

Valiku konstruktsioon ja if-lause

Algoritmide ja programmide koostamisel on üheks oluliseks komponendiks ehk ehituskiviks **valik**. Valiku abil saab jätta otsustamise (kas teha ühtesid toiminguid või teisi toiminguid) algoritmi/programmi täitmise ajaks. Ja seega sõltub otsus antud hetkeks tekkinud olukorrast.

Esimeseks oluliseks valikus kasutatavaks osaks on loogikaavaldis (vt vastavat materjali). Et loogikaavaldisel oli kaks võimalikku väärtust (*True* ja *False*), saab selle järgi otsustada kahe võimaliku toimingute jada vahel. Rohkemate jadade vahel valimiseks on vaja mitut valikukonstruktsiooni koos loogikaavaldistega.

Teistpidi öeldes – valikulause annab võimaluse kirja panna kaks alternatiivset toimingute järgnevust ja programmi / algoritmi täites saab alati valida vaid ühe nendest järgnevustest. Kui loogikaavaldise väärtus on *True*, valitakse üks jada ja kui *False*, siis teine jada.

Plokkiskeemil kasutatavat hargnemist koos valikuplokiga saab üsna üksühele programmeerimiskeelde „tõlkida“. Selleks on peaaegu igas keeles olemas nn **if-lause**, mille toimimispõhimõte erinevates keeltes on väga sarnane. If-lause sisaldab alati loogikaavaldist ning kui selle avaldise väärtus on tõene, täidetakse need laused, mis paiknevad nn if-osas. Peale seda jätkub programmi täitmine if-lausele järgnevate lausetega, st else-osas paiknevatest lausetest hüpatakse üle. Kui loogikaavaldise väärtus on väär, siis hüpatakse üle if-osas olevatest lausetest, täidetakse need, mis jäävad else-ossa ning seejärel jätkatakse programmi täitmist if-lausele järgneva lausega.

Pythoni's on if-lausel mitu erinevat vormi (sõltuvalt valikute arvust), kuid kõige tüüpilisem näeb välja järgmine:

```
if loogikaavaldis:
    toimingud, kui avaldis tõene
else:
    toimingud, kui avaldis väär
```

Siinjuures tuleb arvestada, et if ja else-osas esitatud toimingud tuleb trükkida taandega, teisisõnu – tuleb programmikoodi treppida. Toiminguid (programmi lauseid) võib mõlemas osas olla mitu ja kõigi taane peab olema ühesugune. Taane näitab lausete kuulumist ühte plokki. Lisaks näitab ta, et taandega laused kuuluvad mingi keelekonstruktsiooni sisse (antud juhul kuuluvad need laused if-lause sisse).

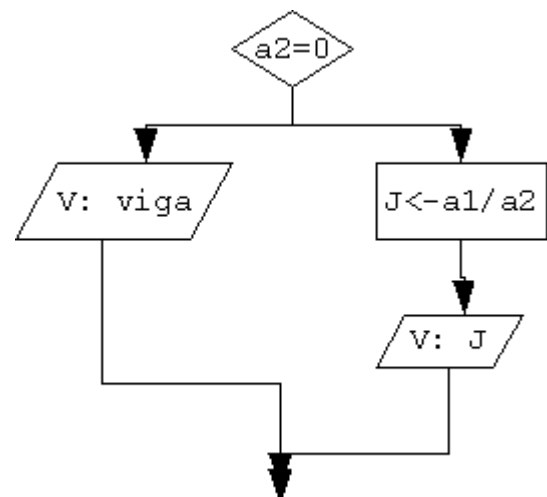
Näide: Jagatakse kaht arvu. Kui jagaja on null, antakse veateade. Pane tähele, et jagamine toimub vaid siis, kui jagaja ei ole 0.

```
if a2 == 0:
    print ("Viga")
else:
    J = a1/a2
    print ("Jagatis on ", J)
```

Tavaliselt saab programmeerimiskeeltes kasutada nn ühe poolega if-lause. See tähendab, et toimingud on ettenähtud vaid juhuks, kui loogikaavaldis oli tõene. Vastasel juhul ei tehta midagi erilist ning programmi täitmine jätkub kohe peale if-lause lõpu. Sellisel juhul else-osa programmi üldse ei lisata.

Pythoni vastav konstruktsioon näeb välja järgmine:

```
if loogikaavaldis:
    toimingud, kui avaldis tõene
```



Mitmene valik. Lisaks saab Pythoni if-lausetes kasutada `elif`-osa, mis on alternatiiviks mitme if-lause üksteise else-osadesse paigutamisele. Selline lause võimaldab teha valikuid paljude võimaluste hulgast ja täidetakse vaid üks valikutest – esimene, mille loogikatingimus on tõene. Edasi jätkub programmi täitmine peale if-lause. Lause `if...elif...else` on alternatiiviks valdavalt teistes keeltes kasutusel olevatele `switch` ja `case`-mitmese valiku lausetele.

```
if loogikaavaldis1:
    toimingud
elif loogikaavaldis2:
    toimingud
:
elif loogikaavaldisN:
```

```
    toimingud
```

```
else:
```

```
    toimingud, kui ükski eelnev avaldis ei olnud tõene
```

Vaata näiteprogramme `kooli_minek.py` `suurem_vaiksem_if.py` ja `suurem_vaiksem_elif.py`

If-lauses võib `else`-osa ka puududa. Sel juhul tehaks midagi vaid siis, kui loogikatingimus tõeseks osutub. Vastasel juhul minnakse kohe peale `if`-lauset olevate lausete kallale. Ka `if - elif` konstruktsiooni puhul võib `else`-osa samuti puududa. Tühja `else`-osa ei ole mõistlik programmi koodi kirjutada.