Sisend ja väljund

Sisend- ja väljundlaused (*input / output*) on ettenähtud (nagu nimigi ütleb) programmile andmete edastamiseks ja tulemuste väljatrükkimiseks (ekraanile). Selleks, et programm oleks interaktiivne, et kasutaja saaks arvutiga programmi läbi suhelda. Lisaks sellele on programmidel teinekord vaja suhelda ka failidega – andmeid lugeda või kirjutada. Piirdume esialgu sisendiga klaviatuurilt ja väljundiga ekraanile, mida tavatsetakse kutsuda standardsisendiks ja standardväljundiks. Nii sisend kui väljund tehakse Python-programmis **sisefunktsioonide** *(built-in function)* abil. Mõlemal puhul on toimunud muutused võrreldes 2.x versiooniga.

Funktsioon print() andmete väljastuseks

Väljundi trükkimiseks ekraanile kasutatakse sisefunktsiooni **`print()`**. Väljatrükitav informatsioon tuleb paigutada sõna `print` järel sulgudesse. Väljastada saab tekste (stringe) ja muutujate väärtuseid. Lisada saab ka reavahetusi ja vormindust. Väljastatav tekst pannakse kas jutumärkide või ülakomade vahele (sel teel tekib stringkonstant ja ta on eristatav muutuja nimest). Järgnevate näidete toimimist tasub ise järgi proovida.

```
# väljastatakse tekst Tere kool! jutumärkideta
print("Tere kool!")
# sõnad Tere ja kool! On eraldi ridadel (vahele on väljastatud reavahetus)
print("Tere\nkool!")
```

Sümbol "**`\n`**" tähendab reavahetust (*new line*), ka mitmes teises keeles, nt C-s. Ükskõik kuhu väljatrükilauses see lisatakse, on tulemuseks reavahetus ekraanile jõudvas tekstis. Sellest jätame meelde, et reavahetus on samamoodi trükitav märk, nagu suvaline täht või number. Ilma seda eraldi märkimata lisatakse reavahetus automaatselt väljundlause lõppu. Väljundlausega kaasnevat automaatset reavahetust saab vältida `end`-parameetri lisamisega. Järgnevas näites lisatakse väljundi viimaseks sümboliks reavahetuse asemel tühik.

```
print("Tere kool!", end=" ")
```

Väljundisse tühja rea saamiseks sobib kirjutada:

```
print()
```

Mitme väärtuse väljastamiseks ühe lausega saab kõik väljundi osad eraldada üksteisest komadega (tekib loetelu). Koma põhjustab väljundis täiendava tühiku. Soovides tühikut vältida (või panna selle asemel mõni teine sümbol), võib lisada `print`-funktsiooni `sep`-parameetri. Näiteks nii:

```
print("Tere","kool!", sep="")
```

Nüüd trükitakse Tere ja kool! kokku: Terekool!

Muutujate väärtuste väljastamiseks on erinevad võimalused. Üks lihtsamaid on järgmine:

```
pindala = 20.12345
print("Pindala on", pindala)
```

Põhimõte on lihtne - erinevad väljatrükitavad osad esitatakse sulgude vahel loeteluna, mille elemendid eristatakse üksteisest komadega.

Sama tulemuse saab väljastada ka kasutades vormindamissümbolit %. Selgituseks: muutuja pindala väärtus pannakse lauseds %f asemele. Lause lõppu lisatakse ka ., täpselt nii, nagu see stringis näha on.

```
print("Pindala on %f." % (pindala))
```

Väljundi ümardamiseks sobib siin kontekstis kasutada samuti vormindamise võimalusi. Et näidata pindala täpsusega 2 kohta peale koma, võime kirjutada lause järgmiselt (`0` peale `%`-märki ütleb, et mingeid lisatühikuid kuhugi ei lisata, `.2` aga nõuab ümardamist sajandikeni):

```
print("Pindala on %0.2f." % (pindala))
```

Vormindussümbolite tähendused:

`%s` – asendatakse teksti ehk stringiga *(string).*

`%d` – asendatakse täisarvuga *(decimal)*

`%f` – asendatakse ujukomaarvuga *(float)*

Kirjeldatud vormindusviis on omane näiteks ka C-keelele. Python 3-s on lisandunud ka alternatiivne variant väljundi moodustamiseks ja vormindamiseks. Eelnev pindala trükkimise näide näeb välja selline:

```
print("Pindala on {:0.2f}.".format(pindala))
```

Arvukaid näiteid nii vana kui ka uue vorminduse võimaluste kohta võib vaadata siit: https://pyformat.info/

Funktsioon input() andmete sisestuseks

Andmete sisseküsimiseks saab kasutada funktsiooni **`input`**. Funktsiooniga saab väljastada ekraanile **küsimuse** *(ik prompt, ek viip)*, mis peaks loogiliselt sisendile eelnema (nt *Sisesta kolmnurga kõrgus*) ja lasta sisestada küsitud väärtus. Väärtuse saab endale muutuja ja see on alati stringi (teksti) tüüpi. Edasi on võimalik teisi funktsioone kasutades muuta sisestatud väärtus näiteks arvuks (täisarvuks, ujukomaarvuks), et rehkendada saaks. Et `input()`-i puhul on tegemist funktsiooniga, tagastab ta sisestatud väärtuse ja see tuleb omistada mingisse muutujasse. Kui omistamine unustada, läheb sisend kaotsi.

Järgnevas näites küsitakse kasutajalt tema nime ja seejärel trükitakse see ekraanile. Vahepeal on nimi muutujas `minu_nimi`.

```
minu_nimi = input("Mis su nimi on? ")
print("Tere",minu_nimi,". Sinuga oli tore tutvuda!")
```

Kui sisestatud väärtusega on vaja pärast rehkendada, tuleb string teisendada arvuks. Selleks sobivad funktsioonid **`int()`** ja **`float()`**. Esimene nendest teeb stringi täisarvuks, teine ujukomaarvuks.

Järgnevas näites sisestatakse kaks arvu ja väljastatakse nende summa. Proovi, mis juhtub arvude summaga, kui täisarvuks teisendamine (`int()`) unustada.

```
# Küsitakse kaks arvu, teisendatakse, liidetakse ja väljastatakse summa
# arv1, arv2 - liidetavad
# summa - arvude summa
arv1 = input("Sisesta 1. arv ")
arv2 = input("Sisesta 2. arv ")
summa = int(arv1) + int(arv2)
print ("Arvude summa on",summa)
```

Kuna stringi tüüpi muutujatega `arv1` ja `arv2` ei ole suure tõenäosusega rohkem midagi teha, võib teisendada ka kohe sisestamise käigus:

```
arv1=int(input("Sisesta 1. arv "))
```

Nüüd on `arv1` täisarv ja liitmisel ei tule teda enam teisendada. Kogu näide on järgmine:

```
arv1 = int(input("Sisesta 1. arv "))
arv2 = int(input("Sisesta 2. arv "))
summa = arv1 + arv2
print ("Arvude summa on",summa)
```

Eelmisest näitest on näha ka olukord, kus kaks funktsiooni on nö üksteise sisse paigutatud. Funktsioonist `input()` saadud väärtus suunatakse otse teisendusfunktsiooni `int()`.

Funktsiooni `input()` saab kasutada ka selleks, et programmi täitmist peatada kuni vajutatakse Enter-klahvi. Näiteks võimaldab see pikema väljundi korral seda osadena väljastada ja lugeda.