Programmeerimine

2. loeng

Täna loengus

- Nimed ja muutujad
- Baastüübid
- Litaraalkonstandid
- Omistamine
- Aritmeetilised avaldised
- Funktsioonide rakendamine
- Standardteegid

Muutujad

- Imperatiivses programmeerimises kasutatav arvutusmudel (Von Neumanni arhitektuur) käsitleb programmi instruktsioonide jadana, mis manipuleerib mälus paiknevaid andmeobjekte.
- Igale andmeobjektile vastab mingi mälupiirkond, mille suurus (ja "tähendus") sõltub andmeobjekti tüübist.
- Muutuja on andmeobjekt, millel on nimi (identifikaator).
- Muutujal on kaks väärtust: L-väärtus ja R-väärtus.
- L-väärtus on vastava andmeobjekti aadress ning on konstantne kogu muutuja eluaja.
- R-väärtus on andmeobjektis asuv "tegelik väärtus", mida on võimalik muuta kasutades omistamist.
- NB! Tavaliselt öeldakse muutuja R-väärtuse kohta lihtsalt "muutuja väärtus".

Identifikaatorid

- Identifikaatorid on nimed, mida kasutatakse erinevate programmielementide (näit. muutujad, protseduurid, jne.) tähistamiseks.
  - Identifikaator koosneb tähtedest ja numbritest.
  - Esimene märk peab olema täht.
  - Alakriipsu (_) loetakse täheks.
- Osad nimed, nn. võtmesõnad, on reserveeritud ja neid ei tohi kasutada identifikaatorite nimedena.

and	**as**	**assert**	**break**	**class**	**continue**
def	**del**	**elif**	**else**	**except**	**False**
finally	**for**	**from**	**global**	**if**	**import**
in	**is**	**lambda**	**None**	**nonlocal**	**not**
or	**pass**	**raise**	**return**	**True**	**try**
while	**with**	**yield**			

- Suur- ja väiketähed erinevad.
  - $return \neq Return \neq RETURN$

Identifikaatorid

- Nimi peaks kirjeldama mida see identifikaator tähistab, kuid samas olema siiski kompaktne.

  ✔ *laius*, *pikkus*, *vanus*

- Liiga pikk nimi on raskesti loetav.

  ✘ *selleKarbiLaius*, *Maa_ja_kuu_vaheline_kaugus*

- Väga lühike nimi ei aita seletada tähendust, kuid võib olla mõistlik näiteks lokaalsete muutujate korral.

  – i, j, x, y

- Üldiselt peaks identifikaatori nime valikul põhikriteeriumiks olema programmi loetavus.
- Tihti kasutatakse identifikaatrite nimetamiseks ka kindlaid konventsioone (näit. Ungari notatsioon).

Literaalkonstandid

- Konstantsete baastüüpi väärtuste esitamiseks kasutatakse literaale.
- Enamikes keeltes on võimalik literaalidena esitada:
  - täis- ja ujukomaarve;
  - tähemärke ja sõnesid;
  - tõeväärtuseid.
- Peale ülalmainitute võib keeles olla veel literaale.
  - Pythonis on lisaks võimalik kirjeldada imaginaararve.

Täis- ja ujukomaarvud

- Pythonis on kaks arvutüüpi:

int	suvalise täpsusega täisarvud
float	topelttäpsusega (64-bitised) ujukomaarvud

Täis- ja ujukomaarvud

- Pythonis on kaks arvutüüpi:

int	suvalise täpsusega täisarvud
float	topelttäpsusega (64-bitised) ujukomaarvud

- Täisarvkonstante on võimalik spetsifitseerida kümnend-, kaheksand- ja kuueteistkümnendsüsteemis.

10nd-süsteemis	`0`	`20`	`64`	`-253`
8nd-süsteemis	`0`	`0o24`	`0O100`	`-0o375`
16nd-süsteemis	`0`	`0x14`	`0X40`	`-0xFD`

- NB! Negatiivne konstant (näit. `-253`) ei ole rangelt võttes mitte literaal, vaid operaatori "unaarne miinus" rakendus vastavale literaalile.

Täis- ja ujukomaarvud

- Pythonis on kaks arvutüüpi:

int	suvalise täpsusega täisarvud
float	topelttäpsusega (64-bitised) ujukomaarvud

- Ujukomaarvude täis- ja murdosa on eraldatud punktiga.
- Lisaks võib konstandil olla eksponentosa.

Literaal	`243.12`	`13.`	`2E-3`	`16.1e2`
Tähendus	243.12	13.0	0.002	1610.0
			$= 2 \cdot 10^{-3}$	$= 16.1 \cdot 10^{2}$

Tähemärgid ja sõned

- Sõne on tähemärkide jada ning ei ole lihttüüp.
- Kuid sõnekonstante on võimalik kirjeldada literaalidena.
- Pythonis on sõned tüüpi *str*.
  - Sõned esitatakse kas ülekomade või jutumärkide vahel. Näiteks: `'A'`, `'Hello, World!'`, `"Good bye"`.
  - Lisaks võib kasutada kolmekordseid jutumärke või ülakomasid reavahetustega teksti esitamiseks.
- Mittetrükitavaid ja teisi eriotstarbelisi tähemärke on võimalik esitada kaldkriipsu abil.

`'\''`	ülakoma	`'\"'`	jutumärgid
`'\\'`	kaldkriips	`'\n'`	reavahetus

Tüübid ja tüübiteisendused

- Iga objekti tüüpi on võimalik küsida funktsiooniga *type*.

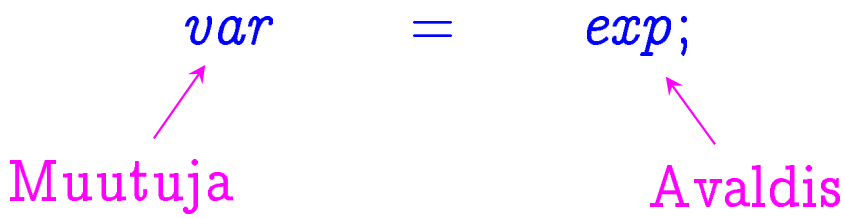
$$\begin{aligned} \mathit{type}(5) &\Longrightarrow \texttt{<class 'int'>} \\ \mathit{type}(5.1) &\Longrightarrow \texttt{<class 'float'>} \\ \mathit{type}(\texttt{'5.1'}) &\Longrightarrow \texttt{<class 'str'>} \end{aligned}$$

- Iga tüübiga on seotud samanimeline tüübiteisendusfunktsioon, mis üritab argumendi teisendada vastavasse tüüpi.

$$\begin{aligned} \mathit{float}(\texttt{'5.1'}) &\Longrightarrow 5.1 \\ \mathit{str}(5.1) &\Longrightarrow \texttt{'5.1'} \\ \mathit{int}(5.1) &\Longrightarrow 5 \end{aligned}$$

Omistamine

- Muutuja väärtuse muutmiseks saab kasutada omistuslauset:



- Omistuslause vasakpool on muutuja, parempool avaldis.
- Omistuslause täitmisel väärtustatakse lause parempool ning selle väärtus salvestatakse vasakpooles oleva muutuja poolt viidatavasse mäluvälja.
- NB! Vasakpooles oleva muutuja varasem väärtus hävib; ta kirjutatakse üle uue väärtusega.
- Pythonis võib semikooloni omistuslause lõpust ära jätta.
- Enamik keeltes tuleb muutuja enne kasutamist deklareerida; Pythonis pole seda vaja teha ning muutuja luuakse temale esmakordsel omistamisel.

Omistamine

$$x = 3$$
$$x = 7$$
$$x = x + 1$$
$$y = x$$
$$x = 9 - y$$

Omistamine

$$x = 3$$
$$x = 7$$
$$x = x + 1$$
$$y = x$$
$$x = 9 - y$$

x | 3

Omistamine

$$x = 3$$
$$x = 7$$
$$x = x + 1$$
$$y = x$$
$$x = 9 - y$$

x | 7

Omistamine

$$x = 3$$
$$x = 7$$
$$x = x + 1$$
$$y = x$$
$$x = 9 - y$$

x | 8

Omistamine

$$x = 3$$
$$x = 7$$
$$x = x + 1$$
$$y = x$$
$$x = 9 - y$$

x	8
y	8

Omistamine

$$x = 3$$
$$x = 7$$
$$x = x + 1$$
$$y = x$$
$$x = 9 - y$$

x	1
y	8

Aritmeetilised avaldised

- Aritmeetilised avaldised koosnevad muutujatest, konstantidest, operaatoritest ja funktsioonidest.
- Igal avaldisel on väärtus, mis arvutatakse "seest väljapoole".
- Õige arvutusjärjekorra määramiseks on operaatoritega seotud prioriteedid ja assotsiatiivsusreeglid.
- Kõrgema prioriteediga operaatorid arvutatakse enne väiksema prioriteediga operaatoreid.

$$3 + 5 * 2 \implies 13$$

- Assotsiatiivsus määrab kas sama prioriteediga operaatorid avutatakse vasakult paremale või vastupidi.

$$5 - 2 - 2 \implies 1$$

- Vaikimisi reegleid saab muuta sulgude abil.

$$(3 + 5) * 2 \implies 16$$
$$5 - (2 - 2) \implies 5$$

Aritmeetilised avaldised

Aritmeetilised operaatorid Pythonis

Operaator	Kirjeldus
`**`	astendamine
`+`, `-`	pluss, miinus (unaarsed)
`*`, `/`, `//`, `%`	korrutamine, jagamine, täisarvuline jagamine, jääk
`+`, `-`	liitmine, lahutamine

- Peale astendamise on kõik ülaltoodud operaatorid vasakassotsiatiivsed.
- Astendamine on paremassotsiatiivne.

Aritmeetilised avaldised

- Aritmeetiliste operaatorite argumendid peaksid reeglina olema sama arvutüüpi.
- Kui argumendid on erinevat tüüpi, siis teisendatakse need automaatselt sobivasse arvutüüpi.
- Tüübiteisendamist on võimalik teha ka ilmutatult tüübiteisendusfunktsioonide abil.

$$\begin{aligned} \mathit{float}(12) &\Longrightarrow 12.0 \\ \mathit{int}(3.14) &\Longrightarrow 3 \end{aligned}$$

- NB! Liitmine ja korrutamine on lisaks arvudele defineeritud ka sõnedel.

```
'Hello' + ' World!'  ⟹  'Hello World!'
'Hello' * 3          ⟹  'HelloHelloHello'
```

Omistamine ja aritmeetilised operaatorid

Kombineeritud omistamine

Lühend	Tähendus
$x \mathrel{+}= 3$	$x = x + 3$
$x \mathrel{-}= 1$	$x = x - 1$
$x \mathrel{*}= y + 1$	$x = x * (y + 1)$
$x \mathrel{/}= 3$	$x = x / 3$
$x \mathrel{//}= 2$	$x = x // 2$
$x \mathrel{\%}= 2$	$x = x \% 2$
$x \mathrel{**}= 4$	$x = x ** 4$

Funktsioonide rakendamine

- Funktsiooniaplikatsioon Pythonis:

$$funName(arg_1, \ldots, arg_n)$$

- $funName$ on funktsiooni nimi.
- arg_i on funktsiooni argumendid.
  - Avaldised, mille väärtused antakse funktsioonile edasi.
  - Sõltuvalt funktsioonist, võib tal argumente olla üks või mitu, või ka olla ilma argumentideta.
- Funktsiooni rakendamise tulemuseks on mingi väärtus.
  - "Ilma väärtuseta" funktsioonide väärtuseks on `None`.

Kasulikke eeldefineeritud funktsioone

Eeldefineeritud funktsioone

Funktsioon	Kirjeldus
abs(n)	arvu absoluutväärtus
round(x,n)	arvu ümardamine etteantud täpsuseni
len(s)	leiab järjendi pikkuse
input(s)	küsib kasutajalt sisendit
print(x_1,...,x_n)	trükib argumendid ekraanile
exit()	lõpetab programmi töö
help(o)	väljastab objekti abiinfo

Standardteegid

- Palju kasulikke funktsioone on Pythonis eeldefineeritud erinevates standardteekides.
  - teek (ingl. *library*) = moodul
- Mooduli kasutamiseks tuleb ta enne *importida*:

import *mname*

- Peale mooduli importimist saab tema atribuute (so. temas defineeritud funktsioone ja muutujaid) kasutada "punkt-notatsioonis": *mname.atrib*.
  - Moodulis defineeritud atribuutide loendi väljastab funktsioon *dir*(*mname*).

Standardteegid

- Moodulis defineeritud identifikaatoreid on võimalik importida ka otse:

  from *mname* **import** *names*

- *names* on imporditavate identifikaatorite loend (eraldatud komadega) või tärn (kõigi moodulis olevate nimede importimiseks).

Standardteegid

Kasulikke standardteeke

Moodul	Teegi kirjeldus
turtle	kilpkonnagraafika
math	matemaatilised funktsioonid
string	sõnetöötlusfunktsioonid
random	juhuarvude genereerimise funktsioonid

Järgmiseks korraks

- Lugeda läbi õpiku peatükid:
  - (Ptk. 2 *"Avaldised ja lihtlaused"*)
  - Ptk. 3 *"Tingimus- ja korduslaused"*
- Kui lugemisel tekkis küsimusi, mille kohta soovite järgmises loengus vastust, siis need võib saata hiljemalt esmaspäeva lõunaks mailiga varmo.vene@ut.ee ja/või helle.hein@ut.ee.

Suur tänu osalemast

ja

kohtumiseni!