatiladi.

```
def salom_ber(ism="Foydalanuvchi"):
    print(f"Salom, {ism}!")
salom_ber()        # Natija: Salom, Foydalanuvchi!
salom_ber("Ali")   # Natija: Salom, Ali!
```

Nomlangan argumentlar bilan ishlash: Foydalanuvchi funksiyaga nom bilan argument uzatsa, argumentlar tartibi muhim bo‘lmaydi.

```
print(toliq_ism(familiya="Qodirov", ism="Olim"))  # Natija: Olim Qodirov
```

`return` operatori yordamida funksiya natijasini chaqiruvchi kodga qaytarish mumkin. `return` ishlatilmasa, funksiya `None` qiymatini qaytaradi.

```
#Misol:
def kubi(x):
    return x  3
natija = kubi(3)
print(natija)  # Natija: 27
```

Bir funksiya ichida bir nechta `return` operatori bo‘lishi mumkin, ammo funksiya birinchi `return`ni uchratishi bilan to‘xtaydi.

```
def tekshir_son(x):
    if x % 2 == 0:
        return "Juft"
    return "Toq"
```

Lambda funksiyalar — qisqa va bir qatorlik funksiyalarni ifodalash uchun ishlatiladigan anonim funksiyalardir.

```
#Lambda funksiyaning umumiy sintaksisi:
lambda parametrlari: ifoda
#Misol:
kvadrat = lambda x: x  2
print(kvadrat(4))  # Natija: 16
```

Lambda funksiyalarni `map()`, `filter()` kabi funksiyalar bilan qo‘llash keng tarqalgan.

```
sonlar = [1, 2, 3, 4, 5]
juftlar = filter(lambda x: x % 2 == 0, sonlar)
print(list(juftlar))  # Natija: [2, 4]
```

Rekursiya — funksiya o‘zini-o‘zi chaqiradigan jarayondir. Ushbu texnikani masalalarni qayta-qayta kichik bo‘laklarga ajratib yechishda qo‘llash mumkin.

#Misol: Faktorialni hisoblash:

```
def faktorial(n):
if n == 0:
return 1
else:
return n * faktorial(n - 1)
print(faktorial(5))  # Natija: 120
```

Rekursiv funksiyalarni ishlatishda stack overflow (funksiyaning haddan tashqari ko‘p chaqirilishi) muammosidan saqlanish uchun har doim tugash shartini yozish muhim.

Python’da o‘zgaruvchilarni ko‘rinish sohasi ikki xil bo‘ladi:

- Mahalliy (local): Funksiya ichida e’lon qilingan va faqat shu funksiya ichida amal qiluvchi o‘zgaruvchilar.
- Global: Funksiyadan tashqarida e’lon qilingan va barcha funksiyalarda ko‘rinadigan o‘zgaruvchilar.

#Misol:

```
x = 10  # Global o'zgaruvchi
def uzgar():
    x = 5  # Mahalliy o'zgaruvchi
    print("Mahalliy x:", x)
uzgar()  # Natija: Mahalliy x: 5
print("Global x:", x)  # Natija: Global x: 10
```

Agar funksiya ichida global o‘zgaruvchini o‘zgartirish kerak bo‘lsa, `global` kalit so‘zi ishlatiladi:

```
x = 10
def uzgartir():
    global x
    x = 20
uzgartir()
print(x)  # Natija: 20
```

Pythonda funksiyalarga o‘zgaruvchan miqdordagi argumentlar berish uchun *args va **kwargs ishlatiladi.

‘*args’ argumentlarni kortej sifatida qabul qiladi.

```
def yigindi(*sonlar):
    return sum(sonlar)
print(yigindi(1, 2, 3, 4))  # 10
```

‘**kwargs’ esa kalit-qiymat juftliklarini lug‘at shaklida qabul qiladi.

```
def ma'lumotlar(kwargs):
    for kalit, qiymat in kwargs.items():
        print(f"{kalit}: {qiymat}")
```

```
ma'lumotlar(ism="Ali", yosh=25, kasb="Dasturchi")
# Natija:
# ism: Ali
# yosh: 25
# kasb: Dasturchi
```

Funksiya ichida boshqa funksiyani chaqirish orqali murakkab vazifalarni oddiy qismlarga bo‘lib yechish mumkin.

#Misol:

```
def tashqi_funksiya(x):
    def ichki_funksiya(y):
        return y  2
    return ichki_funksiya(x) + 10
print(tashqi_funksiya(5))  # Natija: 35
```

Xulosa:

Funksiyalar — Python dasturlash tilida muhim tushunchalardan biri bo‘lib, ular kodni optimallashtirish, takrorlanishni kamaytirish va dasturiy ta'minotning tuzilishini yaxshilashga yordam beradi. Ushbu maqolada parametrlar, qaytish qiymatlari, rekursiya, lambda funksiyalar va *args, **kwargs kabi tushunchalar bilan ishlash ko‘rib chiqildi. To‘g‘ri foydalanilganda, funksiyalar dasturchilar uchun katta qulayliklar yaratadi va kodning tushunarli bo‘lishini ta'minlaydi.

Foydalanilgan adabiyotlar:

1. Anvar Narzullayev Pythonda dasturlash asoslari

2. Mark Lutz, Learning Python. O'Reilly Media, 2020.

3. www.mohirdev.uz