Divide et impera!

Alamprogrammid – protseduurid ja funktsioonid

Järgnevas materjalis on enamus definitsioone võetud EVS-ISO/IEC 2382-15:2001 ehk Infotehnoloogia sõnastikust (http://eki.ee/dict/its/).

Alamprogramm *(subprogram)* on programmeerimiskeeles kirjutatud käskude jada, mis täidab kindlalt ja loogiliselt piiritletud ning programmi jaoks vajaliku ülesande. Alamprogramm kutsutakse välja teisest programmi osast või moodulist.

Erinevates programmeerimiskeeltes kasutatakse mõisteid *procedure*, *function*, *routine*, *method*, *subprogram*, *subroutine*. Eesti keeles peamiselt **alamprogramm**, **protseduur**, **funktsioon**, **meetod**.

Inglise keeles on ka üldmõiste *callable unit*.

Alamprogramme võib jagada kahte klassi:

- **protseduur** - alamprogramm, mis ei tagasta andmeväärtust, välja arvatud parameetri mehhanismi osana.
- **funktsioon** - alamprogramm (harilikult formaalparameetritega), mis tekitab andmeväärtuse, mille ta tagastab väljakutsekohta.

Mõnedes programmikeeltes (näiteks C, C++, Python) ei eristata protseduuri keeletarindit funktsiooni keeletarindist, välja arvatud selles, et tagastatavad andmeväärtused võivad olla tühjad või neid ei kasutata.

Standardfunktsioonid ehk **sisefunktsioonid** *(built-in-functions)* on otse keele koosseisu kuuluvad funktsioonid (reeglina ei paikne eraldi teekides, moodulites).

Programmeerimiskeeltes on vahendid, et programmeerija saaks ise funktsioone lisada.

Funktsiooni **käivitamiseks** *(invoke)* tuleb ta programmi lausete osas **väljakutsuda** *(call)*.

Funktsioonikutse (*function call*) on keeletarind, mis annab tegelikud parameetrid mingi funktsiooni täitmise algatamiseks ja kutsub esile funktsiooni täitmise.

Funktsioonikutset saab kasutada operandina avaldises või tegeliku parameetrina alamprogrammi kutses.

Et Pythonis on pigem funktsioonid, siis nimetame neid edaspidi ka nii.

Milleks funktsioonid?

Funktsioonid ei ole tegelikult hädavajalikud, kõik saaks tehtud ka teisi keele lauseid kasutades, kuid inimese jaoks muudavad nad koodi paremini hallatavaks.

- Saab vaadelda mitut keelelauset ühe toiminguna ja vähendada seeläbi koodi keerukust inimesele – vähem muutujaid ja koodiridu.
- Saab vältida sama koodi kordamist ja kopeerimisel tekkivaid vigu (nt mõne muutujanime muutmine)
- Tagab turvalisema koodi korduvkasutamise ja võimaluse teha hiljem muudatusi ühes kohas.
- Saab luua üldisema meetodi ülesande lahendamiseks – funktsioonile edastatakse algandmed ja ta teeb oma tööd erinevate algandmetega.
- Toetab tööjaotust nii programmi koodi osade kui ka inimeste vahel.

Andmevahetus funktsioonidega - argumendid ja parameetrid

Funktsiooniga andmeid vahetada on soovitav argumentide ehk parameetrite abil. IT sõnastik käsitleb neid mõisteid sünonüümidena

Argument ehk **parameeter** on väärtused, mis antakse funktsioonile ette lähteandmetena, tarkvaramoodulite vaheliseks väärtuste edastuseks kasutatav konstant, muutuja või avaldis .

Eristatakse

- **formaalparameeter** *(formal parameter)* ehk **tühi argument** (*dummy argument)* on teatud moodulite / funktsioonide deklaratsioonis defineeritud parameeter, mis kutses seotakse tegeliku parameetriga.
- **tegelik parameeter** ehk **tegelik argument** *(actual parameter / argument* ka *calling parameter)* on näiteks avaldis, identifikaator või muu keeletarind, mida kasutatakse andmeobjekti sidumiseks vastava funktsiooni deklaratsiooniga kutse korral. Ehk tegelik parameeter on funktsioonikutses.

Parameetrite edastamine *(parameter passing)*

- **Väärtuskutse** ehk **kutse väärtusega** *(call by value).* Kutse, milles kutsuv programm annab kutsutavale moodulile / funktsioonile talle üleantavate parameetrite tegelikud väärtused. Väärtuskutses ei saa kutsutav funktsioon muuta parameetrite väärtusi. Väärtus edastatakse muutuja, konstandi või avaldisena.
- **Aadresskutse** ehk **kutse aadressiga** *(call by reference / address / location).* Kutse, milles kutsuv programm annab kutsutavale moodulile / funktsioonile talle üleantavate parameetrite (mälu)aadressi. Kutsutav funktsioon saab muuta programmi salvestatud parameetrite väärtusi.

Pytonis kasutatakse väärtuskutset.

Parameetrite arv funktsiooni päises peab tüüpiliselt vastama argumentide arvule funktsiooni kutses ning väärtused edastatakse järjekorra alusel.

Sõltuvalt keelest võib lisaks esineda võimalus kirjeldada vaikeväärtustega parameetreid või edastada parameetreid nimeliselt, nn võtmesõnadena (Pythonis on mõlemad võimalused olemas, vt materjali Pythoni funktsioonide kohta).

Moodul, muutujate skoop ja nimeruum

Moodul *(module),* **programmiüksus** *(program unit).* Programmi osa, mis on koostatud eraldatavana või identifitseeritavana kompileerimise, linkimise, täitmise vms seisukohalt ning võib olla interaktsioonis teiste programmide või programmiosadega. Terminiga moodul tähistatav mõiste võib eri programmikeeltes erineda. Pythoni vaates on moodul kogum funktsioonidest, konstantidest, muutujatest jms, mida saab programmis kasutusele võtta käsuga import, samuti peaprogrammi fail. Pythoni moodulist täpsemalt vt materjali Pythoni funktsioonide kohta. Ka teek *(library).*

Nimeruum *(namespace)* – programmis tekkivad muutujad ja nende väärtused. Programmeerimiskeelest sõltuvalt on nimeruumil struktuur ning muutujatel nähtavus ja eluiga.

Nime / muutuja skoop *(scope)* – ka kehtivuspiirkond, osa koodist, milles vastav nimi on seotud tema väärtusega ehk milles (muutuja) deklaratsioon kehtib. Skoobi täpsem määramine varieerub keeleti, kuid on ka üldkehtivad reeglid.

Nimesid võib jaotada kolme rühma:

Mooduli nimed – nimed kõige kõrgemal tasemel – funktsioonide nimed, funktsioonidest väljaspool kirjeldatud muutujate nimed. Mooduli nimed saavad endale väärtused mooduli importimisel või peaprogrammi (Pythoni tähenduses) töö käigus.

Ajutised muutujad – muutujad, mis luuakse funktsiooni sees. Saavad väärtused siis, kui nad funktsiooni väljakutsumise järel luuakse.

Muutuja eluiga *(lifetime)* – aeg programmi töötamisel, millal muutuja on olemas.

Globaalsed ja lokaalsed muutujad *(global and local variables)*

Plokiks *(block)* nnimetatakse mingit osa programmist. See võib olla üks funktsioon, kuid võib ka olla ka üks tsükkel koodis - sõltub keelest. Pythonis võrdsustatakse plokk funktsiooniga. Plokk on oluline mõiste, mille abil määratakse muutujate nähtavus, kehtivuspiirkond, eluiga jms.

Ploki sees deklareeritud muutuja on selle ploki jaoks **lokaalne muutuja**.

Antud plokist kõrgemas plokis deklareeritud muutuja on kõigi sisalduvate plokkide jaoks **globaalne muutuja**. Globaalset muutujat saab tüüpiliselt sisemise ploki / funktsiooni seest

kasutada.

Globaalsete muutujate kasutamist (eriti veel muutmist) funktsioonide seest loetakse halvaks stiiliks, sest sel teel võib kergesti tekkida raskelt avastatavaid vigu.

Globaalsetest muutujatest Pythoni kontekstis saab lugeda Pythoni funktsioonide materjalist.