# Klausimai

## Trumpų Atsakymų Klausimynas

**Kas yra žodynas (angl. dictionary) Python'e?** Žodynas yra nerikiuotas raktas-reikšmė porų rinkinys, naudojamas struktūrizuotai susieti duomenis, kad būtų galima greitai rasti susijusias reikšmes. Raktai turi būti unikalūs ir nekintami.

**Kaip galite sukurti žodyną Python'e? Pateikite du būdus.** Žodyną galima sukurti naudojant lenktinius skliaustus {}, nurodant raktas-reikšmė poras, pvz., {"raktas": "reikšmė"}. Kitas būdas – naudoti dict() funkciją, kur raktai aprašomi kaip vardiniai funkcijų parametrai, pvz., dict(raktas="reikšmė").

**Kokie duomenų tipai gali būti naudojami kaip raktai žodyne? Pateikite pavyzdžių.** Raktai gali būti eilutės (pvz., "vardas"), skaičiai (pvz., 1, 3.14), bool reikšmės (True, False), tuple tipo struktūros (pvz., (1, 2, 3)) ir None. Svarbu, kad raktai būtų nekintami (angl. immutable).

**Kokie duomenų tipai negali būti naudojami kaip raktai žodyne? Kodėl?** Keičiami (angl. mutable) duomenų tipai, tokie kaip sąrašai (list), rinkiniai (set) ir patys žodynai (dictionary), negali būti raktais. Tai yra dėl to, kad raktai turi būti nekintami, kad būtų užtikrintas jų unikalumas ir patikimas veikimas.

**Kaip pasiekti reikšmę žodyne naudojant raktą?** Reikšmę žodyne pasiekiame naudojant laužtinius skliaustus [] ir nurodant norimą raktą, pvz., žodynas["raktas"]. Jeigu raktas neegzistuoja, programa išmes klaidą KeyError, todėl rekomenduojama naudoti get() metodą.

**Kaip pridėti naują raktas-reikšmė porą į žodyną?** Naują raktas-reikšmė porą į žodyną galima pridėti naudojant priskyrimo operatorių =. Pavyzdžiui, žodynas["naujas\_raktas"] = "nauja\_reikšmė" pridės naują porą arba pakeis esamos rakto reikšmę.

**Kaip modifikuoti esamą reikšmę žodyne?** Reikšmę žodyne modifikuojame priskirdami naują reikšmę jau egzistuojančiam raktui. Pavyzdžiui, žodynas["esamas\_raktas"] = "nauja\_reikšmė" pakeis esamą reikšmę.

**Kaip iteruoti per žodyno raktus ir reikšmes?** Pateikite pavyzdžius. Norint iteruoti raktus, naudojamas metodas keys(), pvz., for raktas in žodynas.keys():. Norint iteruoti reikšmes, naudojamas metodas values(), pvz., for reikšmė in žodynas.values():. Norint iteruoti raktus ir reikšmes kartu, naudojamas metodas items(), pvz., for raktas, reikšmė in žodynas.items():.

**Kaip saugiai gauti reikšmę iš žodyno, kad išvengtumėte KeyError?** Saugiam reikšmės gavimui iš žodyno naudojamas metodas get(). Jis gali grąžinti reikšmę pagal raktą arba grąžinti None, jeigu raktas neegzistuoja. Taip pat galima nurodyti numatytą reikšmę, kuri bus grąžinta, jeigu raktas nerastas, pvz., žodynas.get("raktas", "numatyta\_reikšmė").

**Kaip pašalinti raktą iš žodyno?** Pateikite tris būdus. Raktą iš žodyno galima pašalinti naudojant del operatorių, pvz., del žodynas["raktas"], arba pop() metodą, kuris gražina reikšmę, pvz., žodynas.pop("raktas"). Be to, galima naudoti popitem() metodą, kuris pašalina paskutinę raktas-reikšmė porą iš žodyno ir ją gražina.

## Atsakymų Raktas

Žodynas yra nerikiuotas raktas-reikšmė porų rinkinys, naudojamas struktūrizuotai susieti duomenis, kad būtų galima greitai rasti susijusias reikšmes. Raktai turi būti unikalūs ir nekintami.

Žodyną galima sukurti naudojant lenktinius skliaustus {}, nurodant raktas-reikšmė poras, pvz., {"raktas": "reikšmė"}. Kitas būdas – naudoti dict() funkciją, kur raktai aprašomi kaip vardiniai funkcijų parametrai, pvz., dict(raktas="reikšmė").

Raktai gali būti eilutės (pvz., "vardas"), skaičiai (pvz., 1, 3.14), bool reikšmės (True, False), tuple tipo struktūros (pvz., (1, 2, 3)) ir None. Svarbu, kad raktai būtų nekintami (angl. immutable).

Keičiami (angl. mutable) duomenų tipai, tokie kaip sąrašai (list), rinkiniai (set) ir patys žodynai (dictionary), negali būti raktais. Tai yra dėl to, kad raktai turi būti nekintami, kad būtų užtikrintas jų unikalumas ir patikimas veikimas.

Reikšmę žodyne pasiekiame naudojant laužtinius skliaustus [] ir nurodant norimą raktą, pvz., žodynas["raktas"]. Jeigu raktas neegzistuoja, programa išmes klaidą KeyError, todėl rekomenduojama naudoti get() metodą.

Naują raktas-reikšmė porą į žodyną galima pridėti naudojant priskyrimo operatorių =. Pavyzdžiui, žodynas["naujas\_raktas"] = "nauja\_reikšmė" pridės naują porą arba pakeis esamos rakto reikšmę.

Reikšmę žodyne modifikuojame priskirdami naują reikšmę jau egzistuojančiam raktui. Pavyzdžiui, žodynas["esamas\_raktas"] = "nauja\_reikšmė" pakeis esamą reikšmę.

Norint iteruoti raktus, naudojamas metodas keys(), pvz., for raktas in žodynas.keys():. Norint iteruoti reikšmes, naudojamas metodas values(), pvz., for reikšmė in žodynas.values():. Norint iteruoti raktus ir reikšmes kartu, naudojamas metodas items(), pvz., for raktas, reikšmė in žodynas.items():.

Saugiam reikšmės gavimui iš žodyno naudojamas metodas get(). Jis gali grąžinti reikšmę pagal raktą arba grąžinti None, jeigu raktas neegzistuoja. Taip pat galima nurodyti numatytą reikšmę, kuri bus grąžinta, jeigu raktas nerastas, pvz., žodynas.get("raktas", "numatyta\_reikšmė").

Raktą iš žodyno galima pašalinti naudojant del operatorių, pvz., del žodynas["raktas"], arba pop() metodą, kuris gražina reikšmę, pvz., žodynas.pop("raktas"). Be to, galima naudoti popitem() metodą, kuris pašalina paskutinę raktas-reikšmė porą iš žodyno ir ją gražina.

## Esė Klausimai

Paaiškinkite, kodėl žodynai yra svarbi duomenų struktūra Python'e. Aptarkite, kaip jie gali būti naudojami organizuoti ir valdyti duomenis, pateikdami konkrečių pavyzdžių.

Apibūdinkite žodynų lankstumą, kalbant apie raktų ir reikšmių tipus. Išanalizuokite, kokiais atvejais tai gali būti naudinga, ir kada reikėtų būti atsargiems, renkantis raktų tipus.

Palyginkite ir supriešinkite skirtingus būdus iteruoti per žodynus (per raktus, per reikšmes, per raktų ir reikšmių poras). Aptarkite, kada kiekvienas iš šių būdų yra tinkamiausias.

Aptarkite žodynų sąrašų naudojimą. Kaip galėtumėte apdoroti sudėtingas struktūras, kai sąrašai ir žodynai yra kombinuojami? Pateikite pavyzdžių, kaip tai galėtų būti pritaikyta realiose situacijose.

Apžvelkite skirtingus būdus, kaip pašalinti elementus iš žodyno. Išanalizuokite, kokios yra kiekvieno metodo ypatybės, ir kada būtų tinkamiausia naudoti del, pop() arba popitem() metodus.

## Žodynėlis

**Žodynas (dictionary)**: Python'e, duomenų struktūra, kuri saugo raktas-reikšmė poras.

**Raktas (key)**: Unikalus identifikatorius žodyne, naudojamas pasiekti susijusią reikšmę. Turi būti nekintamas duomenų tipas.

**Reikšmė (value)**: Duomenys, susieti su raktu žodyne. Gali būti bet kokio duomenų tipo.

**Raktas-reikšmė pora (key-value pair)**: Vienas elementas žodyne, kurį sudaro raktas ir jam priskirta reikšmė.

**Iteravimas (iteration)**: Procesas, kai einama per visus duomenų rinkinio elementus (pvz., žodyno raktus arba reikšmes)

**Mutable**: Keičiamas duomenų tipas, kurį galima modifikuoti. Pvz., sąrašas.

**Immutable**: Nekeičiamas duomenų tipas, kurį sukurus nebegalima modifikuoti. Pvz., eilutė ar skaičius.

**KeyError**: Klaida, kuri iškyla, kai bandote pasiekti žodyno reikšmę su neegzistuojančiu raktu.

**get() metodas**: Žodyno metodas, kuris saugiai grąžina reikšmę pagal raktą, arba nurodytą numatytą reikšmę, jei rakto nėra.

**keys() metodas**: Žodyno metodas, kuris grąžina visus žodyno raktus kaip sąrašą.

**values() metodas**: Žodyno metodas, kuris grąžina visas žodyno reikšmes kaip sąrašą.

**items() metodas**: Žodyno metodas, kuris grąžina visas žodyno raktų-reikšmių poras kaip sąrašą.

**del**: Operatorius, naudojamas pašalinti raktą iš žodyno.

**pop() metodas**: Žodyno metodas, naudojamas pašalinti raktą ir grąžinti jo reikšmę.

**popitem() metodas**: Žodyno metodas, naudojamas pašalinti ir grąžinti paskutinę raktų-reikšmių porą.

**clear() metodas**: Žodyno metodas, naudojamas pašalinti visus elementus iš žodyno.

**zip() funkcija**: Funkcija, kuri sujungia du ar daugiau sąrašų elementus į poras.

## DUK

1. **Kas yra žodynas (angl. dictionary) Python'e ir kam jis naudojamas?**   
   Žodynas Python'e yra nerikiuota raktas-reikšmė porų rinkinys. Jis naudojamas struktūrizuotai susieti duomenis, kad būtų galima greitai rasti susijusias reikšmes pagal unikalų raktą. Raktai žodyne turi būti unikalūs ir nekintami. Žodynai yra labai lanksti duomenų struktūra, leidžianti saugoti įvairius duomenis ir lengvai juos pasiekti.
2. **Kaip galima sukurti žodyną Python'e? Kokiais būdais galima pridėti raktas-reikšmė poras?**   
   Žodyną Python'e galima sukurti dviem pagrindiniais būdais:
   1. Naudojant lenktinius skliaustus {} ir nurodant raktas-reikšmė poras, pavyzdžiui, {"raktas": "reikšmė"}.
   2. Naudojant dict() funkciją, kur raktai aprašomi kaip vardiniai funkcijų parametrai, pavyzdžiui, dict(raktas="reikšmė"). Naujas raktas-reikšmė poras galima pridėti naudojant priskyrimo operatorių =, pavyzdžiui, žodynas["naujas\_raktas"] = "nauja\_reikšmė". Taip pat galima naudoti update() metodą, norint pridėti kelias poras iš karto.
3. **Kokie duomenų tipai gali būti naudojami kaip raktai žodyne, o kurie negali? Kodėl?**   
   Kaip raktai žodyne gali būti naudojami nekintami (angl. immutable) duomenų tipai, tokie kaip:
   1. Eilutės (pvz., "vardas").
   2. Skaičiai (pvz., 1, 3.14).
   3. Bool reikšmės (pvz., True, False).
   4. Tuple tipo struktūros (pvz., (1, 2, 3)).
   5. None. Keičiami (angl. mutable) duomenų tipai, tokie kaip sąrašai (list), rinkiniai (set) ir patys žodynai (dictionary), negali būti raktais, nes jie gali būti modifikuojami, ir tai pažeistų žodyno veikimo principą, nes raktai turi būti unikalūs ir nekintami.
4. **Kaip pasiekti reikšmę žodyne naudojant raktą? Kas nutinka, jei raktas neegzistuoja?**  
   Reikšmę žodyne galima pasiekti naudojant laužtinius skliaustus [] ir nurodant norimą raktą, pavyzdžiui, žodynas["raktas"]. Jei nurodytas raktas neegzistuoja, programa išmes klaidą KeyError. Norint to išvengti, rekomenduojama naudoti get() metodą.
5. **Kaip saugiai gauti reikšmę iš žodyno, kad išvengtumėte KeyError? Kaip naudoti get() metodą?**   
   Saugiam reikšmės gavimui iš žodyno rekomenduojama naudoti get() metodą. Šis metodas grąžina reikšmę pagal raktą, jei jis egzistuoja, arba grąžina None, jei rakto nėra. Taip pat galima nurodyti antrą argumentą get() metode, kuris bus grąžintas kaip numatytoji reikšmė, jei rakto nėra, pavyzdžiui, žodynas.get("raktas", "numatyta\_reikšmė").
6. **Kaip modifikuoti esamą reikšmę žodyne? Ar galima pridėti kelis naujus raktus iš karto?** Norint modifikuoti esamą reikšmę žodyne, reikia priskirti naują reikšmę jau egzistuojančiam raktui, pavyzdžiui, žodynas["esamas\_raktas"] = "nauja\_reikšmė". Kelis naujus raktus su reikšmėmis galima pridėti iš karto naudojant update() metodą ir nurodant naujas raktas-reikšmė poras kaip žodyną, pavyzdžiui, žodynas.update({"naujas\_raktas1": "reikšmė1", "naujas\_raktas2": "reikšmė2"}).
7. **Kaip iteruoti per žodyno raktus ir reikšmes? Kokius metodus reikia naudoti?**   
   Galima iteruoti per žodyną trimis būdais:
   1. Iteruoti per raktus naudojant keys() metodą: for raktas in žodynas.keys():.
   2. Iteruoti per reikšmes naudojant values() metodą: for reikšmė in žodynas.values():.
   3. Iteruoti per raktus ir reikšmes kartu naudojant items() metodą: for raktas, reikšmė in žodynas.items():. keys() ir values() metodai grąžina sąrašus, kuriuos galima iteruoti. Metodas items() grąžina sąrašą su tuple tipo objektais, kiekvienas turintis raktą ir reikšmę.
8. **Kaip pašalinti elementus (raktas-reikšmė poras) iš žodyno? Kokius metodus galima naudoti?**   
   Elementus iš žodyno galima pašalinti trimis pagrindiniais būdais:
   1. Naudojant del operatorių, nurodant raktą, kurį norite pašalinti: del žodynas["raktas"].
   2. Naudojant pop() metodą, kuris pašalina raktą ir grąžina jo reikšmę: žodynas.pop("raktas"). Jei raktas neegzistuoja, išmes klaidą KeyError, bet galima nurodyti numatytąją reikšmę, kuri bus grąžinta, jei rakto nėra.
   3. Naudojant popitem() metodą, kuris pašalina paskutinę raktas-reikšmė porą iš žodyno ir ją grąžina. Be to, galima naudoti clear() metodą, kuris pašalina visus elementus iš žodyno.