

LEVINUIMAD KOODIELEMENDID

Matemaatilised operatsioonid

`sum()` # leiab summa

`average()` # leiab keskmise

`abs()` # leiab absoluutväärtuse

`sqrt()` # võtab ruutjuure

`sin()` # leiab siinuse

`cos()` # leiab koosinuse

`pi` # leiab ringi diameetri suhte ringi ümbermõõtu ehk Pii

Järgmises näites kasutatakse järjendit `juku`, mis on defineeritud järgmiselt: `juku = [2, 1, 5, 6, 1, 9]`

`juku.count(1)` # loeb kokku, mitu korda arvu üks järjendis esineb – vastuseks on 2, sest 1 esineb järjendis `juku` 2 korda; arvu võib asendada mõne teise arvu, sõne, järjendi vms

Järjendid ja nende organiseerimisega seotud meetodid

`shuffle()` # ajab järjendis olevad elemendid segamini

Järgmises näites kasutatakse järjendit `juku`, mis on defineeritud järgmiselt: `juku = [2, 1, 5, 6, 1, 9]`

`juku.sort()` # järjestab järjendi elemendid väiksemast suuremaks

`juku.append()` # lisab järjendi lõppu sulgudesse sisestatud arvu; `juku.append(8)` lisab järjendi lõppu kaheksa

`print(juku(-1))` # prindib järjendi viimase elemendi

`range()` # vaikimisi annab järjendi, mis sisaldab arve nullist kuni sulgudesse sisestatud väärtuseni miinus üks; nt `range(6)` annab `[0,1,2,3,4,5]` – sama meetod lubab sisestada ka algus ja lõpp-punkti ning sammu, millega lõppu peaks jõudma; nt `range(2, 32, 10)` annab vastuseks `[12, 22]`

Juhuslikustamisega seotud meetodid

`random()` # leiab juhusliku arvu nullist üheni selliselt, et iga nulli ja ühe vahele jääva väärtuse tõenäosus on sama (`numpy.random()`) – sisendina võib anda ka arvu, mitut juhuslikku arvu peaks meetod andma (vaikimisi üks)

`random.normal()` # leiab juhusliku arvu selliselt, et keskmine on 0 ja standardhälve 1; funktsioon võtab sisendina prinditava järjendi suuruse – nt kui kirjutada sulgudesse size (1, 10), siis annab meetod järjendi, milles on üks rida ja kümme veergu. NB! Selleks, et meetod PsychoPys tööle läheks, siis tuleks esmalt PsychoPy keskkonda laadida ka vajaminev moodul. Seda saab teha järgmise käsuga: `from numpy import random`

`t` # aeg rutiini algusest sekundites

`globalClock.getTime()` # aeg, mis on katse algusest alates möödunud

`expInfo['participant']` # süntaks infokasti sisestatud väärtuste leidmiseks – antud juhul leitakse 'participant' lahtrisse sisestatud väärtus.

expInfo-sse sisestatud väärtused kodeeritakse ennikutena (tuple). Muutuja tüübi nägemiseks saab kasutada meetodit `type()`:

`print(type(expInfo['participant']))`

`".join(muutuja_nimi)` # enniku sõneks muutmiseks kasutatav meetod

`float (muutuja_nimi)` # enniku ujukomaarvuks muutmiseks kasutatav meetod

Infokasti sisestatut saab kasutada ka tekstikasti sisendina. Näiteks oletame, et infokasti “participant” lahtrisse sisestati Juku. Kui tekstikasti kirjutada järgnev süntaks:

`$'Tere, ' + expInfo['participant'] + '!',`

siis prinditakse järgmine kiri: Tere, Juku!. NB! Dollari märk süntaksi ees ütleb PsychoPy-le, et sisestatud teksti tuleks tõlgendada kui Pythoni koodi. Jutumärgid teksti ümber viitavad PsychoPy-le, et tegemist on sõnega. `expInfo['participant']` ümber jutumärke panna ei tohiks, sest sellisel juhul peab PsychoPy ka seda lihtsalt tekstiks, kuid tegelikult viitame siin `expInfo` nime kandvale Pythoni struktuurile, mida kutsutakse sõnaraamatuks.

Enamasti aga sooviksime lisaks tervitusele veel midagi lisada. Näiteks, et jätkamiseks tuleks vajutada tühikuklahvi. Üks võimalus, kuidas seda teha oleks eelneva süntaksiga lisada lause, mis seda ütleks:

`$'Tere, ' + expInfo['participant'] + '!' + 'Jätkamiseks vajuta palun tühikut...'`

kuid see ei ole enamasti väga hea lahendus (eriti kui tegemist on pikema tekstiga), sest tekst prinditakse ühes jorus. Õnneks on Pythonis olemas süntaks, mis võimaldab järgmiselt realt alustada või ridu vahele jätta. Järgmiselt realt alustamiseks tuleks järgmisele reale kirjutatava teksti ette kirjutada `\n`. Näiteks kui sooviksime, et klahvivajutusest rääkiv tekst oleks ülejärgmisel real (ehk rea vahele jätta) võiksime kirjutada:

`$'Tere, ' + expInfo['participant'] + '!' + '\n\nJätkamiseks vajuta palun tühikut...'`

Pythoni keeles on funktsioon `dir()`, mis aitab uurida, millised atribuudid mingil muutujal on.
Näide: `print(dir(expInfo))`

`frameN` # ütleb mitu värskendust on rutiini algusest möödunud (NB! Esimesele värskendusele vastab 0, teisele 1, kolmandale 2 jne)

`if frameN % 5 == 0:`
..... # punktiir tuleks asendada süntaksiga, mida sooviksid igal viiendal värskendusel joosutada. Modulo operatsioon jagab kaks arvu teineteisega ja tagastab jagamisel saadud jäägi. Näiteks `5 % 2` on võrdne ühega, sest kaks mahub viie sisse kaks korda ja jääk on üks.

`continueRoutine=False` # see süntaks lõpetab jooksva rutiini

`trials.finished=True` # lõpetab jooksvate seeriade esitamise – “trials” viitab siin rutiinide ümber defineeritud aasale, mille nimi vaikide on “trials” – kui seda nime muuta, siis tuleks nimetus asendada ka süntaksis.

`stim.status==FINISHED` # kontrollib, kas komponent “stim” on töö lõpetanud – komponendi staatus võib olla veel alustatud (STARTED) ja mittealustatud (NOT_STARTED). NB!
Kindlasti tuleks siin kaks võrdusmärki lisada, sest kontrollime võrdsust.

`randint(minimaalne_väärtus, maksimaalne_väärtus_pluss_üks)` # võtab sisendina kaks arvu ja annab juhusliku täisarvu, mis jääb esimese ja teise sisestatud arvu vahele selliselt, et maksimaalseks väärtuseks on teisena sisestatud väärtus miinus üks.

`thisExp.addData('veerg', 'väärtus')` # see süntaks võimaldab meil andmetabelisse veerge lisada ja sinna veergu väärtusi printida; näiteks süntaks `thisExp.addData('cue', 'vasak')` tekitab uue veeru (kui seda juba loodud ei ole) ja kirjutab sinna sõna vasak. Andmetabelisse kirjutatav väärtus võib olla ka numbriline – näiteks juhuslikult genereeritud seeriavaheline aeg, mille soovid igal seerial andmetabelisse kirjutada.

Võimalik süntaks pausi esitamiseks. Lisa see süntaks koodielemendile, mis paikneb samas rutiinielemendis, kuhu salvestasid ka pausi teksti ja vastuseklahvi komponendid.

Selgituseks süntaksile: Kõigepealt kontrollib, kas esituse järjekorranumber on suurem kui 0 ehk kas tegemist on esimese esitusega. Kui mitte, siis kontrollib, kas tegu on kolmekümnnenda esitusega. Kui vastus on eitav (ehk `trials.thisN` ei jagu täpselt kolmekümnega, siis peatab rutiini). Vastasel korral jätkab rutiini ja näitab pausi.

`if trials.thisN > 0 and trials.thisN % 30 != 0:`

`continueRoutine = False` # peatab rutiini

`totalScore = score.count(1)` # loendab, mitu korda 1 järjendis score esineb