

PRAKTIKUM 7: Posneri katse

Selles praktikumis loome Posneri katse.¹ Õpime stiimulite kestusi värskenduste kaudu defineerima ja lisame katsele ka treeningseeriad. Lisaks liidame katsega ka pausi, mida esitatakse ainult teatud arvu esituste järel. Vaatleme millised võimalused on PsychoPy-s esituste järjestuse juhuslikustamiseks ehk randomiseerimiseks.

Meenutuseks: visuaalsete stiimulite kettalt lugemine võtab aega (eriti kui tegu on kõrge pikslitihedusega piltidega), visuaalne stiimul tuleks sünkroniseerida ekraani värskendamisega, lühikese kestusega stiimulite esitamisel on kasulik kasutada ekraanivärskenduste arvu, klaviatuuri ja hiire vastused varieeruvad ning ideaalis oleks hea nende ajastust kontrollida.

Kõrget ajalist täpsust nõudvate ja lühikese kestusega visuaalsete stiimulite ajastamisel on eelistatum kasutada **ekraanivärskenduste arvu** ja nii ka Posneri katse puhul. Monitor esitab visuaalseid stiimuleid täisarvuks värskendusteks ekraanile ja kõrget ajastustäpsust nõudvates katsetes tuleks seda arvesse võtta. Kui näiteks monitori värskendussageduseks on 60 Hz (tüüpilise LCD monitori värskendussagedus), siis leiame kõik võimalikud esituste ajad järgmiselt $t = N/60$, kus N vastab värskenduste arvule ja t stiimuli kestusele sekundites. Näiteks kui esitaksime stiimuli ekraanile kaheteistkümneks värskenduseks, siis saaksime, et selle kestuseks oleks 0,2 s ehk $t = 12/60$. Kuna ühe värskenduse kestus on 1/60 ehk umbes 16,66(6) ms, siis järgmisena kasutatav kestus on 13 värskendust ehk 0,216(6). Kui aga defineerime stiimuli kestuse sekundites, siis lubab PsychoPy kestuseks valida mistahes arvu, kuid kahjuks ei tähenda see seda, et stiimuli sekundites defineerimine oleks täpsem kui ekraanivärskenduste kaudu defineerimine. Sõltumata kestuse defineerimise viisist esitatakse stiimuleid ekraanile kindlaks arvuks värskendusteks. Näiteks kui kirjutame, et soovime esitada ekraanile stiimuli 0,210 sekundiks, siis esitatakse stiimul ekraanile 0,216 sekundiks, mis on võrdne 13 värskendusega, sest värskendusi saab olla, kas 12 või 13, kuid mitte 12,6 nagu 0,210 sekundiks esitamise puhul tarvis läheks.

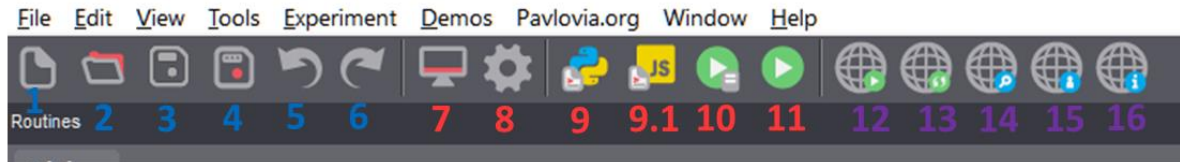
Posneri ruumitähelepanu katses instrueeritakse katseisikuid hoidma pilku ekraani keskel paikneval fiksatsiooniristil ja raporteerida kummale poole, vasakule või paremale, sihtstiimul (*target*) ilmus. Enne sihtstiimuli ilmutist esitatakse katseisikule vihje, mis viitab kummale poole sihtstiimul ilmub. Enamikel juhtudel (80%) langeb vihje suund ja sihtstiimuli asukoht kokku, kuid väiksel arvul seeriast tüsib vihje vastajat ja sihtstiimul esitatakse hoopis teisele poole (20%). Varasemad tulemused näitavad ootuspäraselt, et katseisikute vastused on enamasti kiiremad siis, kui vihje suund ja sihtstiimuli asukohta kokku langevad, kuid huvitaval kombel toimib see selliselt ainult juhul kui vihje esitatakse kuni 300 ms enne sihtstiimuli ilmutist. Kui vihje ilmub varem, siis on katseisikute vastused hoopis aeglasemad.

¹ Vaata ka, kuidas PsychoPy looja, professor Jonathan Pierce, Posneri katse loomist demonstreerib: https://www.youtube.com/watch?v=ZQd2QEK_Gn4

Ülesanne 0. Loo uus kaust, kuhu eksperimendi juhtfaili ja tingimuste failid saaksid paigutada.

Ülesanne 1. Muuda monitori sätteid selliselt, et need vastaksid sinu monitori parameetritele.

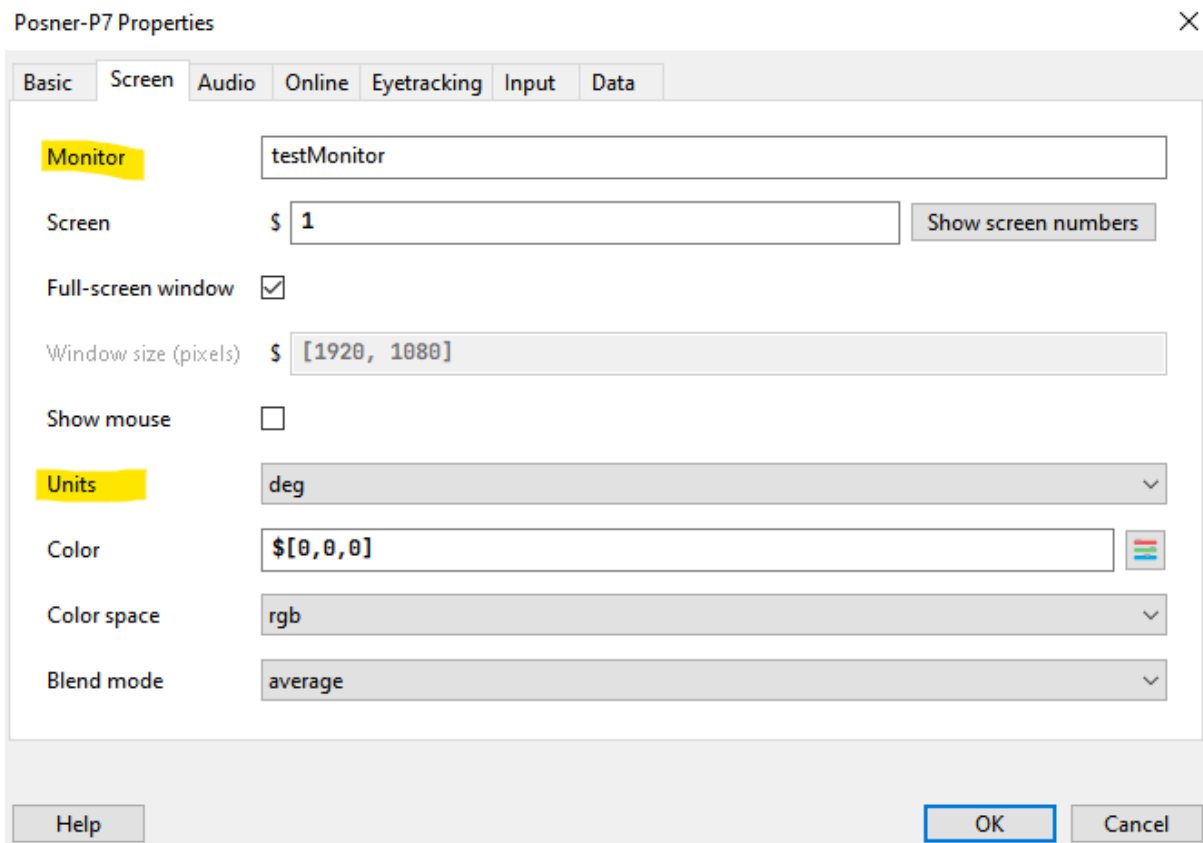
Seadete muutmiseks klõpsa ikoonide menüüs monitori pildiga ikoonil (7) ja lisa monitori kaugus katseisikust (nt 57 cm), ekraani suurus pikslites (nt 1920 x 1080) ning ekraani laius (nt 30 cm).



Ülesanne 2. Muuda eksperimendi seadeid selliselt, et PsychoPy kasutaks stiimulite esitamisel eelmises ülesandes muudetud monitori seadeid (nt testMonitor). Lisaks muudame seadeid selliselt, et eksperiment kasutaks stiimulite esitamiseks vaikimisi nägemisnurga kraade.

Eksperimendi seadete muutmiseks vajutame mutriga ikoonil (8) ja avame ekraani seadete saki (**Screen**), milles kirjutame esimesse lahtrisse (**Monitor**) eelmises ülesandes salvestatud monitori seadete nime.

Selleks, et eksperiment kasutaks vaikimisi nägemisnurga kraade vali eksperimendi seadete all ühikuteks (**Units**) deg ehk kraadid.



Ülesanne 3. Vali komponentide menüüst visuaalne stiimul, millest saab katses kasutatav vihjav kolmnurk (*Polygon*), anna sellele nimeks *cue* ja lisa see *trial* rutiinielemendi koosseisu. Muuda selle komponendi järgnevaid parameetreid:

- sea algushetkeks rutiinielemendi algus ehk 0 ja defineeri selle kestus ekraanivärskendustes, andes sellele väärtuseks 12
- määra komponendi suuruseks 2 x 4 kraadi
- kuigi see ei oma funktsionaalset tähtsust, siis võid komponendi oodatud kestuseks (Expected duration (s)) määrata 12/60 ehk 0,2 sekundit – sellisel juhul kuvab komponendi kestuse ka graafilises liideses, kuid selle mõju on ainult graafilise liidese visuaalile ja stiimuli tegeliku kestuse määrab ikka lahtrisse **duration (frames)** kirjutatu

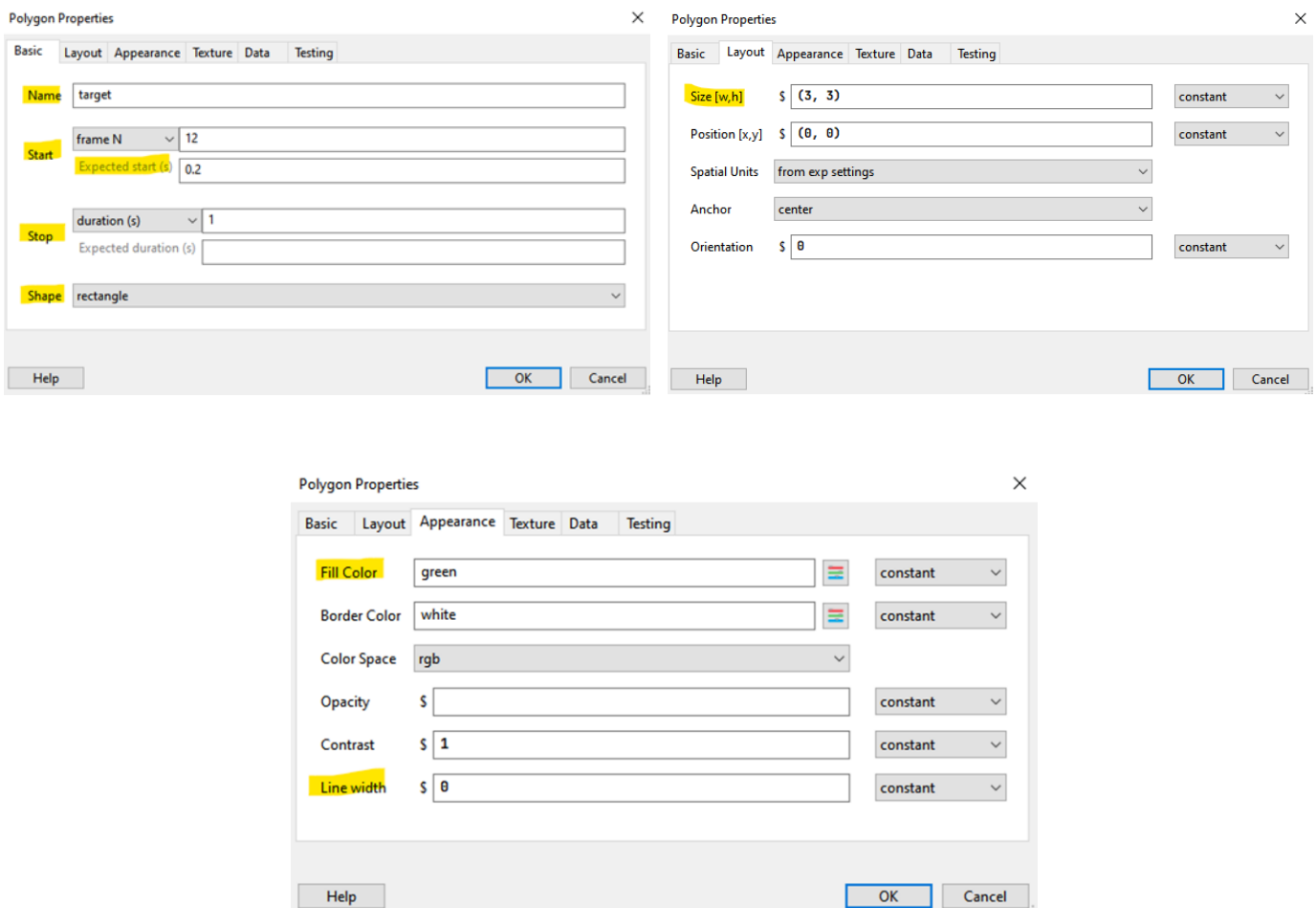
The screenshot shows the 'cue Properties' dialog box with the 'Basic' tab selected. The 'Name' field contains 'cue'. Under the 'Start' section, 'time (s)' is set to 0.0. Under the 'Stop' section, 'duration (frames)' is set to 12, and 'Expected duration (s)' is set to 0.2. The 'Shape' is set to 'triangle' and 'Num. vertices' is set to 4. The 'constant' dropdowns for 'Shape' and 'Num. vertices' are also visible.

Kuna komponendi kujuks on juba vaikselt kolmnurk, siis seda eraldi muuta ei ole tarvis. Muuta ei ole vaja ka komponendi suurusühikuid, sest muutsime need juba eksperimendi seadete juures kraadideks ja vaikselt pärivad komponendid enda ühikud sealt (vt alumisel joonisel *from exp settings*).

The screenshot shows the 'cue Properties' dialog box with the 'Layout' tab selected. The 'Size [w,h]' is set to (2, 4), 'Position [x,y]' is set to (0, 0), 'Spatial Units' is set to 'from exp settings', 'Anchor' is set to 'center', and 'Orientation' is set to 0. The 'constant' dropdowns for 'Size [w,h]', 'Position [x,y]', and 'Orientation' are also visible.

Ülesanne 4. Vali komponentide menüüst visuaalne stiimul, millest saab katses kasutatav sihtstiimul (*Polygon*), anna sellele nimeks *target* ja lisa see *trial* rutiinielemendi koosseisu. Muuda selle komponendi järgnevaid parameetreid:

- sea algushetkeks vihjava stiimuli lõpp ehk 12 värskendust
- määra komponendi suuruseks 3 x 3 kraadi
- määra komponendi värvuseks roheline ja anna kasti ümbritseva raami suuruseks 0, mis tagab, et kasti ümber raami ei esitataks
- kuigi see ei oma funktsionaalset tähtsust, siis võid komponendi oodatud algusajaks (Expected duration (s)) määrata 12/60 ehk 0,2 sekundit – sellisel juhul kuvatakse komponendi kestus ka graafilises liideses, kuid selle mõju on ainult graafilise liidese visuaalile ja stiimuliku tegeliku kestuse ja algusaja määrab vastavalt **duration** ja **frame N** lahtritesse kirjutatu



Ülesanne 5. Vali komponentide menüüst vastuseklahv, anna sellele nimeks *resp* ja lisa see *trial* rutiinielemendi koosseisu. Muuda selle komponendi järgnevaid parameetreid:

- sea algushetkeks sihtstiimuli (*target*) ekraanile ilmumise aeg: 12 värskendust (oodatud algusajaks võime siingi kirjutada 0,2 sekundit); kuna soovime, et katseisikud vastuse üle pikalt mõtlema ei jääks, siis seame komponendi kestuseks 2 sekundit, mis tagab selle, et katse läheb edasi, kui kahe sekundi jooksul vastust ei registreeritud
- sea lubatud klahvivajutusteks vasak ja parem

resp Properties

Basic Data Testing

Name: resp

Start: frame N 12
Expected start (s) 0.2

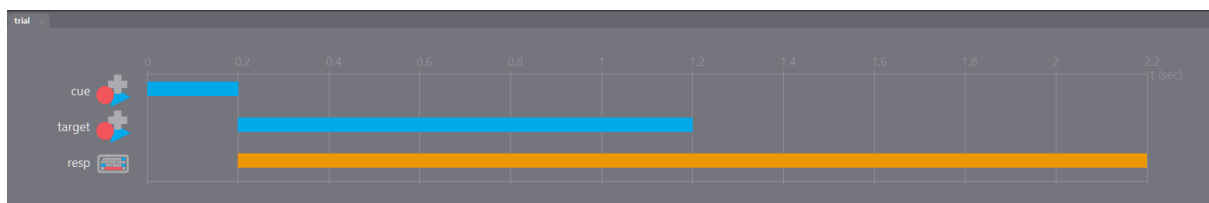
Stop: duration (s) 2
Expected duration (s)

Force end of Routine ☒

Allowed keys: \$ 'left', 'right' constant

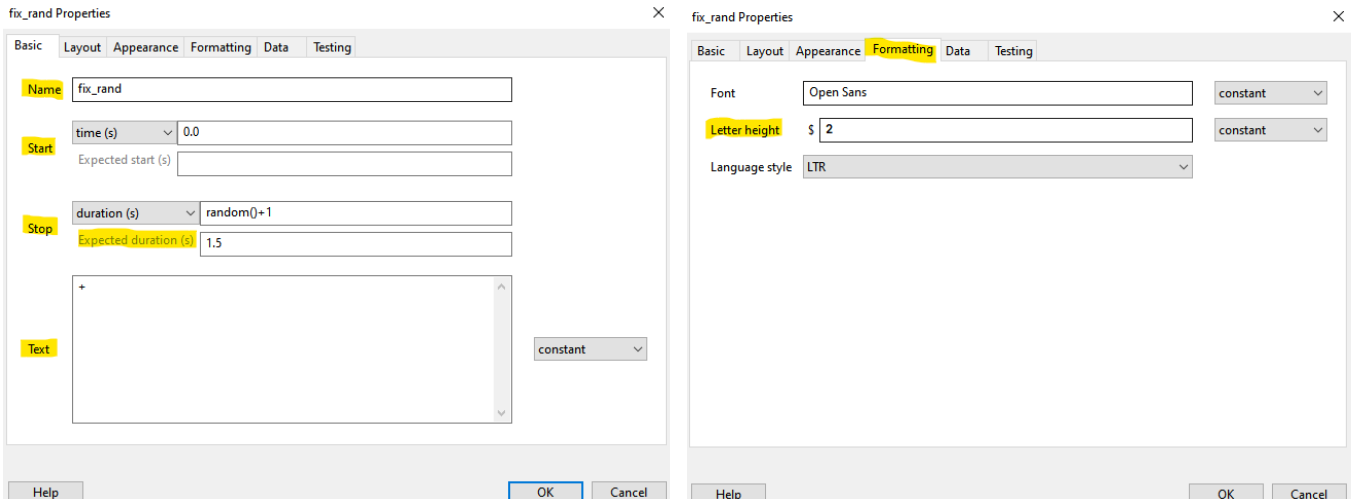
Help OK Cancel

Pärast neid muudatusi peaks *trial* rutiin välja nägema umbes selline:



Ülesanne 6. Järgmiseks liidame katsele rutiinielemendi, anname sellele nimeks ITI (*inter trial interval*) ja liidame selle koosseisu teksti komponendi (Text) ja muudame selle järgmisi parameetreid:

- anname komponendile nimeks *fix_rand*
- rutiinikomponendi sisselülitamise ajaks määrame (0) ja pane selle kestuse ühest kahe sekundini juhuslikult varieeruma – selleks kirjuta `duration` lahtrisse `1+random()` (oodatud algusajaks võime siin kirjutada 1.5, sest keskel läbi on seeriade vaheliseks ajaks 1.5 sekundit)
- sisesta tekstikasti plussmärk (+)
- määra teksti kõrguseks 1 kraad



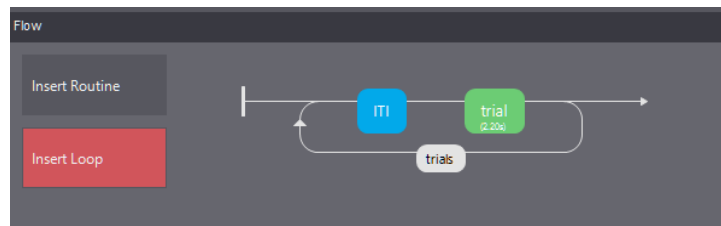
Kui sa pole veel proovinud katset käivitada, siis tee seda kindlasti enne järgmiste ülesannete juurde minemist. Kui oled veendunud, et kõik toimib, siis liigu järgmiste ülesannete juurde.

Ülesanne 7. Loo tingimuste fail, mille veerud defineerivad vihje poole (cueSide), kaldenurga (cueOri), sihtstiimuli x-koordinaadi (targetX), õige vastuse klahvi (corrAns) ja vihje valiidsuse (valid). Loo tingimuste fail selliselt, et pooltel juhtudel esitataks sihtstiimul vasakule ja pooltel paremale (sh mittevaliidsel seerialtel), ning et kahekümnel protsendil juhtudest viitaks vihje valele poole.

Minimaalne tingimustabeli ridade arv, mis neid eeldusi rahuldab on 10, sest ühel mittevaliidsel juhul peaks vihje näitama vasakule ja teisel juhul paremale, mis on kokku kaks rida. Kui tahame, et need esitused oleksid 20% ülejäänud esitustest, siis peame juurde lisama 8 rida, mis annab tingimuste tabeli ridade koguarvuks kümme.

	A	B	C	D	E	F
1	cueSide	cueOri	targetX	corrAns	valid	
2	left	-90	-7	left	1	
3	left	-90	-7	left	1	
4	left	-90	-7	left	1	
5	left	-90	-7	left	1	
6	right	90	7	right	1	
7	right	90	7	right	1	
8	right	90	7	right	1	
9	right	90	7	right	1	
10	right	90	-7	left	0	
11	left	-90	7	right	0	
12						

Ülesanne 7. Seejärel liida *ITI* ja *trial* rutiinielementide ümber aas (Insert Loop) ja lisa sellele loodud tingimuste tabel (nt conditions.xlsx)



trials Properties

Name: trials

loopType: random

Is trials: ☒

nReps: 5

Selected rows:

random seed:

Conditions: conditions.xlsx

10 conditions, with 5 parameters
[cueSide, cueOri, targetX, corrAns, valid]

Help OK Cancel

Ülesanne 7. Järgmiseks muuda *trial* rutiinielemendi koosseisus olevate vihje, sihtstiimuli ja vastuse komponendi seadeid selliselt, et neis võetaks arvesse tingimuste tabelis defineeritud väärtused.

cue Properties

Basic Layout Appearance Texture Data Testing

Size [w,h]: (2, 4) constant

Position [x,y]: (0, 0) constant

Spatial Units: from exp settings

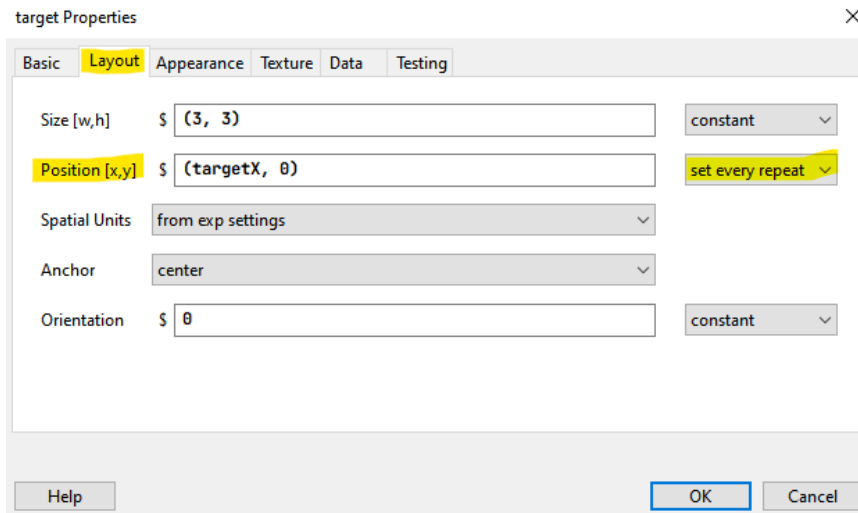
Anchor: center

Orientation: cueOri set every repeat

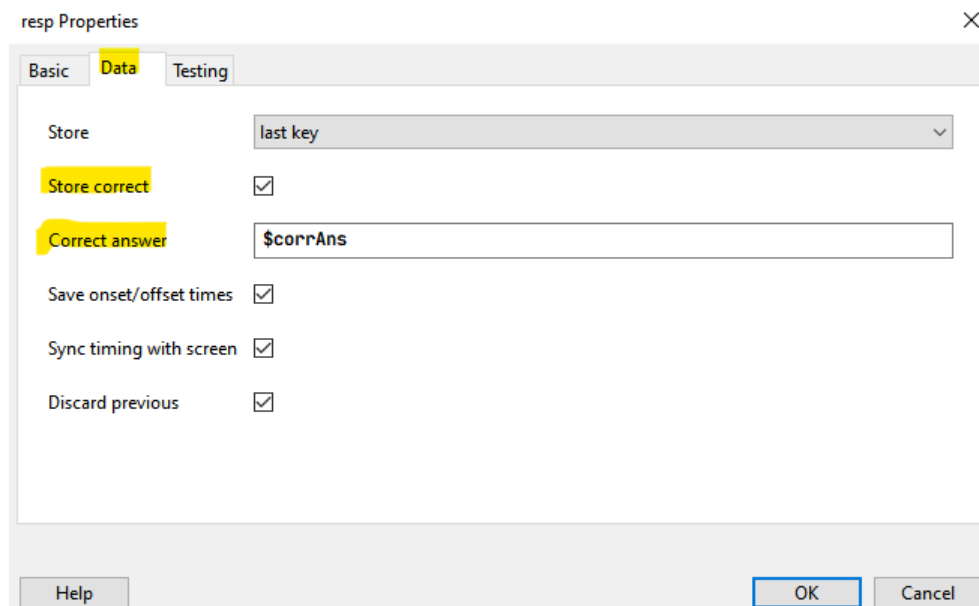
Help OK Cancel

Vihje puhul muudame igal esitusel vihje kaldenurka ehk seda, kas vihje näitab vasakule või paremale. Selle muudatuse tegemiseks klõpsa vihje komponendil (cue) ja kirjuta lahtrisse **Orientation** veeru nimi, kuhu vihje kaldenurga kirjutasid (cueOri). Pane kaldenurk kindlasti ka igal esitusel varieeruma (*set every repeat*).

Sihtstiimuli muutmiseks klõpsa komponendil *target*, ava **Layout** sakk ja kirjuta sihtstiimuli asukohta defineerivasse lahtrisse (**Position**) x-koordinaadi kohale veeru nimetus, kuhu kirjutasid sihtstiimuli x-koordinaadid (*targetX*). Pane kindlasti ka komponendi positsioon igal esitusel varieeruma (*set every repeat*).



Vastuse klahvi muutmiseks klõpsa vastuse (*resp*) komponendil ja lisa **Data** saki all linnuke kastikesse, mis lülitab vastuste õigsuse kontrollimise sisse (**Store correct**). Selle all paiknevasse lahtrisse (**Correct answer**) kirjuta veeru nimi, kuhu salvestasid õigete klahvivajutuste nimetused ja lisa selle ette ka dollari märk (*\$corrAns*).

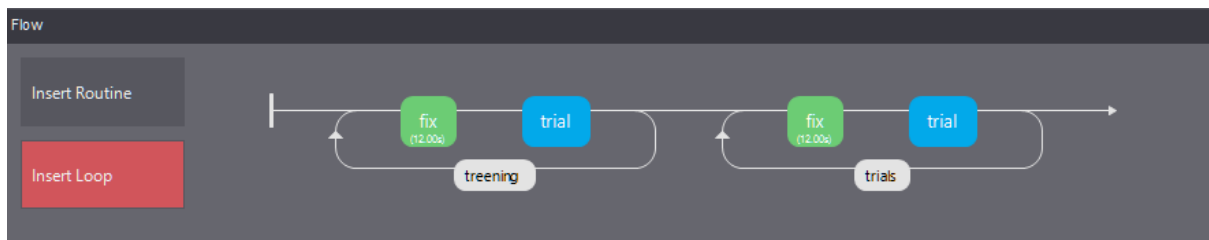


Ülesanne 8. Enamasti tahame katsele lisada ka treeningseeriad, mille käigus on katseisikul võimalik katsega tutvuda. Treeningseeriade lisamiseks saame kasutada pesastatud süntaksit. Selleks loome uue tingimuste tabeli (nt *training.xlsx*) ja lisame sellesse tabelisse esitused, mida soovime, et katseisik treenida saaks. Antud näite kontekstis võib olla kasulik demonstreerida, et vihje ei ole alati korrektne.

Oletame, et lisame treeningseeriade tingimuste faili neli veergu, millest kahel juhul on vihje valiidne ja kahel mittevaliidne. Sellisel juhul näeks meie tabel välja selline:

	A	B	C	D	E	F
1	cueSide	cueOri	targetX	corrAns	valid	
2	right	90	7	right	1	
3	left	-90	-7	left	1	
4	right	90	-7	left	0	
5	left	-90	7	right	0	
6						
7						

Üks viis, kuidas katsele treeningseeriaid lisada on korrata samu rutiinielemente kaks korda ja lisada kummaldi korral vastav tingimuste tabel (nt training.xlsx või conditions.xlsx) ehk nii nagu näidatud sellel joonisel:



Kuigi saab ka nii, siis eelistatum on kasutada teist lahendust, mis ei nõua samade rutiinielementide taaskasutamist.

Ülesanne 8.1. Loo *trials* aasa ümber veel üks aas, mis esitaks kõigepealt kõik treeningseeriade tabelisse (training.xlsx) lisatud read ja seejärel kordaks kõiki põhikatsse tingimustabeli ridu (conditions.xlsx).

Selleks tekitame *trials* aasa ümber veel ühe aasa (*Insert Loop*), millele võime anda nimeks *blocks*. See aas määrab ära selle, kumba tingimuste tabelit parasjagu kasutatakse. Tingimuste tabelite nimed salvestame kolmandasse tabelisse (nt blocks.xlsx), kuhu kirjutame eraldi reale katses kasutatavate tingimuste failide nimed (condFile). Lisaks saame sinna sisestada veeru, mis defineeriks, mitu korda tingimuste faili katse jooksul esitatakse (nRep).

	A	B	
1	condFile	nRep	
2	training.xlsx	1	
3	conditions.xlsx	5	

Seadistame aasa seaded selliselt, et ta loeks sisse eelnevalt loodud tingimuste faili blocks.xlsx. Lisaks muudame juhuslikustamise tüüpi. Vaikimis on loopType järel valitud juhuslikustamise tüübiks random, mis esitab meie poolt valitud tingimustefaili nReps korda. Antud juhul tahaksime, et programm valiks kõigepealt ühe (training.xlsx) ja siis teise tingimustefaili (conditions.xlsx) ja seega valime juhuslikustamise tüübiks *sequential* ehk järjest esitamise. Lisaks võime ära võtta linnukese „Is trials“ eest, mis annab PsychoPyle märku, et tegemist ei ole seeriatega.

blocks Properties

Name: blocks

loopType: sequential

Is trials: ☐

nReps: \$ 1

Selected rows:

random seed: \$

Conditions: blocks.xlsx

2 conditions, with 2 parameters [condFile, nRep]

Help OK Cancel

Kui need muudatused on tehtud, siis avame *trials* aasaga seotud seaded ja kirjutame **nReps** lahtrisse veeru nime, milles defineerisime, mitu korda sama tingimuste faili tuleks korrata (nRep). Seejärel kirjutame **Conditions** lahtrisse veeru nime, kuhu salvestasime tingimuste failide nimetused (condFile). Lisame veeru nimetuse ette ka dollari märgi, sest PsychoPy peab meie poolt sisestatud sõna vaikumisi tingimuste faili nimetuseks, kuid antud hetkel viitame hoopis veerule tingimuste tabelis. nResp puhul ei pidanud me seda tegema, sest selle veeru ees on juba vaikumisi dollari märk (vt alumine joonis).

trials Properties

Name: trials

loopType: random

Is trials: ☒

nReps: \$ nRep

Selected rows:

random seed: \$

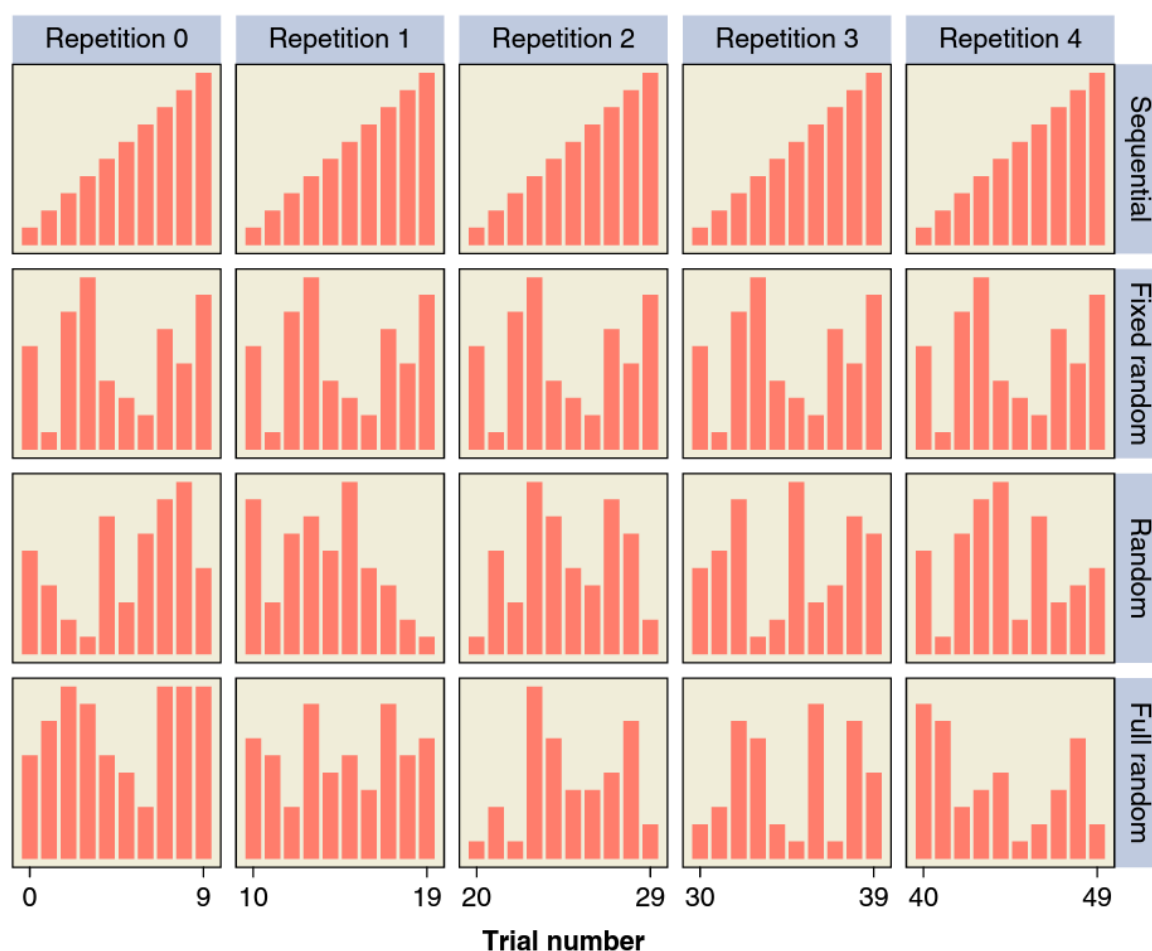
Conditions: \$condFile

No parameters set (conditionsFile not found)

Help OK Cancel

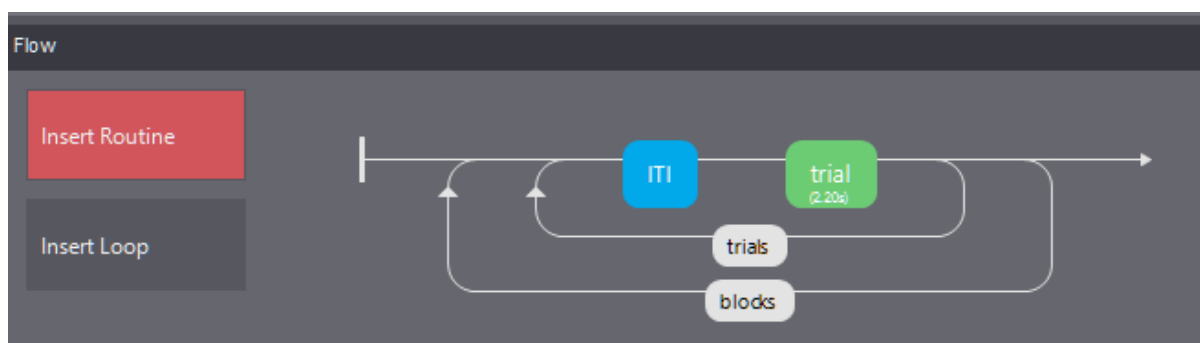
Randomiseerimine ehk juhuslikustamine võib toimuda uurimistöö läbiviimise väga erinevatel tasemetel ja etappides. Näiteks võime olla huvitatud, et meile satuksid

juhuslikud katseisikud mõnest populatsioonist. Võime soovida, et erinevate katseplokkide või tingimuste esitus oleks katseisikute vahel tasakaalustatud. Või soovime, et esitatud stiimuli mingi parameeter (nt kestus, asukoht vms) varieeruks juhuslikult esitusest esitusse. Randomiseerimine on seega eksperimentaalse uurimistöö üheks levinuimaks töövaheniks. PsychoPy randomiseerimise kaks kõige olulisemat parameetrit on randomiseerimise tüüp (*sequential*, *fixed random*, *random*, *full random*) ja korduste arv (nReps) ehk see, mitu korda tingimuste faili katse jooksul korratakse. Toodud joonisel on varieeritud randomiseerimise tüüpi selliselt, et nReps on 5.



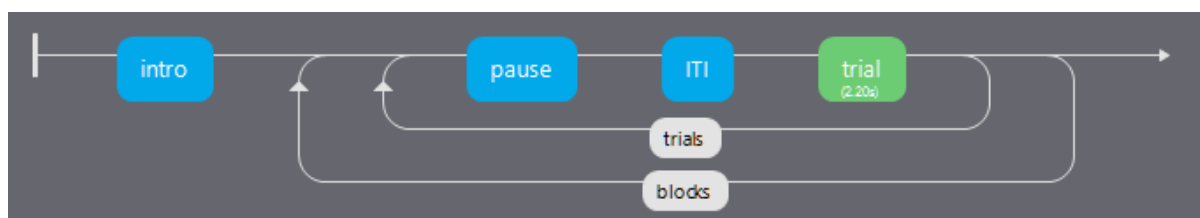
Eelmises katses vaatlusime kaht viisi, kuidas tingimuste tabeli veergude esitamist seadistada. Katse plokkide järjestamisel valisime *trials* seadetes randomiseerimise tüübiks (*loopType*) järjestikuse esitamise (*sequential*) ja seeriade juhuslikustamiseks kasutasime meetodit *random*. Kui järjest esitamise loogika on lihtsamini hoomatav, siis erinevat tüüpi juhuslikustamised tahavad rohkem läbimõtlemist. Näiteks PsychoPy *random* tüüpi juhuslikustamisel esitab programm tingimuste failis olevaid ridu määratud arvu kordi, selliselt, et kõigepealt võetakse tingimustabelist juhuslikult ridu, tehes seda seni kuni kõik read on otsa saanud. Seejärel võtab programm taas algse tingimuste tabeli ette ja hakkab sealt uuesti juhuslikult ridu võtma, seni kuni vajalik korduste arv on täis. Seda tüüpi juhuslikustamist demonstreerib üleval paikneval joonisel rida Random. Joonis kujutab olukorda, kus tingimuste tabelis on täpselt kümme veergu. Random tüüpi juhuslikustamine tähendab seega, et kõigepealt esitatakse juhuslikult kõik tingimuste tabelis olnud kümme rida ja alles siis saab mõni seal olnud rida järgmisel esitusel kordusele tulla. Postide kõrgus vastab rea järjekorra numbrile. Näeme, et järjest esitamisel (*sequential*) meenutavad joonised treppi. Programm esitab järjest kõik tingimustetabeli kümme veergu ja teeb seda

viis korda. Sellele järgnev *fixed random* ehk fikseeritud juhuslikustamine tähendab seda, et tingimuste tabeli read aetakse segi ja kõigil järgnevatel kordustel kasutatakse sama juhuslikku järjestust. See oleks sarnane olukorraga, kus enne katse käivitamist ajaksime tingimuste read segamini ja valiksime juhuslikustamise tüübiks *sequential* ehk järjest esitamise. *Fixed random* puhul tehakse aga katse käivitamisele eelnenud juhuslikustamine meie eest. PsychoPy-s on veel neljas valik, *full random*, mis oleks antud näite puhul sarnane olukorrale, kus muudaksime tingimuste faili selliselt, et kordaksime selle ridu viis korda, mis annaks kokku 50 rida. Seejärel käivitaksime katse *random* juhuslikustamist kasutades, aga tingimustabeli korduste arvuks määraksime ühe (ehk $nReps = 1$). Teiste sõnadega erineb see esituse viis *random* meetodist selle poolest, et selles võivad samad veerud korduda ka enne kui kõik veerud on ära esitatud. Seda mõtet illustreerib joonise all kõige vasakpoolsemas nurgas olev joonis, milles kümnendat rida esitatakse esimese kümne seeria jooksul neli korda.



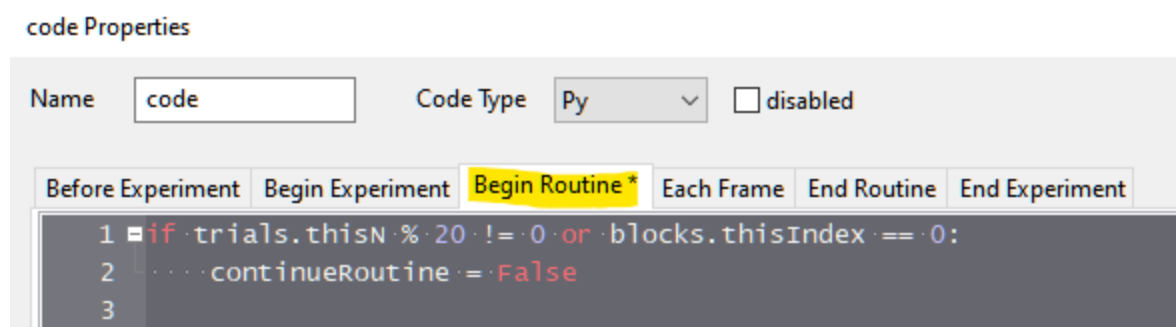
Kuna katse võib pikale minna, siis oleks hea anda katseisikule aeg-ajalt võimalus katsest veidi puhata. Selleks tekitame iga mingi hulga seeriade järel pausi, millest katseisik saab klahvivajutusega ise edasi minna. Selleks on mitmeid võimalusi, kuid selles näites lisame *ITI* rutiini ette uue rutiini, mis pausi teksti kuvaks.

Ülesanne 9. Lisa *ITI* rutiini elemendi ette uus element, mis esitaks teksti ja laseks katseisikul klahvivajutusega katset jätkata. Lisa rutiini elemendile ka koodielement ja seadista seda selliselt, et rutiini element käivituks ainult pärast treeningesitust ja ülejäänud katse vältel iga kahekümne esituse järel.



Kui lisada *ITI* rutiini elemendi ette „pause“ ja lisada selle koosseisu teksti ja vastuseklahvi komponendid, siis vaikimis käivitatakse see rutiini element iga esitusel. Selleks, et teksti kuvataks ainult meie poolt valitud esitustel saame teksti ja vastusekomponendi juurde lisada ka koodikomponendi, mille `BeginRoutine` saki all defineerime, millistel seeriatel rutiini esitatakse ja millistel mitte. Alumisel joonisel defineeritud koodikomponendis vastab `trials.thisN` jooksvale seeriale ehk see on muutuja, mis võtab endale iga seerial uue ja

eelmisest kõrgema väärtuse. Loendur `trials.thisN` läheb pärast iga ploki nulli. Meie loodud katses on kaks ploki: treening ja katseseeriad. Tegemist on sama arvuga, mis jõuab andmetabelisse veergu nimega `trials.thisN` ja samal moel saab kasutada ka teistesse veergudesse salvestatavat infot.



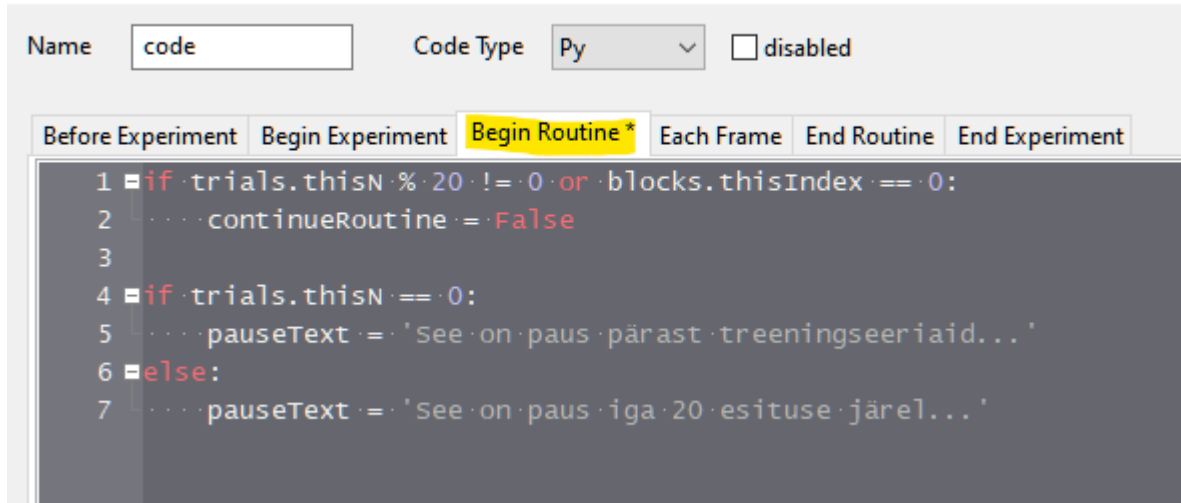
Kui sooviksime, et paus esitataks katseseeriade jooksul pärast iga teatud arvu seeriaid (näites 20), siis saame kasutada Pythoni modulo operatsiooni, mille tähiseks on protsendi märk. Modulo operatsioon ütleb, kas kaks arvu jaguvad teineteisega ja tagastab jagamisel saadud jäägi. Näiteks $5 \% 2$ on võrdne ühega, sest kaks mahub viie sisse kaks korda ja jääk on üks. Selle näite korral jagatakse `trials.thisN` arvuga 20 ja kui selle jääk on midagi muud kui 0 ehk arv ei jagu täpselt kahekümnega ($\neq 0$), siis rutiini ei käivitata (`continueRoutine = False`) ja pausi ei ilmu. Pane tähele, et Pythonis alustatakse loendamist nullist mitte ühest ja seega vastab kahekümnes seeria tegelikult kahekümne esimesele. Kuigi nulliga jagamine on problemaatiline, siis Python meile siin veateadet ei anna ja seda operatsiooni kasutades võetakse vaikiv eeldus, et mistahes arvu nulliga jagamine annab jäägiks nulli. Seega käivituks rutiin kahel juhul: 1) kui `trials.thisN` on null ja 2) kui `trials.thisN` jagub kahekümnega. Selles katses on `trials.thisN` null kahel korral: treeningseeriade ja katseseeriade alguses.

Tegelikult ei juhtuks midagi kui jätaksime ka treeningseeriade ette pausi, kuid järgnevalt vaatleme olukorda, kus sooviksime pausi esitada ainult treeningseeriade lõpus ja katses iga kahekümne seeria järel. Treeningseeriade järel pausi esitamine annaks eksperimendi läbiviijale hea võimaluse veel viimast korda veenduda, et katseisik mõistab katseinstruktsioone. Lisaks aitab see katseisikul mõista, millal treeningseeriad katseseeriadeks üle lähevad. Kui soovime, et enne esimest treeningseeriat pausi ei kuvataks (kui `trials.thisN` on null), siis võime süntaksi `trials.thisN % 20` järele kirjutada `or blocks.thisIndex == 0`, mis tähendab, et rutiini käivitamine katkestatakse ka juhul, kui tegemist on esimese ploki ehk treeningseeriatega. Sellisel juhul esitatakse paus ainult järgmise ploki alguses (ja iga kahekümne esituse järel), mis vastabki treeningseeriade lõpule ja seega rohkem muudatusi if-lause süntaksisse tegema ei pea.

Võib olla, et tahaksime pausi teksti varieerida sõltuvalt sellest, kas teksti paus esitatakse vahetult pärast treeningseeriaid või katse jooksul. Näiteks võiksime pärast treeningseeriaid kuvada ekraanile kirja, mis teataks katseisikule, et treeningseeriad on läbi, ning et nüüd on hea võimalus eksperimentaatorilt täiendavaid küsimusi küsida. Kiri võib ka paluda katseisikul oodata kuni eksperimentaator lülitab sisse psühhofüsioloogiliste signaalide salvestamise jms. Katse jooksul tahaksime ilmselt lihtsalt kommunikeerida, et tegemist on pausiga ja millise klahvi abil katseisik edasi liikuda saab.

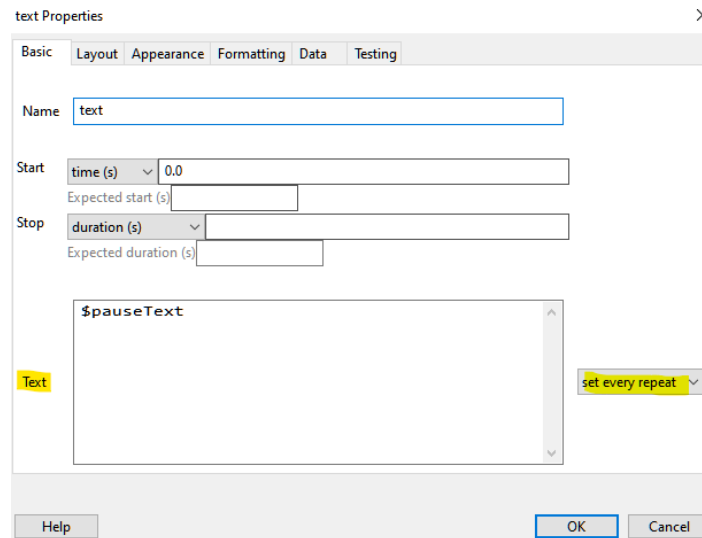
Esitatava teksti varieeruma panemiseks peaksime defineerima teksti sisaldava muutuja (kutsutakse ka sõneks), mis võtaks endale väärtuse sõltuvalt sellest, kas tegu on treeningseeriatega järgse pausi või tavalise pausiga.

code Properties

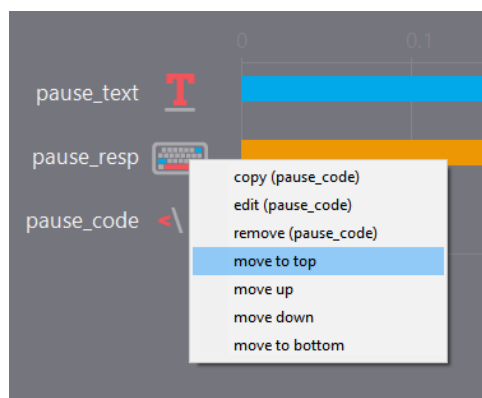


Ridadel 4-7 tehakse seda selliselt, et kõigepealt kontrollitakse, kas tegu on esimese seeriaga – seda, et tegemist ei oleks treeningseeriaga ei pea selles avaldises enam kontrollima, sest treeningseeriatel katkestatakse rutiini esitamine pärast teise rea (`continueRoutine = False`) käivitamist ja juhul kui tegu on treeningseeriatega, siis programm ei jõuagi hilisemate ridade jooksumiseni. Kood jõuab nende ridadeni seega ainult kahel juhul: 1) kui `trials.ThisN` on null ja tegu pole treeningseeriatega või 2) kui `trials.ThisN` jagub kahekümnega. Ühel juhul on tegemist treeningule järgnevate seeriatega algusega ja teisel juhul pausiga. Seega saame kirjutada, et kui `trials.ThisN` on võrdne nulliga ehk tegu on esimese seeriaga, siis on pausi ajal esitatav tekst ühesugune ja kui see võrdsus rahuldatud ei ole siis teistsugune, sest järelikult on siis tegemist tavalise pausiga. Sõne defineerimiseks anname muutujale nime (antud näites on selleks `pauseText`) ja kirjutame võrdusmärgi järel ja jutumärkide sisse teksti, mida sooviksime selles tingimuses kuvada.

Selleks, et muutujasse defineeritud lause ka ekraanile ilmuks peaksime tekstikasti kirjutama dollari märgiga selle muutuja nime (sarnaselt sellele nagu Stroopi katses kirjutasime sinna sõnade veeru nimetuse) ja paneme tekstikasti igal seerial muutuma (ehk *set every repeat* valik tekstikasti järel).



NB! Meenutuseks, et programm hakkab komponente sisse lülitama ülevalt alla. Selleks, et katse kokku ei jookseks peaksime koodikomponendi tekstikomponendist varasemaks tõstma, sest muidu käivitab programm teksti komponendi, kuid kuna muutuja pauseText pole veel defineeritud, siis jookseb katse kokku.



Kui lisada katsele ka sissejuhatav tekst ja lõppu tänusõnad, siis on tegu juba täiesti toimiva katsega.

Vaikimisi prindib PsychoPy katsefaili ka komponentide sisse- ja väljalülitamise ajad. Kõiki neid aegu ei ole meil enamasti siiski tarvis – eriti kui tegu on instrueerivate tekstide ja nende juurde kuuluvate vastusekomponentide aegadega. Nende aegade andmetabelisse jõudmise takistamiseks peaksime klõpsama komponendil, millega seotud aegu me andmetabelisse printida ei soovi ja **Data** saki all võtta ära linnuke **Save onset/offset times** eest.

