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 alati 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.

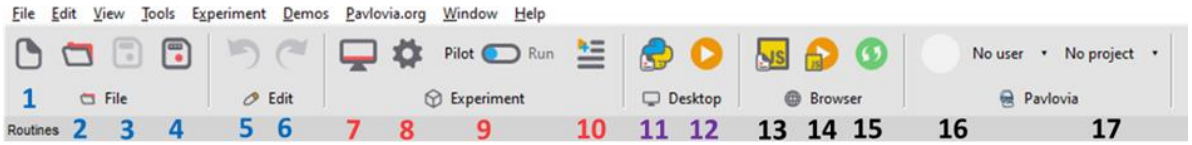
Posneri ruumitähelepanu katses instrueeritakse katseisikuid hoidma pilku ekraani keskel paikneval fiksatsiooniristil ja raporteerida kummale poole, vasakule või paremale, sihtstiimul (*traget*) ilmus. Enne sihtstiimuli ilmumist esitatakse katseisikule vihje, mis viitab kummale poole sihtstiimul ilmub. Enamikel juhtudel (80%) langeb vihje suund ja sihtstiimuli asukoht kokku, kuid väiksel arvul seeriatest tüssab 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 ilmumist. Kui vihje ilmub varem, siis on katseisikute vastused hoopis aeglasemad.

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

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

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

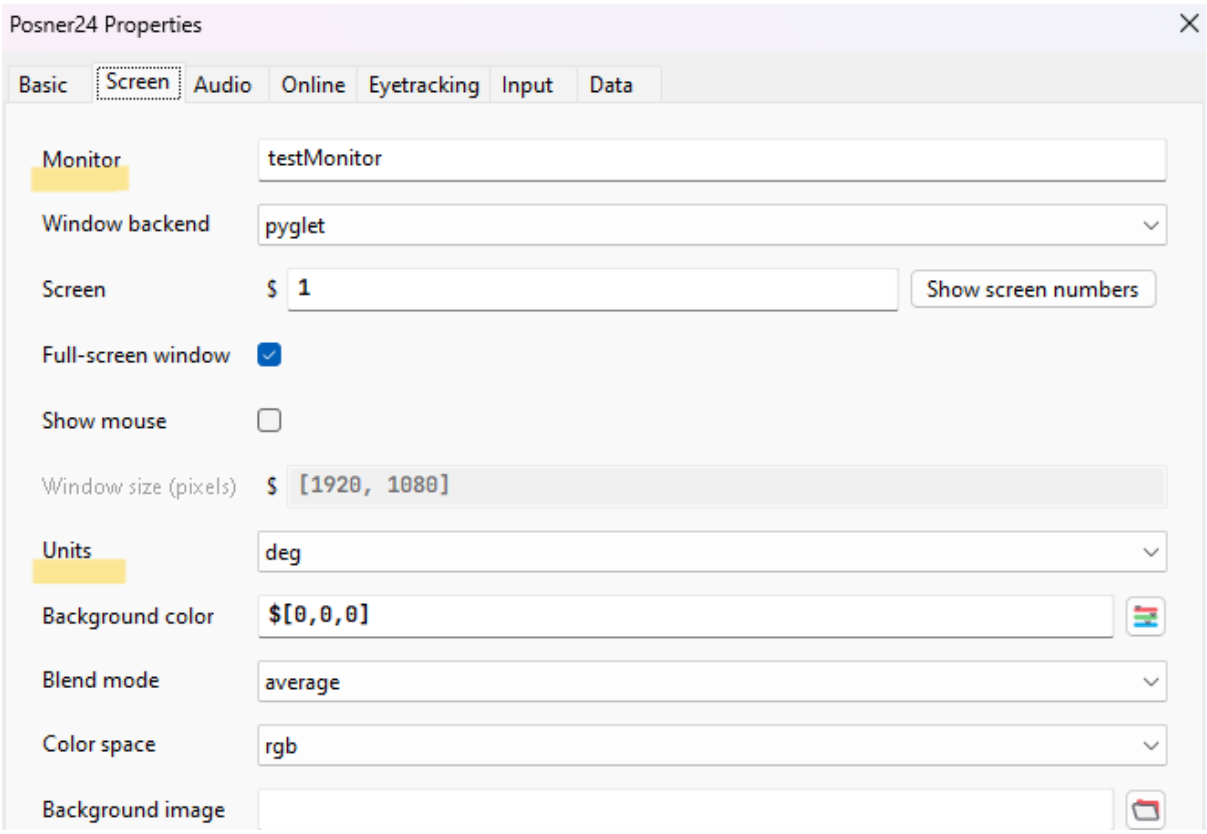
Seadete muutmiseks klõpsa ikoonide menüüs monitori pildiga ikoonil (**7**) ja lisa monitori kaugus katseisikust (nt 57 cm), ekraani suurus pikslites (nt 1920 x 1080) ning ekraani laius (nt 30 cm).



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 (nt testMonitor).

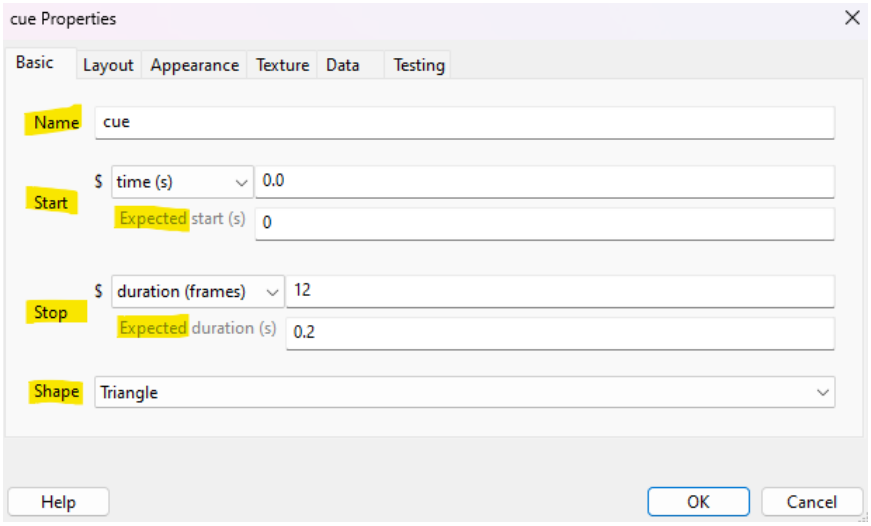
Selleks, et eksperiment kasutaks vaikimisi nägemisnurga kraade, vali eksperimendi seadete all ühikuteks (Units) deg ehk kraadid.



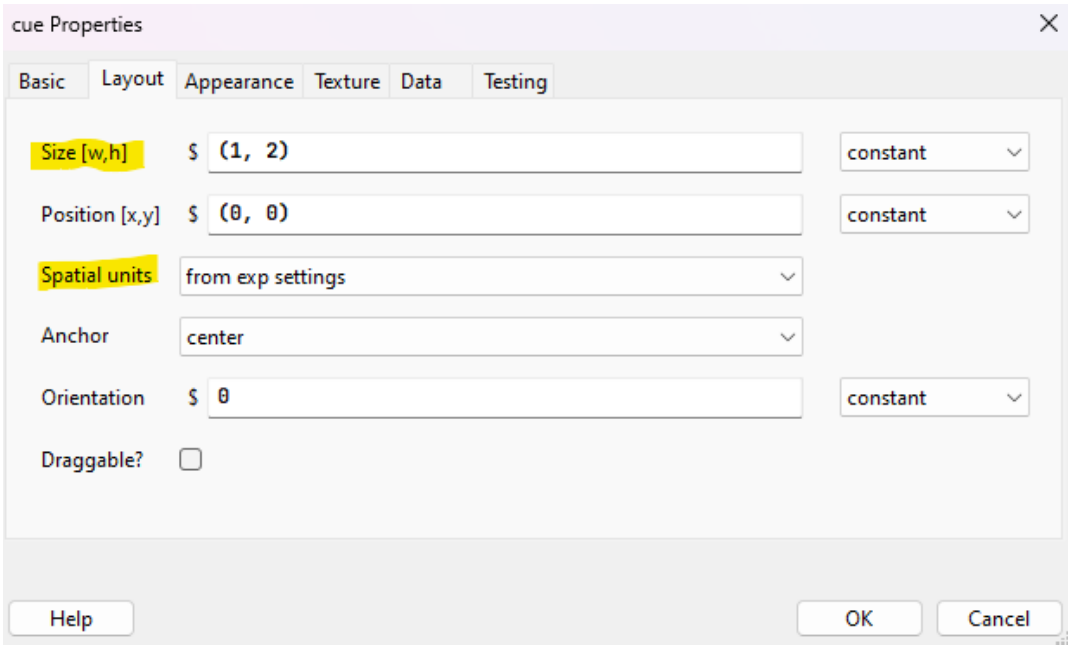
Ülesanne 3. Vali komponentide menüüst visuaalne stiimul, millest saab katses kasutatav vihjav kolmnurk (*Polygon*), anna sellele nimeks *cue* ja lisa see *trial* rutiinielemendi koosseisu. Muuda selle komponendi järgnevaid parameetreid:

a) sea algushetkeks rutiinielemendi algus ehk 0 ja defineeri selle kestus ekraanivärskendustes, andes sellele väärtuseks 12

b) määra komponendi suuruseks 1 x 2 kraadi
c) 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 väljanägemisele ja stiimuli tegeliku kestuse määrab ikka lahtrisse duration (frames) kirjutatu



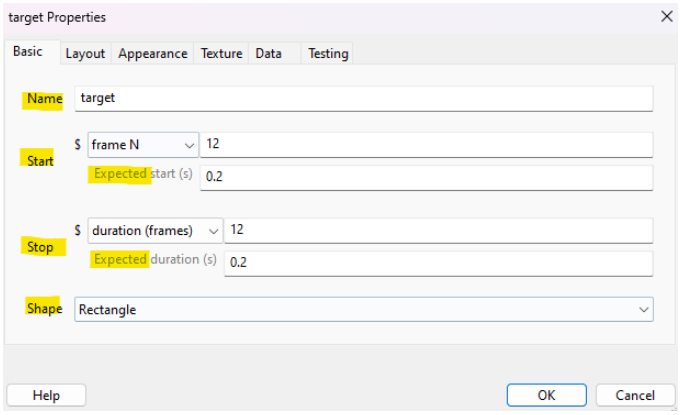
Kuna komponendi kujuks on juba vaikimisi kolmnurk, siis seda eraldi muuta ei ole tarvis. Muuta ei ole vaja ka komponendi suurusühikuid, sest muutsime need juba eksperimendi seadete juures kraadideks ja vaikimis pärivad komponendid enda ühikud sealt (vt alumisel joonisel *from exp settings*).

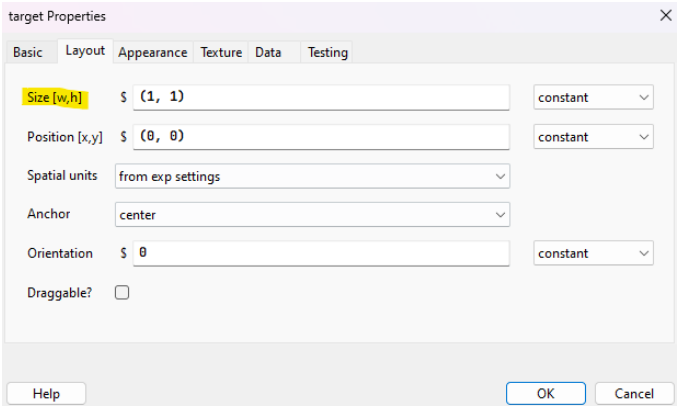


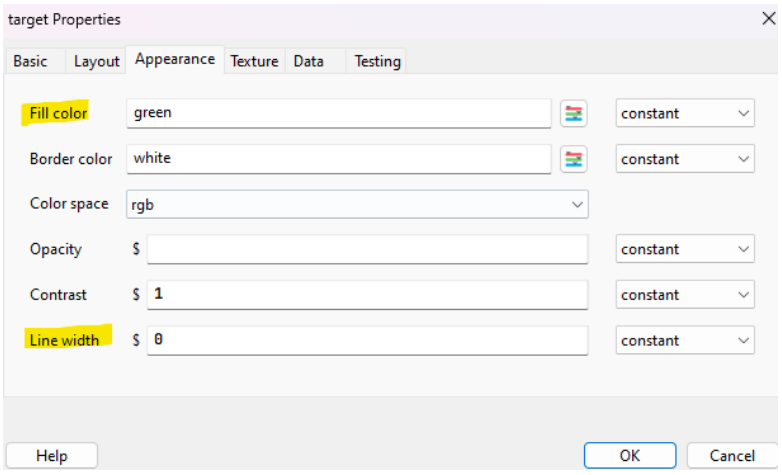
Ülesanne 4. Vali komponentide menüüst visuaalne stiimul, millest saab katses kasutatav sihtstiimul (*Polygon*), anna sellele nimeks *target* ja lisa see *trial* rutiinielemendi koosseisu. Muuda selle komponendi järgnevaid parameetreid:

a) sea algushetkeks vihjava stiimuli lõpp ehk 12 värskendust

b) määra komponendi suuruseks 1 x 1 kraadi
c) määra komponendi värvuseks roheline ja anna kasti ümbritseva raami suuruseks 0, mis tagab, et kasti ümber raami ei esitataks
d) kuigi see ei oma funktsionaalset tähtsust, siis võid komponendi oodatud algusajaks (Expected duration (s)) siingi määrata 12/60 ehk 0,2 sekundit – sellisel juhul kuvatakse komponendi kestus ka graafilises liideses, kuid selle mõju on ainult graafilise liidese välimusele ja stiimuli tegeliku kestuse ja algusaja määrab vastavalt duration ja frame N lahtritesse kirjutatu

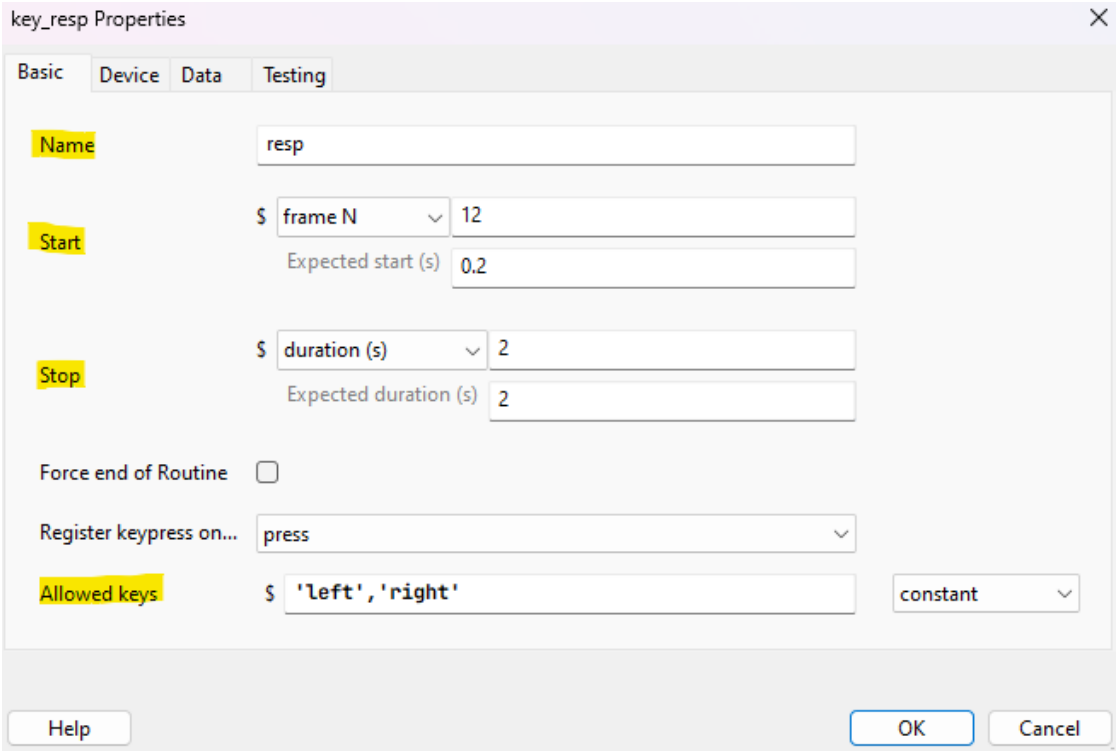




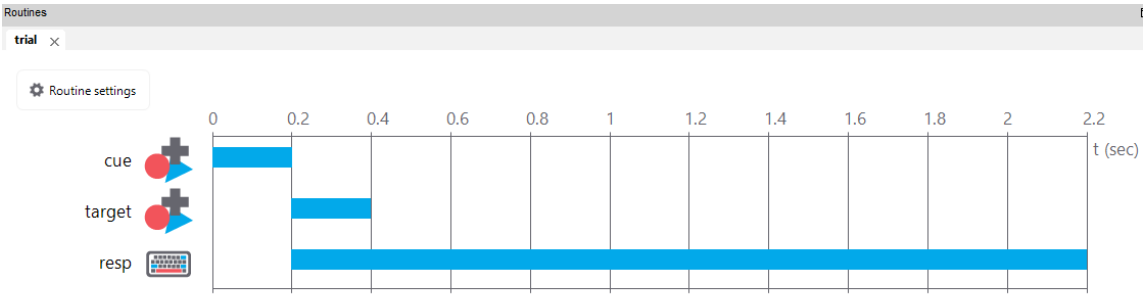


Ülesanne 5. Vali komponentide menüüst vastuseklahv, anna sellele nimeks *resp* ja lisa see *trial* rutiinielemendi koosseisu. Muuda selle komponendi järgnevaid parameetreid:

a) 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
b) sea lubatud klahvivajutusteks vasak ja parem

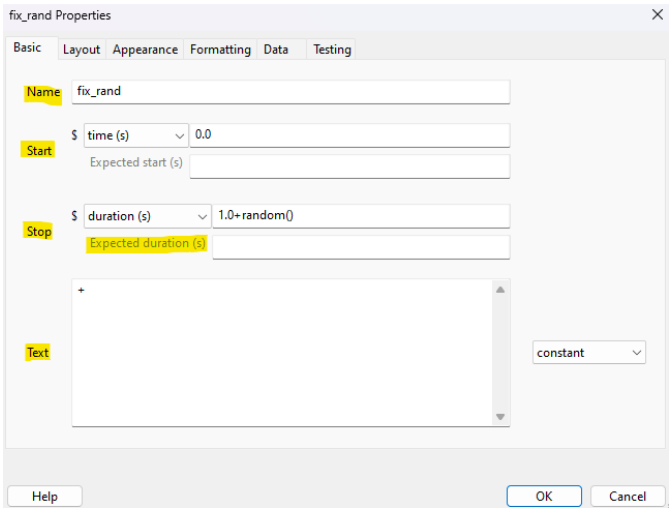


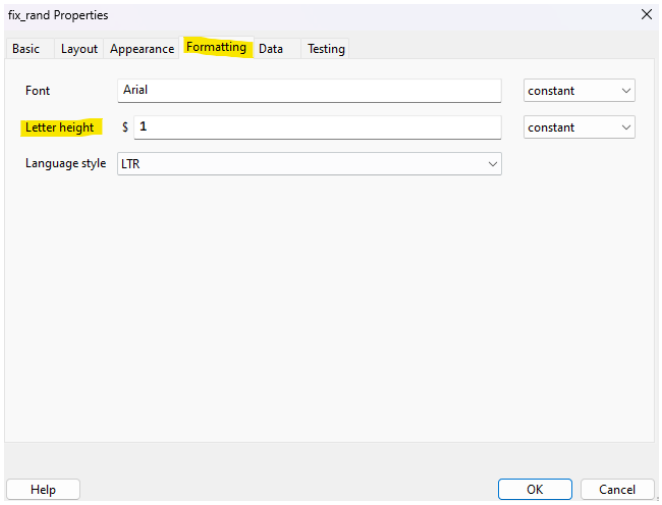
Pärast neid muudatusi peaks *trial* rutiin välja nägema umbes selline:



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:

a) anname komponendile nimeks *fix_rand*
b) 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 keskeltläbi on seeriate vaheliseks ajaks 1.5 sekundit)
c) sisesta tekstikasti plussmärk (+)
d) määra teksti kõrguseks 1 kraad





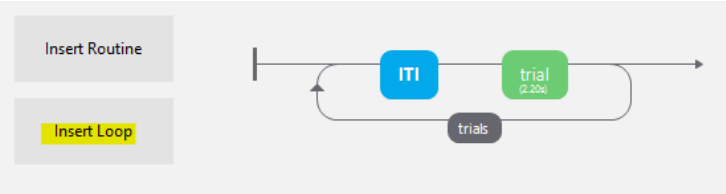
Kui sa pole veel proovinud katset käivitada, siis tee seda kindlasti enne järgmiste ülesannete juurde minemist. Kui oled veendunud, et kõik toimib, siis liigu järgmiste ülesannete juurde.

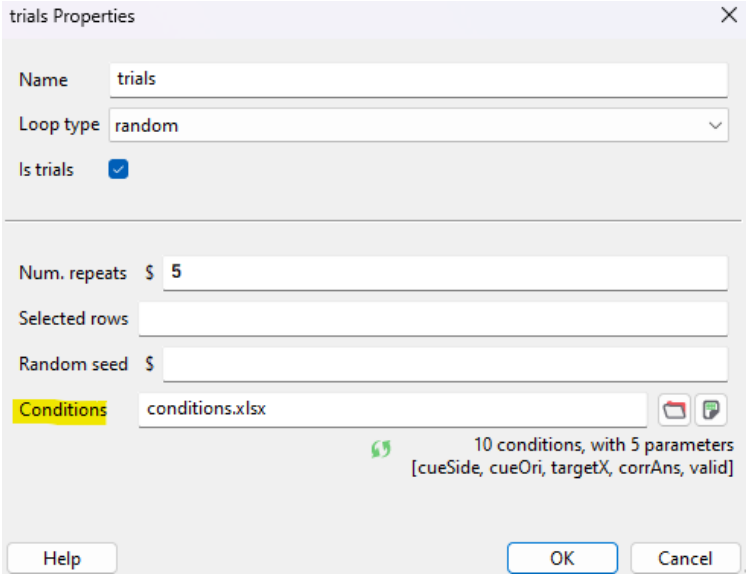
Ülesanne 7. Loo tingimuste fail, mille veerud defineerivad vihje poole (*cueSide*), kaldenurga (*cueOri*), sihtstiimuli x-koordinaadi (*targetX*), õige vastuse klahvi (*corrAns*) ja vihje valiidsuse (*valid*). Loo tingimuste fail selliselt, et pooltel juhtudel esitataks sihtstiimul vasakule ja pooltel paremale (sh mittevaliidsetel seeriatel), 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.

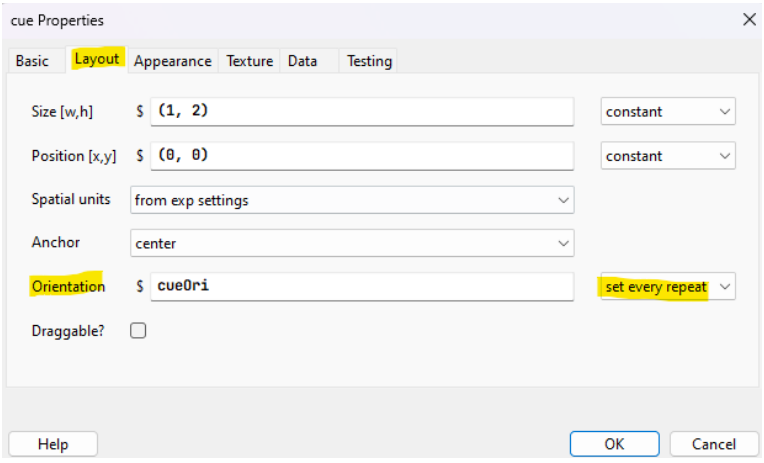
	A	B	C	D	E	F
1	cueSide	cueOri	targetX	corrAns	valid	
2	left	-90	-7	left	1	
3	left	-90	-7	left	1	
4	left	-90	-7	left	1	
5	left	-90	-7	left	1	
6	right	90	7	right	1	
7	right	90	7	right	1	
8	right	90	7	right	1	
9	right	90	7	right	1	
10	right	90	-7	left	0	
11	left	-90	7	right	0	
12						

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



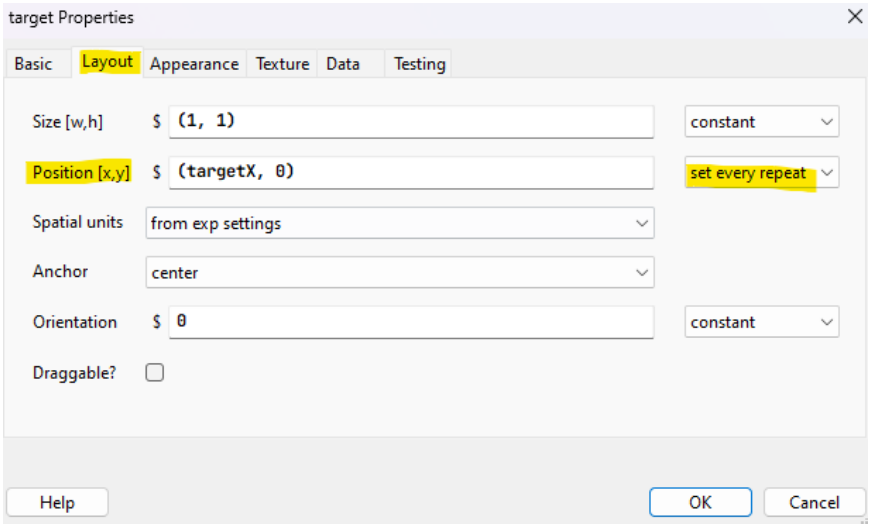


Ülesanne 7. Järgmiseks muuda *trial* rutiinielemendi koosseisus olevate vihje, sihtstiimuli ja vastuse komponendi seadeid selliselt, et neis võetaks arvesse tingimuste tabelis defineeritud väärtused.

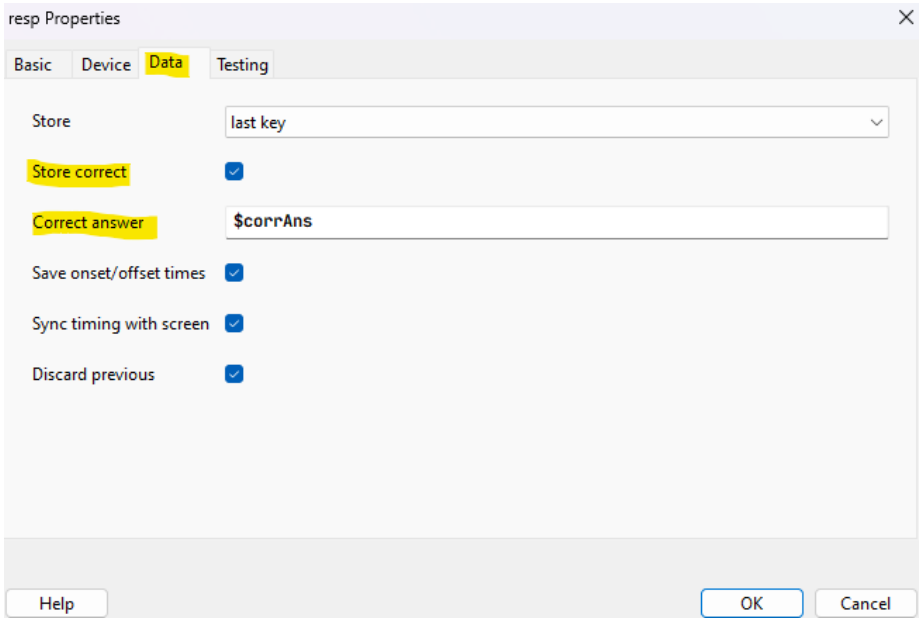


Vihje puhul muudame igal esitusel vihje kaldenurka ehk seda, kas vihje näitab vasakule või paremale. Selle muudatuse tegemiseks klõpsa vihje komponendil (*cue*) ja kirjuta lahtrisse Orientation veeru nimi, kuhu vihje kaldenurga kirjutasid (*cueOri*). Pane kaldenurk kindlasti ka igal esitusel varieeruma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



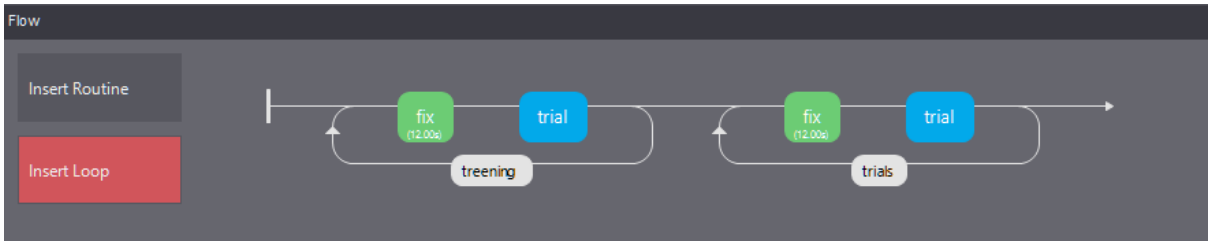
Ülesanne 8. Enamasti tahame katsele lisada ka treeningseeriad, mille käigus on katseisikul võimalik katsega tutvuda. Treeningseeriate lisamiseks saame kasutada pesastatud süntaksit. Selleks loome uue tingimuste tabeli (nt training.xlsx) ja lisame sellesse tabelisse esitused,

mida soovime, et katseisik treeningul läbiks. Antud näite kontekstis võib olla kasulik demonstreerida, et vihje ei ole alati korrektne.

Oletame, et lisame treeningseeriate tingimuste faili neli rida, 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 kummalgi 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.

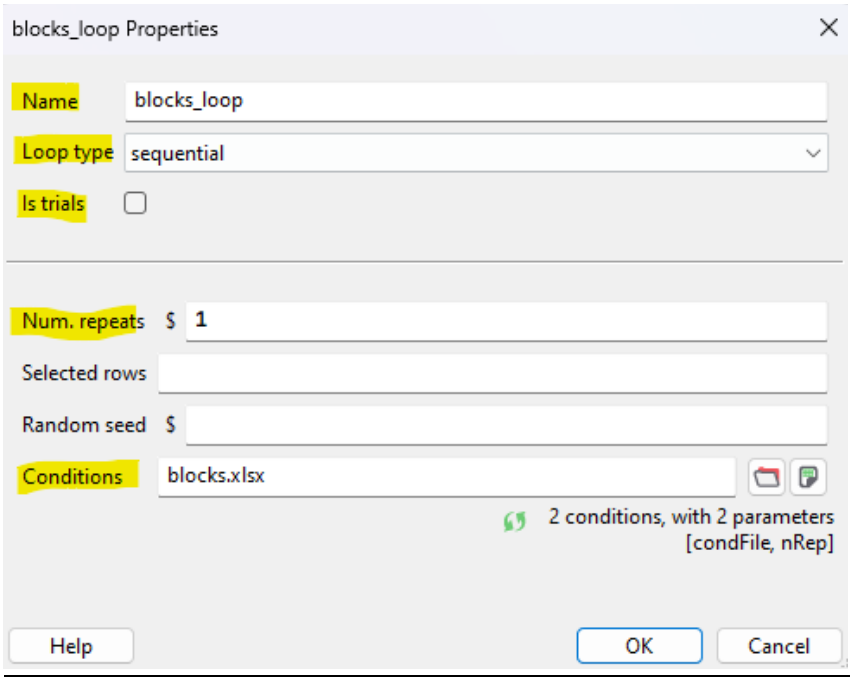
Ülesanne 8.1. Loo *trials* aasa ümber veel üks aas, mis esitaks kõigepealt kõik treeningseeriate tabelisse (training.xlsx) lisatud read ja seejärel kordaks kõiki põhikatse tingimustabeli ridu (conditions.xlsx).

Selleks tekitame *trials* aasa ümber veel ühe aasa (*Insert Loop*), millele võime anda nimeks *blocks_loop*. 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*).

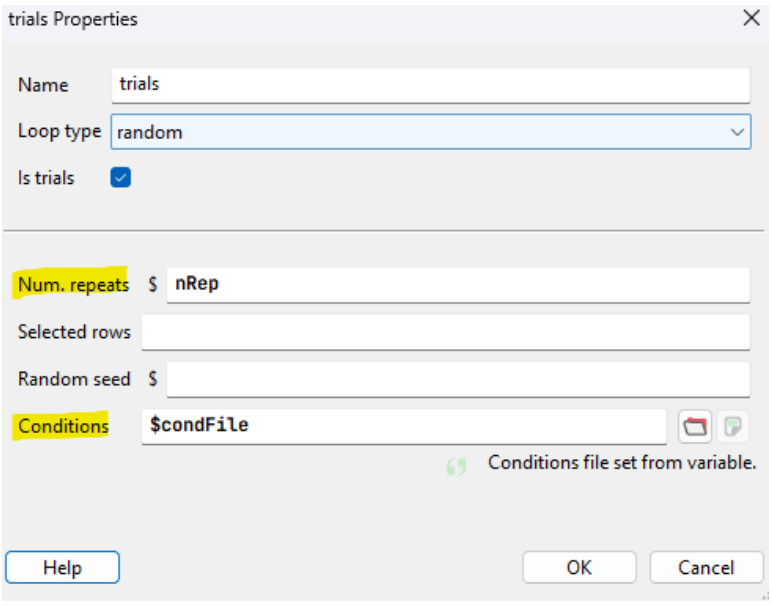
	A	B
1	condFile	nRep
2	training.xlsx	1
3	conditions.xlsx	5

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 tingimustefaile Num. repeats korda. Antud juhul tahaksime, et programm valiks kõigepealt ühe (training.xlsx) ja siis teise tingimustefaili (conditions.xlsx) ja seega valime juhuslikustamise tüübiks *sequential* ehk

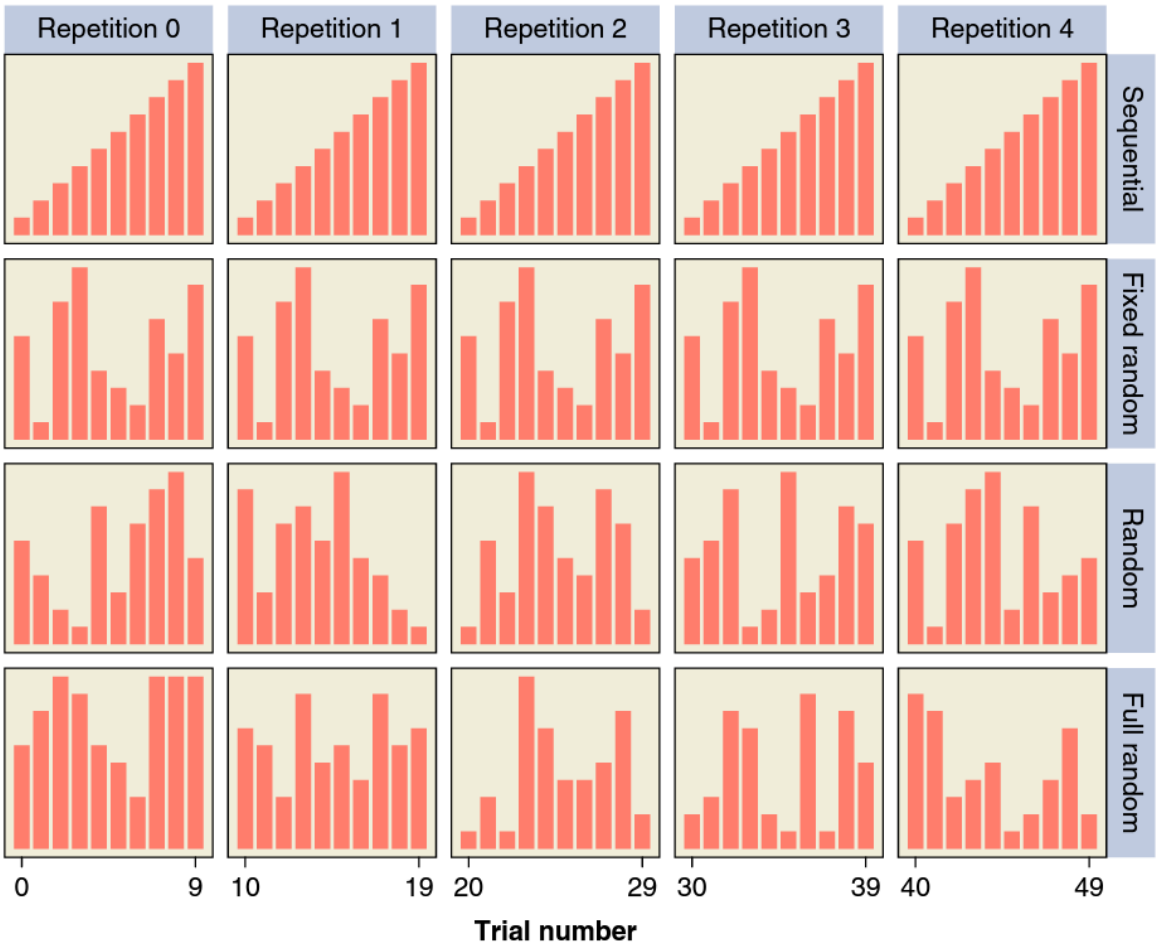
järjest esitamise. Lisaks võime ära võtta linnukese „Is trials“ eest, mis annab PsychoPyle märku, et tegemist ei ole seeriatega.



Kui need muudatused on tehtud, siis avame *trials* aasaga seotud seaded ja kirjutame Num. repeats 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 vaikimisi tingimuste faili nimetuseks, kuid antud hetkel viitame hoopis veerule tingimuste tabelis. Pööra tähelepanu, et Num. repeats puhul ei pidanud me seda tegema, sest selle veeru ees on juba vaikimisi dollari märk (vt alumine joonis).

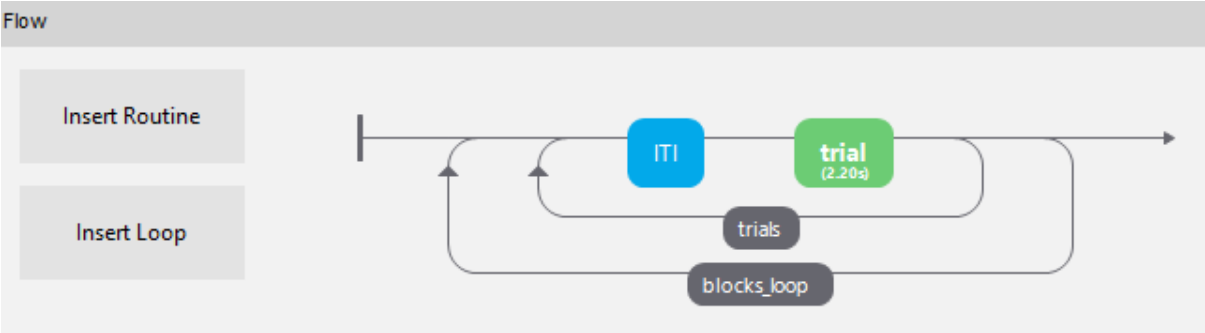


Randomiseerimine ehk juhuslikustamine võib toimuda uurimistöö läbiviimise väga erinevatel tasemetel ja etappides. Näiteks võime olla huvitatud, et meile satuksid juhuslikud katseisikud mõnest populatsioonist. Võime soovida, et erinevate katseplokkide või tingimuste esitus oleks katseisikute vahel tasakaalustatud. 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 (Num. repeats) ehk see, mitu korda tingimuste faili katse jooksul korratakse. Toodud joonisel on varieeritud randomiseerimise erinevaid tüüpe selliselt, et Num. repeats on 5.



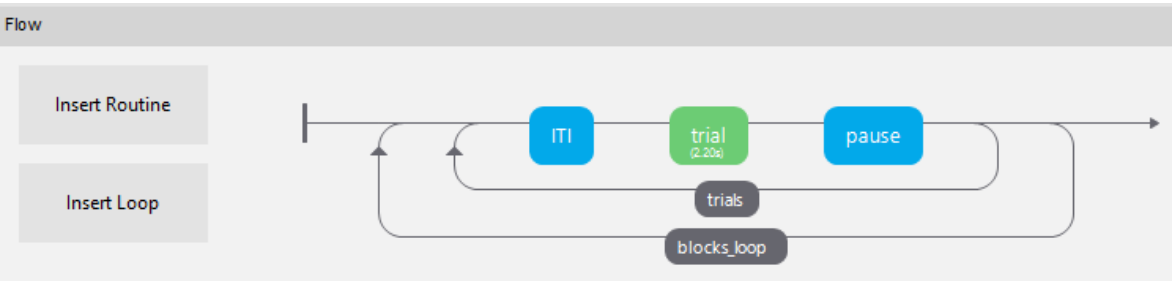
Eelmises katses vaatlesime kaht viisi, kuidas tingimuste tabeli veergude esitamist seadistada. Katse plokkide järjestamisel valisime *trials* seadetes randomiseerimise tüübiks (loopType) järjestikuse esitamise (*sequential*) ja seeriate juhuslikustamiseks kasutasime meetodit *random*. Kui järjest esitamise 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 juhulikult 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 *sequentsial* 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 Num. repeats = 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 kõige vasakpoolsemas nurgas olev joonis, milles kümnendat rida esitatakse esimese kümne seeria jooksul neli korda.



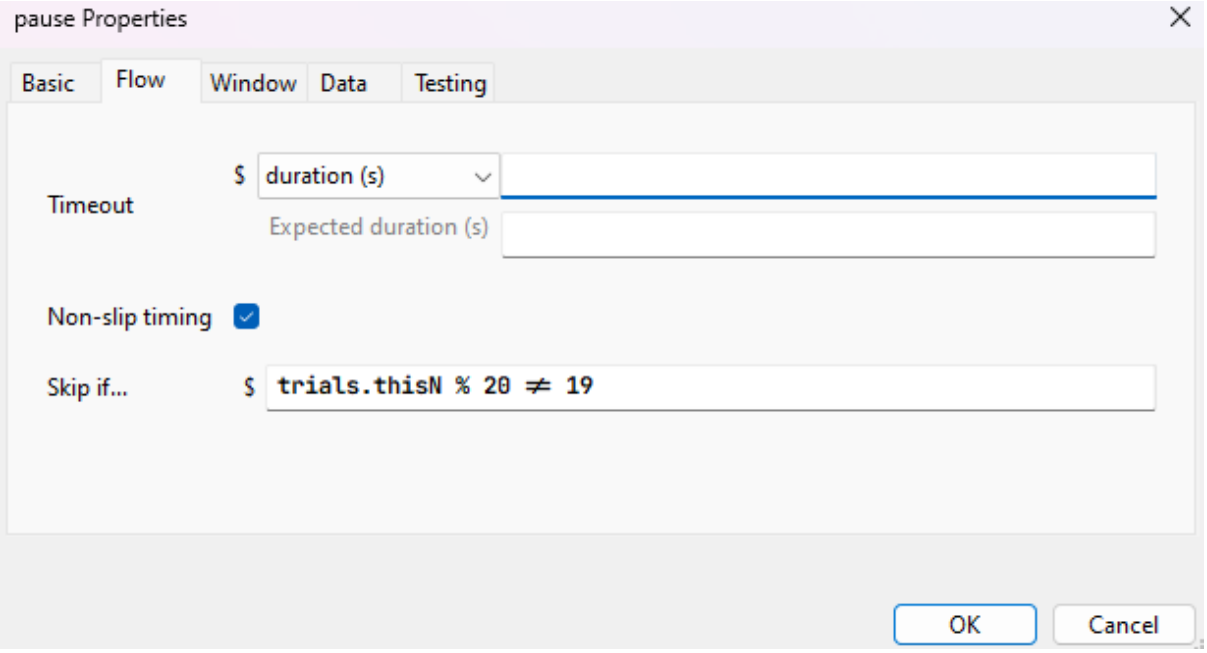
Kuna katse võib pikale minna, siis oleks hea anda katseisikule aeg-ajalt võimalus katsest veidi puhata. Selleks tekitame iga mingi hulga seeriate järel pausi, millest katseisik saab klahvivajutusega ise edasi minna. Selleks on mitmeid võimalusi, kuid selles näites lisame ITI rutiini ette uue rutiini,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

Ülesanne 9. Lisa *trial* rutiinielemendi järele uus element, mis esitaks teksti ja laseks katseisikul klahvivajutusega katset jätkata. Muuda loodud rutiinielemendi sätteid selliselt, et see käivituks vaid iga kahekümne esituse järel.



Kui lisada *trial* rutiinielemendi järele „pause“ ja lisada selle koosseisu teksti ja vastuseklahvi komponendid, siis vaikimis käivitatakse see rutiinielement igal esitusel. Selleks, et teksti

kuvataks ainult meie poolt valitud esitustel tuleks muuta rutiini sätteid[2] selliselt, et kirjutame rutiini seadete Flow saki all oleva „Skip if…“ järele, et sooviksime rutiini käivitada iga 20 seeria järel. Selles näites kasutame Pythoni modulo operatsiooni, mille tähiseks on protsendi märk. 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. Kui protsendimärgist vasakule jääv arv on väiksem kui sellest paremale poole jääv arv, siis tagastab Python protsendi märgist vasakule jääva arvu. Seega võime „Skip if…“ järele kirjutada: trials.thisN % 20 != 19, kus trials viitab meie esituste aasale, mis hetkel kannab nimetust trials ja thisN viitab sellele mitmenda esitusega on tegemist (näed seda sama nimetust ka andmetabelis ja kõiki andmetabelis nähtavaid veerge saab PsychoPy siseselt kasutada).



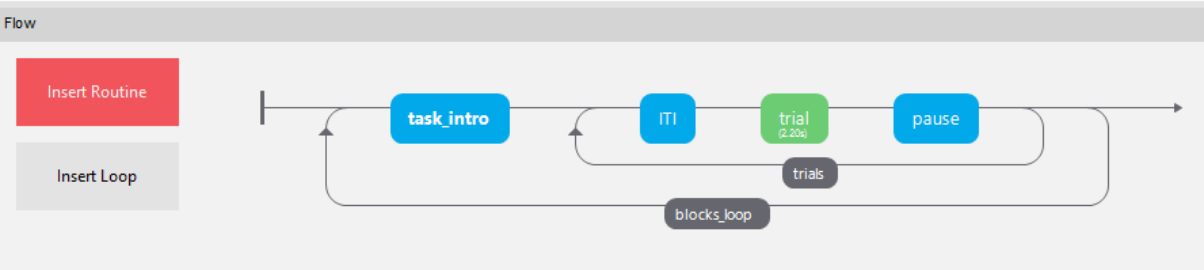
See süntaks ütleb PsychoPy-le, et käivita rutiin ainult siis, kui jääk on 19. See võib juhtuda kahel juhul: kui thisN ehk esituste arv on 19 või kui see on X × 20 + 19. Võib tekkida õigustatud küsimus, miks me loendame üheksateistkümneni, kui ülesandeks oli pärast iga kahekümnendat esitust paus teha. See on seetõttu, et Python ja seega ka PsychoPy alustab loendamist nullist ja lihtsalt võtame seda siin arvesse.

Ülesanne 10. Lisa *blocks_loop* aasa sisse rutiinielement, mis esitaks katseisikule katseinstruktsiooni, mille sisu sõltub sellest, kas katseisik alustab treening või katse esituste sooritamist. Seadista rutiin selliselt, et selle kestuks sõltuks katseisiku klaviatuuriklahvi vajutusest.

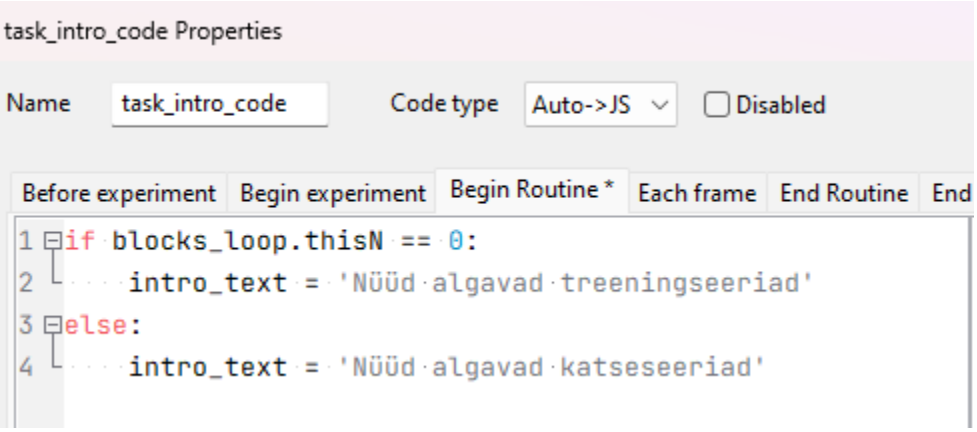
Esmalt lisame blocks_loop aasa koosseisu, kuid trial aasast väljapoole (vt alumine joonis), rutiinielemendi, mille koosseisu lülitame koodikomponendi (task_intro_code), tekstikomponendi (task_intro_text) ja klaviatuurikomponendi (task_intro_key).

[2] Varasemates PsychoPy versioonides ei olnud veel rutiinielementidel oma seadeid ja rutiini katkestamine tuli seadistada koodikomponendis. Teatud juhtudel võib seda lähenemist endiselt tarvis minna, kuid enamikel juhtudel saame kasutada rutiinielementide Flow saki all olevat „Skip if“ kastikest.

Kõigepealt lisan uue rutiinielemendi, millele annan nimetuseks task_intro:



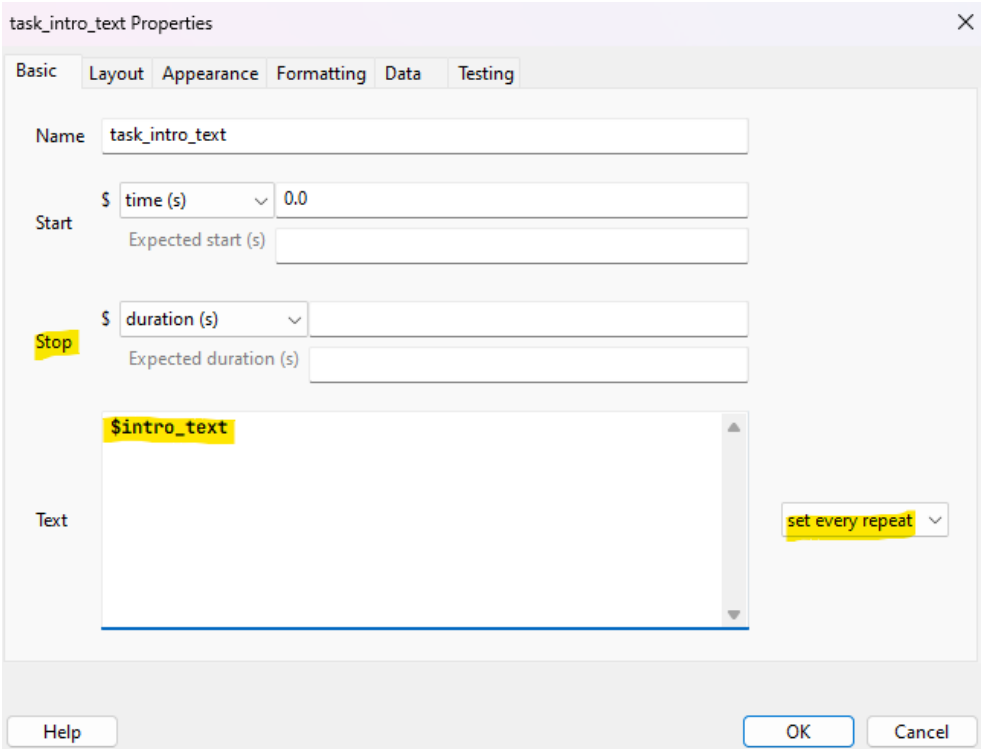
Seejärel lisan selle koosseisu koodikoponendi[3] (*task_intro_code*), mille Begin Routine saki alla lisan järgmise süntaksi:



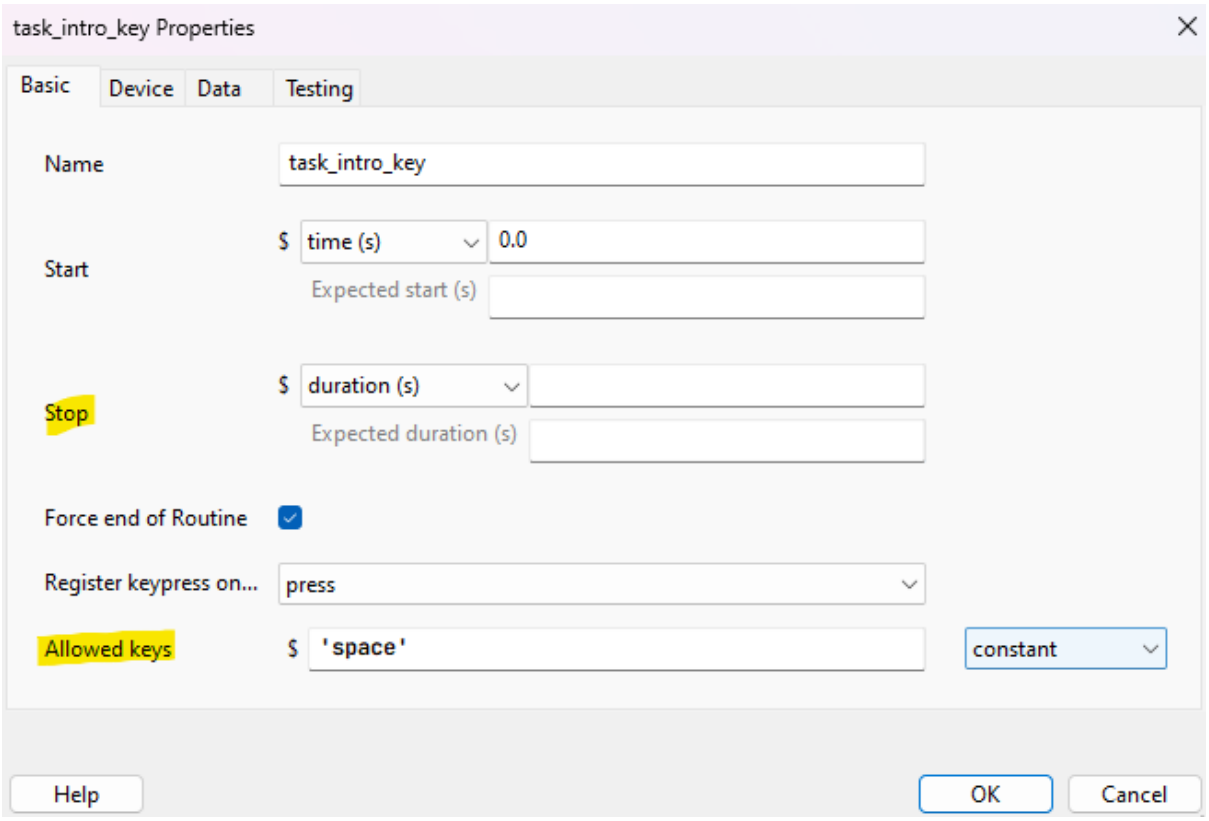
See süntaks küsib, kas on tegemist blocks_loop aasa esimese esitusega (kasutame siin andmetabeli veergu blocks_loop.thisN). Kui on tegemist esimese esitusega ehk blocks_thisN on 0, siis antakse muutujale intro_text üks väärtus (antud juhul: „Nüüd algavad treeningseeriad“) ja kui mitte (ehk on tegemist teise seeriaga), siis saab intro_text teise väärtuse (antud näites „Nüüd algavad katseseeriad“).

Järgmiseks lülitame task_intro rutiinielemendi koosseisu tekstikomponendi, mis meie poolt koodielemendis defineeritud teksti esitama hakkab. Kuna esituse lõpetab klaviatuuriklahvi vajutus, siis jätame kestuse defineerimata. Tekstikasti sisestame aga dollari märgi järele meie poolt loodud muutuja (intro_text). Oluline on muuta tekstikasti järel olevaid sätteid selliselt, et tekstikasti sisu värskendataks pärast iga esitust. Vastasel korral ei leia PsychoPy meie poolt defineeritud muutujat ja annab veateate.

[3] NB! Nende komponentide järjekord on antud juhul väga oluline, sest PsychoPy hakkab komponente sisse lülitama ülevalt alla. Selleks, et katse kokku ei jookseks, peaks koodikomponent enne tekstikomponenti olema. Vajadusel saame selle varasemaks tõsta. Selleks tuleks koodikomponendi peal teha parem hiireklõps ja valid „move to top“.



Lõpuks lisame ka klaviatuurikomponendi. Anname sellele nimeks taks_intro_key ja jätame kestuse defineerimata, sest linnuke Force end of Routine järel hoolitseb selle eest, et rutiin pärast klahvivajutust (antud näites tühik) peatuks.



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.