Veleučilište u Virovitici

Ekonomija, Turizam, Telekomunikacije i Računarstvo

Vol. VIII, br. 1, 2026.

Virovitica University of Applied Sciences

Economics, Tourism, Telecommunications and Computer science

Vol. VIII, No. 1,
2026.

Impressum

Nakladnik/Publisher:
Veleučilište u Virovitici - Virovitica University of Applied Sciences

Glavni urednik/Editor in chief:
nas.izv.prof.dr.sc. Dejan Tubić, prof.struč.stud, glavni urednik

Izvršni urednik/Executive Editor:
dr.sc. Željka Kadlec, prof.struč.stud.

Lektura/Linguistic Adviser:
Ivana Vidak Teskera, dipl.bibl. i prof.

Tehnički urednik/Technical Editor:
Siniša Kovačević, mag.ing.tech.inf., pred.

Adresa uredništva/Address of the Editorial Board:
Veleučilište u Virovitici
Matije Gupca 78, 33000 Virovitica
Tel: +385 33 721 099
Fax: +385 33 721 037
E-mail: urednik@vuv.hr

Naslovnica/Front Page:
Veleučilište u Virovitici/Virovitica University of Applied Sceince

Grafičko oblikovanje/Graphic Design:
Veleučilište u Virovitici/Virovitica University of Applied Sceince

Izlazi od/Since:
2019. godina/Year 2019.

Učestalost izlaženja časopisa/Publishing frequency:
Dva puta godišnje/Biannually

ISSN 2670-8930
DOI: https://doi.org/10.70077/et2er

Dijamantni časopis / Diamond Journal - Časopis je usklađen s kriterijima baze *Diamond Discovery Hub*

Časopis je uvršten u **ERIH PLUS** (European reference index fo the humanites and social sciences) bazu, čime je postao časopis koji se kategorizira u znanstvene radove druge skupine (a2).

Predgovor

"

Časopis „ET²eR – ekonomija, turizam, telekomunikacije i računarstvo" obuhvaća teme iz područja ekonomije, s posebnim naglaskom na poduzetništvo i menadžment, turizma, kao i teme iz domene informacijskih i komunikacijskih tehnologija te računalnog programiranja. Časopis se bavi i onim temama koje su povezane s problematikom interdisciplinarnog pristupa gore navedenih područja.

Časopis „ET²eR" namijenjen je svima koji žele dati doprinos poticanju i razvijanju primijenjene stručne djelatnosti. Svrha časopisa je upoznavanje šire javnosti s novostima iz navedenih područja i popularizacija struke. Stoga ohrabrujem sve potencijalne autore da prijave svoje radove za objavljivanje.

Zahvaljujem se svim autorima, recenzentima, uredništvu časopisa na znanju i trudu uloženom na kreiranje ovog broja časopisa „ET²eR – ekonomija, turizam, telekomunikacije i računarstvo".

"

Glavni urednik

nas.izv.prof.dr.sc. Dejan Tubić, prof. struč. stud.

Recenzenti - *Reviewers*

Marta Alić
Tehničko veleučilište u Zagrebu - *Zagreb University of Applied Sciences*

Božidar Jaković
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Marijana Špoljarić
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Vlado Halusek
Sveučilište Sjever – *University North*

Mladena Bedeković
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Dubravka Maras
Sveučilište Vern - *Vern University*

Siniša Kovačević
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Damir Vuk
Veleučilište u Virovitici (umirovljeni profesor) - *Virovitica University of Applied Sciences*

Ivana Vidak Teskera
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Rikard Bakan
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Dejan Tubić
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Nikolina Pleša Puljić
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Zrinka Blažević Bognar
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Damir Ribić
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Irena Bosnić
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Marko Hajba
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Kristijan Čović
Veleučilište Baltazar Zaprešić - *Baltazar Zaprešić Polytechnic*

Anita Prelas Kovačević
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Danijela Vakanjac
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Ivan Heđi
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Matko Zrnić
Veleučilište u Virovitici - *Virovitica University of Applied Sciences*

Sendi Deželić
Veleučilište Baltazar Zaprešić- Baltazar Zaprešić Polytechnic

Sadržaj

ET²eR vol. VIII, br. 1, 2026., str. 104-111

Veleučilište u Virovitici

Stručni rad (Professional paper)
DOI: https://doi.org/10.70077/et2er.8.1.13

Programska implementacija 2AFC eksperimenta u PsychoPy okruženju uz praćenje pokreta miša

Programmatic Implementation of a 2AFC Experiment in PsychoPy with Mouse Tracking

Marko Malikoviću[1]

[1]*Filozofski fakultet Sveučilišta u Rijeci, Sveučilišna avenija 4, 51000 Rijeka, Hrvatska, marko@uniri.hr*

Sažetak

Programski jezik Python i PsychoPy omogućuju izradu i provođenje eksperimenata u raznim znanstvenim područjima uz visoku preciznost i fleksibilnost, pri čemu PsychoPy Builder olakšava implementaciju standardnih paradigmi, posebno u psihologiji, kognitivnim znanostima i neuroznanosti, bez potrebe za programerskim znanjem, dok se za složenije funkcionalnosti koristi programski pristup. U radu je prikazana implementacija 2AFC eksperimenta s dodatnim prikupljanjem podataka o pokretima miša te su istaknute prednosti kodiranog pristupa u odnosu na Builder, osobito u kontekstu nestandardnih i proširivih eksperimentalnih procedura. Poseban naglasak stavljen je na pristupačnost izrade eksperimenata korisnicima bez naprednog programerskog znanja, s ciljem približavanja osnovnih i naprednijih mogućnosti PsychoPy okruženja kroz praktičan primjer.

Ključne riječi

2AFC, eksperiment, pokreti računalnog miša, programiranje, PsychoPy

Abstract

Python programming language and PsychoPy enable the design and implementation of experiments across various scientific fields with high precision and flexibility, while the PsychoPy Builder facilitates the implementation of standard paradigms, particularly in psychology, cognitive science, and neuroscience, without the need for programming knowledge, whereas more complex functionality requires a programmatic approach. This paper presents the implementation of a 2AFC experiment with additional collection of mouse movement data and highlights the advantages of the coded approach over the Builder, especially in the context of non-standard and extensible experimental procedures. Special emphasis is placed on the accessibility of experiment development for users without advanced programming knowledge, with the aim of bridging basic and more advanced capabilities of the PsychoPy environment through a practical example.

Keywords

programming, 2AFC, experiment, PsychoPy, computer mouse movements

Uvod

Programski jezik Python koristi se za izradu eksperimenata u psihologiji, kognitivnim znanostima i neuroznanosti, ekonomiji i drugim društvenim znanostima, kao i u biologiji, medicini, računalnim znanostima, umjetnoj inteligenciji, marketingu i drugdje, gdje omogućuje provođenje, automatizaciju i analizu rezultata eksperimentalnih

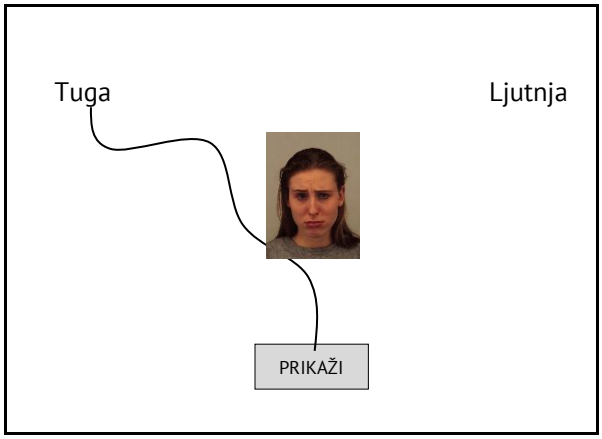
istraživanja. Također, s obzirom na to da ga često koriste neprogrameri, Python se za navedene namjene koristi zbog svoje jednostavne sintakse, fleksibilnosti i dostupnosti brojnih znanstvenih biblioteka razvijenih za obradu podataka, upravljanje eksperimentalnim postupcima, kontrolu i automatizaciju eksperimentalnih uvjeta, prikupljanje podataka tijekom istraživanja, analizu rezultata istraživanja i tako dalje.

Programsko okruženje PsychoPy je temeljeno na Pythonu i posebno omogućuje izradu i provođenje eksperimenata u područjima kao što su kognitivna psihologija, psihofizika i neuroznanost, u kojima je dodatno važna preciznost prezentacije podražaja i točnost mjerenja reakcija ispitanika (Peirce, 2007).

Primjer često korištenog eksperimentalnog postupka u različitim znanstvenim područjima je test prisilnog izbora s dvije alternative[1], čija će moguća programska implementacija u okruženju PsychoPy biti prikazana u ovom radu. U toj vrsti testa od ispitanika se u zadacima traži da od dvije ponuđene opcije odabere jednu, pri čemu ne postoji mogućnost da ne odgovori čak i kada nije siguran u odgovor (Leontyev, Yamauchi, 2021). Na primjer, u (Ely, Ambrus, 2025) ispitanicima su prikazivane fotografije lica s različitim emocionalnim izrazima, dok su ponuđeni odgovori bili zadani riječima (sretno, ljuto, tužno, neutralno). Zbog toga što svodi odluku na binarnu diskriminaciju i smanjuje utjecaj subjektivnih kriterija odgovora, ova se paradigma smatra standardnim pristupom za mjerenje perceptivne osjetljivosti i sposobnosti razlikovanja podražaja (Macmillan, Creelman, 2005). 2AFC testovi najčešće se koriste za ispitivanje detekcije i prepoznavanja podražaja te perceptivnih pragova u uvjetima nesigurnosti (Jogan, Stocker, 2014; Yeshurun, Carrasco, Maloney, 2008).

U 2AFC testovima se najčešće bilježe odgovori ispitanika, odnosno njihova točnost, kao i vrijeme potrebno za odgovor. Međutim, u slučaju kada se zadaci rješavaju pomoću računalnog miša, mogu se prikupljati i putanje i klikovi koje ispitanik radi mišem dok odgovara na pitanja. Zato su Spivey i sur. (2005) osmislili računalni eksperimentalni dizajn 2AFC testa koji se i danas često primjenjuje i u kojem se ispitaniku najprije na dnu ekrana prikazuje početni gumb (npr. "PRIKAŽI") na koji treba kliknuti mišem da bi se pojavili mogući odgovori. Nakon toga ispitanik daje odgovor klikom na jednu od dvije ponuđene opcije u lijevom i desnom gornjem dijelu ekrana. Na Slici 1 je prikazan primjer takvog testa zajedno s putanjom miša koju je učinio ispitanik za vrijeme odgovaranja. Svrha početnog gumba *PRIKAŽI* je ta da sve putanje miša počnu na istom mjestu, što kasnije omogućava njihovu zajedničku obradu i međusobno uspoređivanje.

SLIKA 1. RAČUNALNI DIZAJN 2AFC TESTA I PUTANJA MIŠA



Izvor: autor

Da bi u okruženju PsychoPy izrađivali ovakve eksperimente, korisnici mogu koristiti grafičko sučelje PsychoPy Builder ili implementirati vlastiti programski kôd koristeći Python i PsychoPy biblioteku za izradu složenijih funkcionalnosti i eksperimentalnih postupaka (Peirce i sur., 2019). PsychoPy Builder omogućava izradu eksperimenata bez pisanja programskog kôda, kroz vizualno slaganje komponenti na vremensku traku. Umjesto programiranja logike, petlji i slično, korisnik sve slaže grafički, dok se programski kôd automatski generira u pozadini. Zbog toga što omogućuje izradu eksperimenata bez znanja programiranja, Builder je posebno pogodan za neprogramere i početnike u programiranju.

Iako PsychoPy Builder omogućuje izradu širokog spektra psihologijskih eksperimenata, njegova primjena je primarno usmjerena na standardne eksperimentalne paradigme koje se mogu implementirati pomoću unaprijed definiranih komponenti, rutina i petlji. U skladu s time, njegova funkcionalnost je ograničena u odnosu na potpunu programsku fleksibilnost, pa je u složenijim slučajevima ipak potrebno pisati Python i PsychoPy programski kôd, za što se može koristiti PsychoPy Coder. U slučaju 2AFC eksperimenta se jedno od ograničenja Buildera odnosi na proširenja u smislu adaptivnih procedura u kojima se parametri podražaja (njihovo trajanje, intenzitet i sl.) dinamički mijenjaju ovisno o odgovorima ispitanika (Levitt, 1971; Kontsevich, Tyler, 1999). Takve procedure

[1] Engl. *Two-alternative forced choice - 2AFC.*

zahtijevaju kontinuirano računanje i ažuriranje parametara tijekom izvođenja eksperimenta, što prelazi okvire standardnog Builder okruženja i zahtijeva korištenje programskog kôda. Također, složenije upravljanje tijekom eksperimenta, poput uvjetnog grananja ovisno o odgovorima ispitanika ili generiranja novih uvjeta tijekom provođenja eksperimenta, nije u potpunosti podržano unutar Builder sustava (Peirce i sur., 2019). Nadalje, u Builderu je oblik izlaznih podataka eksperimenta uglavnom zadan i sustav ih automatski sprema u izlazne datoteke. Njihov oblik je u Builderu moguće dijelom ali ne u potpunosti prilagoditi pa je za detalje potrebno koristiti programski kôd. Zaključno, kao što se i navodi u tehničkoj dokumentaciji i opisu sustava, Builder je namijenjen bržoj implementaciji standardnih eksperimenata, dok se za veću fleksibilnost i nestandardne funkcionalnosti koristi integracija s kôdom (Peirce, 2024).

Opcija pisanja Python kôda u PsychoPy Coderu temelji se na korištenju PsychoPy biblioteke. Ona predstavlja skup gotovih funkcija i alata za prikaz podražaja, prikupljanje odgovora i upravljanje eksperimentalnim tijekom. Takav pristup omogućuje veću fleksibilnost i kontrolu nad programom te izradu prilagođenih funkcionalnosti koje je teško ili nemoguće izvesti u Builderu.

Osim pisanja kompletnog programa u Coderu, u Builderu postoji i mogućnost dodavanja samo manjih dijelova kôda kao nadopune grafički izrađenom eksperimentu putem opcije koja se zove Code Components. Takav pristup omogućuje kombiniranje jednostavnosti Buildera s fleksibilnošću programskog kôda, bez potrebe da se cijeli eksperiment implementira isključivo kroz kôd.

Cilj ovog rada je prikazati izradu 2AFC eksperimenta pisanjem Python i PsychoPy programskog kôda s prikupljanjem podataka o pokretima miša te istaknuti prednosti kodiranog pristupa u odnosu na Builder, posebno za korisnike bez programerskog iskustva.

Ostatak ovog rada podijeljen je u još tri cjeline. U prvom poglavlju se opisuje jedan od načina općenitog programskog prikupljanja podataka o pokretima miša kao samostalne funkcionalne komponente koja omogućuje neovisno prikupljanje i pohranu podataka o pokretima miša, u drugom poglavlju opisana je programska implementacija psihologijskog 2AFC eksperimenta, u koju je integrirana ta komponenta, dok se u posljednjem poglavlju navode najvažniji zaključci rada.

1. Programsko prikupljanje podataka o pokretima miša

U ovom poglavlju se najprije prikazuje općenita mogućnost prikupljanja pokreta miša u Python/PsychoPy programskom kôdu, što se kasnije primjenjuje u konkretnom eksperimentu. Prikazana programska rješenja čitatelj može direktno isprobati u PsychoPy Coder-u. Za razvoj programskog rješenja korišteno je okruženje PsychoPy (verzija 2026.1.2) i programski jezik Python (verzija 3.10.11).

Za praćenje pokreta računalnog miša potrebno je prije svega kreirati aktivan prozor u kojem se prati kretanje. Potrebno je učitati klasu *Window* iz modula *visual*, koji je dio biblioteke *PsychoPy*, a zatim stvoriti objekt te klase koji se može nazvati *prozor* i kojem se mogu zadati željena svojstva:

Isječak programskog koda 1. Učitavanje klase Window i stvaranje objekta

```
from psychopy.visual import Window
prozor = Window(size=(800,600), color=(1,1,1),
units='pix', checkTiming=False, fullscr=False)
```

Detaljan opis dostupnih parametara klase *Window* kao i drugih klasa biblioteke *psychopy* koje se koriste u ovom radu može se naći u službenoj dokumentaciji biblioteke PsychoPy (Peirce i sur., 2026).

Iako operacijski sustav kontinuirano prati pokrete miša radi upravljanja korisničkim sučeljem, PsychoPy ih sam po sebi ne bilježi dok to nije definirano u programu, jer te informacije ne dobiva automatski od operacijskog sustava. Zato je potrebno najprije iz PsychoPy modula *event* učitati klasu *Mouse*, a zatim stvoriti objekt te klase (može ga se nazvati *mis*) koji omogućuje njegovo praćenje i povezati ga s kreiranim prozorom nad čijim koordinatama će se miš pratiti:

Isječak programskog koda 2. Učitavanje klase Mouse i stvaranje objekta

```
from psychopy.event import Mouse
mis = Mouse(win=prozor)
```

Kao što će biti objašnjeno, objekt *mis* će omogućiti dohvaćanje i spremanje podataka o koordinatama miša posebnom metodom klase *Mouse*. Prije toga, da bi se uz koordinate miša mogao očitati i spremiti podatak o vremenu u kojem je miš bio na određenoj poziciji, potrebno je učitati klasu *Clock* iz modula *core* i kreirati objekt *sat* koji će omogućiti bilježenje vremena tijekom eksperimenta:

Isječak programskog koda 3. Učitavanje klase Clock i stvaranje objekta

```
from psychopy.core import Clock
sat = Clock()
```

Za dobivanje koordinata miša u određenom trenutku izvršavanja programa koristi se metoda *getPos()* klase *Mouse*. Ako se želi kontinuirano pratiti položaj miša tijekom određenog razdoblja, to se može postići kontinuiranom petljom koja pri svakom izvršavanju primjenjuje metodu *getPos()*. Ako se želi da period praćenja miša iznosi 5 sekundi, može se upotrijebiti petlja *while* sa uvjetom *sat.getTime()<5* koji koristi metodu *getTime()* klase *Clock* da bi se pri svakom prolasku kroz petlju izvršila provjera je li proteklo 5 sekundi od pokretanja programa. Prije petlje treba stvoriti praznu Python listu *pokreti* koja služi tome da se u nju tijekom izvođenja petlje spremaju podatci:

Isječak programskog koda 4. Praćenje i bilježenje pokreta miša

```
pokreti = []
while sat.getTime() < 5:
 pozicija = mis.getPos()
 x = pozicija[0]
 y = pozicija[1]
 t = sat.getTime()
 pokreti.append((float(t),float(x),float(y)))
 prozor.flip()
prozor.close()
```

Prvom naredbom u petlji se u varijablu *pozicija* spremaju koordinate miša u obliku *n*-torke od dva člana *x* i *y* (dakle, u obliku uređenog para). Zatim slijede dvije naredbe pomoću kojih varijable *x* i *y* preuzimaju pojedinačne elemente iz *n*-torke. Nakon toga se u varijablu *t* bilježi koliko je vremena prošlo od pokretanja programa. Na listu *pokreti* se metodom *append()* dodaju vrijednosti *t*, *x* i *y*. Iz kôda se vidi da se ti brojevi ne dodaju kakvi jesu nego se prije toga funkcijom *float()* pretvaraju u decimalne brojeve. To u većini slučajeva i nije potrebno i neće imati nikakvog učinka jer su očitano vrijeme i koordinate miša i inače decimalni brojevi. Ipak, u nekim slučajevima može doći do neočekivanih vrijednosti. Na primjer, PsychoPy u pozadini često koristi posebne biblioteke za brže računanje, npr. biblioteku NumPy (NumPy developers, 2025), pa metode poput *getTime()* ili *getPos()* ponekad vrate broj u NumPy formatu (oblika numpy.float64(1.234) umjesto 1.234). U tom slučaju će funkcija *float()* pretvoriti broj u očekivani oblik. Posljednja naredba u petlji je *prozor.flip()*, koja osvježava prikaz na ekranu. Iako u ovom programu ta naredba neće imati izravno vidljiv učinak, odnosno neće prikazivati novi sadržaj jer program ništa ne prikazuje, važna je jer održava prozor aktivnim i usklađuje očitavanje koordinata miša i vremena s ritmom osvježavanja ekrana. Bez te naredbe prozor bi se mogao zamrznuti, operacijski sustav mogao bi ga označiti kao neaktivnog, a vrijeme izvođenja moglo bi biti nekonzistentno. Metoda *flip()* određuje ritam izvođenja programa tako da se ekran, primjerice, osvježava 60 puta u sekundi, čime se i izvođenje petlje sinkronizira s tim ritmom. To znači da se dobiva stabilnije vremensko uzorkovanje, odnosno ujednačenije izmjerene koordinate i vrijeme. Nakon što petlja završi s radom, prozor se zatvara metodom *close()*.

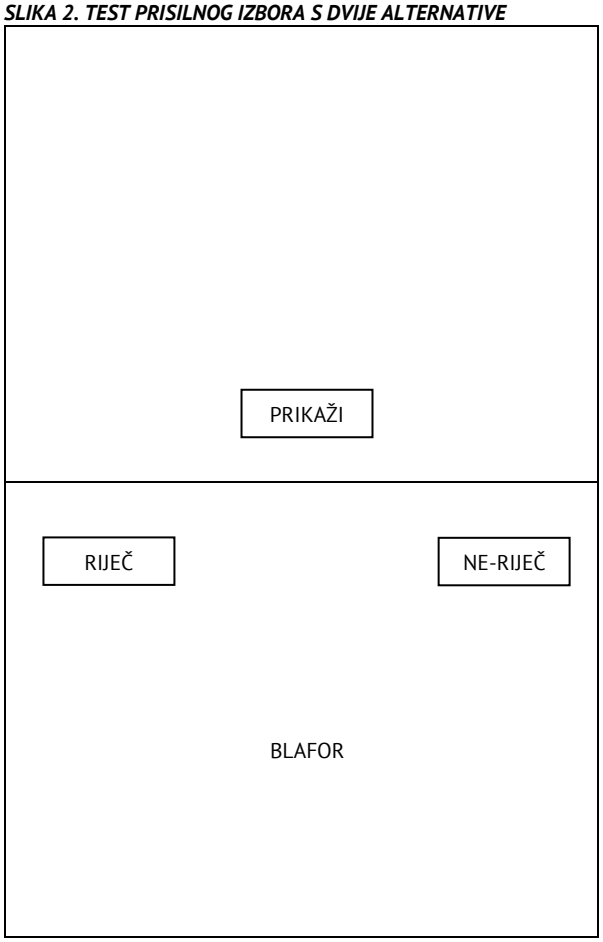
Kada se program izvede, u listi *pokreti* bit će zapisani podaci u obliku niza uređenih trojki kao na primjer:

```
[(0.03592910000588745, 18.0, 49.0), (0.0498719000024721, 29.0, 44.0),
(0.06711459998041391, 34.0, 43.0), (0.08451240009162575, 36.0, 41.0),
(0.09970640006940812, 36.0, 41.0), (0.11754020000807941, 48.0, 41.0),
(0.13302350009325892, 84.0, 41.0), (0.14979220007080585, 149.0,
41.0), (0.16784590005408973, 204.0, 40.0),…
```

2. Eksperiment

Kao primjer eksperimenta s testom prisilnog izbora s dvije alternative (2AFC), odabran je zadatak leksičke odluke po uzoru na klasičan eksperiment opisan u (Meyer, Schvaneveldt, 1971). To je primjer testa u kojem se ispitanicima na ekranu prikazuju riječi i takozvane ne-riječi. Ne-riječ je niz slova koja izgleda kao riječ ali zapravo nije dio jezika (na primjer "blafor"). U najjednostavnijoj verziji tog eksperimenta, ispitanik treba za svaki od ponuđenih nizova slova što brže i točnije odgovoriti predstavlja li taj niz stvarnu riječ ili ne-riječ. Na dnu ekrana se, iz razloga koji su objašnjeni u uvodu ovog rada, prije svakog zadatka prikazuje početni gumb PRIKAŽI na kojeg ispitanik treba kliknuti mišem da bi se pojavio podražaj. Ponuđeni odgovori (Riječ i Ne-riječ) se uz podražaj pojavljuju u gornjem lijevom i gornjem desnom dijelu ekrana, a ispitanik daje odgovor klikom na jednu od te dvije opcije. U svakom zadatku bilježe se odgovor ispitanika, vrijeme reakcije i putanja računalnog miša od klika na gumb PRIKAŽI do klika na jedan od odgovora. Na Slici 2 gore vidi se izgled ekrana s početnim gumbom PRIKAŽI, a na Slici 2 dolje podražaj i ponuđeni odgovori.

SLIKA 2. TEST PRISILNOG IZBORA S DVIJE ALTERNATIVE



Izvor: autor

Programska verzija eksperimenta će sadržavati osam riječi (*stolica*, *prozor*, *knjiga*, *more*, *cesta*, *sunce*, *cvijet*, *grad*) i osam ne-riječi (*blafor*, *mepra*, *slonak*, *drimel*, *zernak*, *vudra*, *klenor*, *trefin*) koje će se na ekranu pojavljivati slučajnim redoslijedom.

Slijedi opis cijelog programa. Najprije se iz PsychoPy modula *visual*, *core* i *event* i standardnog Python modula *random* učitavaju potrebne klase za crtanje prozora, rad s vremenom, rad s mišem, kreiranje podražaja u obliku gumba i teksta te funkcija za nasumično miješanje elemenata u listi:

Isječak programskog koda 5. Učitavanje potrebnih klasa i funkcija

```
from psychopy.visual import Window, ButtonStim,
TextStim
from psychopy.core import Clock
from psychopy.event import Mouse
from random import shuffle
```

Nakon toga se kreira prozor određene veličine, boje i drugih osobina kao i objekti klasa *Mouse* i *Clock* koji će omogućiti praćenje miša i vremena kako je objašnjeno u Poglavlju 1:

Isječak programskog koda 6. Kreiranje potrebnih objekata klasa Window, Mouse i Clock

```
prozor = Window(size=(800,600), color=(1,1,1),
units='pix', checkTiming=False, fullscr=False)
mis = Mouse(win=prozor)
sat = Clock()
```

Da bi se na ekranu mogli prikazati gumbi PRIKAŽI, RIJEČ i NE-RIJEČ, potrebno je najprije kreirati objekte klasa *ButtonStim* s parametrima koji određuju njihova svojstva (tekst, boju teksta, poziciju, veličinu, boju gumba i drugo prema potrebi):

Isječak programskog koda 7. Kreiranje objekata klase ButtonStim

```
gumbPrikazi = ButtonStim(prozor,
text='PRIKAŽI', color='white', pos=(0, -200),
size=(150, 50), fillColor='blue')
gumbRijec = ButtonStim(prozor, text='RIJEČ',
color='white', pos=(-250, 200), size=(150, 50),
fillColor='blue')
gumbNerijec = ButtonStim(prozor, text='NE-
RIJEČ', color='white', pos=(250, 200),
size=(150, 50), fillColor='blue')
```

Riječi i ne-riječi će se zapisati u listu, a nakon toga će njihov redoslijed biti izmiješan po slučaju funkcijom *shuffle()*. Također, pokreti miša i odgovori ispitanika će se zapisivati u liste pa je prije toga potrebno kreirati ih kao prazne liste:

Isječak programskog koda 8. Kreiranje potrebnih lista i miješanje liste podražaja

```
podrazaji = ["stolica", "prozor", "knjiga",
"more", "cesta", "sunce", "cvijet", "grad",
"blafor", "mepra", "slonak", "drimel",
"zernak", "vudra", "klenor", "trefin"]
shuffle(podrazaji)
pokreti = []
odgovori = []
```

Slijedi glavni dio programa u obliku *for* petlje koja će iz izmiješane liste *podrazaji* uzimati jedan po jedan element odnosno riječ ili ne-riječ i dodjeljivati je varijabli *podrazaj*. U *while* petlji koja je ugniježđena u glavnoj *for* petlji će program na ekranu prikazati gumb PRIKAŽI i čekati klik ispitanika pomoću metode *isPressedIn()*. U trenutku kada se registrira klik miša bit će očitane i zabilježene njegove koordinate i vrijeme proteklo od početka programa. Nakon toga će rad prve *while* petlje biti prekinut naredbom *break*. Sadržaj ekrana se zatim briše naredbom *prozor.flip()* i pokreće se nova *while* petlja u kojoj se prikazuje podražaj i dva gumba: RIJEČ i NE-RIJEČ. Petlja čeka klik miša na neki od gumba s odgovorom i taj odgovor se zapisuje na listu *odgovori*. Također, podaci o koordinatama miša i vrijeme se, kako je opisano u Poglavlju 1, kontinuirano zapisuju u za to pripremljenu listu

putanja. Prilikom svakog klika na gumb PRIKAŽI, vrijeme se resetira kako bi svaka nova putanja počela od nule:

Isječak programskog koda 9. Glavni dio programa

```
for podrazaj in podrazaji:
 putanja = []
 while True:
  gumbPrikazi.draw()
  if mis.isPressedIn(gumbPrikazi):
   pozicija = mis.getPos()
   x = pozicija[0]
   y = pozicija[1]
   sat.reset()
   t = sat.getTime()
   putanja.append((float(t),float(x),float(y)))
   break
  prozor.flip()
 tekst = TextStim(prozor, text=podrazaj,
color='black')
 while True:
  pozicija = mis.getPos()
  x = pozicija[0]
  y = pozicija[1]
  t = sat.getTime()
  tekst.draw()
  gumbRijec.draw()
  gumbNerijec.draw()
  if gumbRijec.isClicked:
   odgovori.append("rijec")
   putanja.append((float(t),float(x),float(y)))
   break
  elif gumbNerijec.isClicked:
   odgovori.append("ne-rijec")
   putanja.append((float(t),float(x),float(y)))
   break
  putanja.append((float(t),float(x),float(y)))
  prozor.flip()
 pokreti.append(putanja)
prozor.close()
```

Na kraju svake iteracije *for* petlje, putanja se zapisuje kao zaseban element na listu *pokreti* koja će sadržavati sve pokrete miša tijekom izvršavanja programa. Kada se prikažu svi podražaji, petlja *for* će završiti s radom, a metoda *close()* će zatvoriti prozor na ekranu.

Prikupljeni podaci o pokretima miša odnosno putanje koje čini ispitanik dok rješava zadatke se mogu obrađivati i analizirati u specijaliziranim programima kao što je na primjer Mousetrap (Wulff i sur., 2025). Takvih programa ima više ali svaki zahtijeva da se datoteka s podacima pripremi u obliku koji se može direktno učitati u takav program. Jedna od pogodnih struktura datoteke za program Mousetrap je *CSV* (*Comma-Separated Values*) u kojoj svaki redak predstavlja jednu putanju miša i u kojima imamo redom: redni broj putanje, niz vremena u sekundama odvojenih razmakom, niz *x*-koordinata odvojenih razmakom i niz *y*-koordinata također odvojenih razmakom. U prvom retku datoteke treba biti zaglavlje s nazivima stupaca:

```
trial_id,timestamps,xpos,ypos
1,0.02 0.04 0.06 0.07 0.11 0.13 0.14,0 3 7 12 18 25 33,0 15 30 45 60 75 90
2,0.01 0.02 0.03 0.05 0.06 0.08 0.10,0 4 9 15 22 30 39,0 16 32 48 64 80 96
...
```

Moguće su i nešto drugačije strukture. Datoteku u kojoj će podaci biti zapisani na takav način je u našem programu moguće kreirati i napuniti sljedećim programskim kôdom:

Isječak programskog koda 10. Kreiranje i punjenje datoteke s podacima

```
dat = open("pokreti_mis.csv", "a")
dat.write("trial_id,timestamps,xpos,ypos")
dat.write("\n")
for i in range(len(pokreti)):
 dat.write(str(i))
 dat.write(",")
 for k in range(3):
  for j in range(len(pokreti[i])):
   dat.write(str(pokreti[i][j][k]))
   dat.write(" ")
  dat.write(",")
 dat.write("\n")
dat.close()
```

Dakle, najprije se otvara datoteka *pokreti_mis.csv* s mogućnošću nadopunjavanja (atribut *a* odnosno *append* metode *open*) i u nju se upisuje prvi redak s nazivima stupaca. Zatim se pokreće *for* petlja čija kontrolna varijabla *i* poprima redom sve cjelobrojne vrijednosti od 0 do ukupnog broja podražaja u programu. Naredbe u toj petlji zapisuju podatke iz liste *pokreti* u obliku koji je pogodan za obradu u programu *Mousetrap*. Na kraju se datoteka zatvara metodom *close()* kako bi Python dovršio zapisivanje podataka na disk i oslobodio resurse koje je koristio tijekom rada s datotekom.

Opisanim se kôdom u datoteku upisuju samo pokreti miša. Liste prikazanih podražaja i odgovora ispitanika, koje se prikupljaju tijekom izvođenja programa, mogu se zapisati u istu ili zasebnu datoteku te uključiti u analizu.

Prikazana implementacija prilagođena je početnicima u programiranju te je usmjerena na jasnoću i preglednost kôda. Također, predstavlja temeljnu strukturu eksperimenta pogodnu za realizaciju u PsychoPy Builder okruženju bez potrebe za opsežnijim programiranjem, osim u dijelu prilagođenog zapisivanja rezultata, koje bi ipak zahtijevalo dodatne programske komponente. Međutim, implementacija nekih naprednijih procedura, od kojih su neki spomenuti u uvodu, zahtijevala bi složeniju kontrolu tijeka eksperimenta te bi nadilazila mogućnosti realizacije isključivo pomoću Builder sučelja pa bi bilo potrebno napisati dodatni programski kôd. Na primjer, da bi se u

postojeći program uvela adaptivna procedura kojom bi se vremensko ograničenje za odgovaranje mijenjalo ovisno o točnosti prethodnog odgovora, prvo bi bilo potrebno promijeniti samu logiku dijela programa u kojem ispitanik daje odgovor. U trenutnoj verziji program koristi beskonačnu petlju oblika *while True*, koja se prekida isključivo klikom na odgovor. Takvu strukturu bilo bi potrebno zamijeniti petljom koja uz ponašanje ispitanika uzima u obzir i proteklo vrijeme od početka prikaza podražaja, jer adaptacija zahtijeva i informaciju o tome je li odgovor dan prije isteka vremenskog ograničenja. U praksi to znači da je, odmah nakon linije u kojoj se resetira sat naredbom *sat.reset()* i započinje mjerenje vremena, potrebno dodati varijablu koja definira vremensko ograničenje i promijeniti uvjet u petlji *while*:

Isječak programskog koda 11. Promjena uvjeta prekida zadatka

```
vrijeme_limit = 3.0
while sat.getTime() < vrijeme_limit
```

Unutar te petlje ostaje dio koji prikazuje podražaj i provjerava klikove na gumbe ali se sada petlja automatski prekida i ako ispitanik ne odgovori na vrijeme. Nakon nje potrebno je dodati logiku koja razlikuje dva ishoda: odgovor unutar vremena ili izostanak odgovora. Stoga se dodaje uvjet:

Isječak programskog koda 12. Naredba if koja razlikuje dva ishoda

```
if sat.getTime() >= vrijeme_limit:
 odgovori.append("bez_odgovora")
```

Kako bi se implementirala adaptivna komponenta, potrebno je nakon svake iteracije petlje ažurirati vrijednost varijable *vrijeme_limit*. Zato se nakon linije *pokreti.append(putanja)* dodaje adaptivno pravilo:

Isječak programskog koda 13. Implementacija adaptivnog pravila

```
if odgovori[-1] == "bez_odgovora":
 vrijeme_limit = vrijeme_limit + 0.3
 if vrijeme_limit > 5.0:
  vrijeme_limit = 5.0
else:
 vrijeme_limit = vrijeme_limit - 0.2
 if vrijeme_limit < 1.5:
  vrijeme_limit = 1.5
```

To pravilo dinamički mijenja vremensko ograničenje za odgovor na temelju uspješnosti ispitanika u prethodnom pokušaju. Ako ispitanik ne da odgovor unutar zadanog vremena, program povećava dopušteno vrijeme za sljedeći pokušaj za 0.3 sekunde ali najviše do gornje granice od 5 sekundi. Time se ispitaniku daje više vremena kada pokazuje poteškoće u pravovremenom odgovaranju. Ako ispitanik uspješno odgovori unutar zadanog vremena, vremensko ograničenje se smanjuje za 0.2 sekunde ali ne može pasti ispod donje granice od 1.5 sekundi. Na taj način se postupno povećava vremenski pritisak kada ispitanik stabilno i uspješno izvršava zadatak. Ukupno gledano, ovo pravilo stvara jednostavan adaptivni mehanizam koji održava zadatak na približno optimalnoj razini težine. U skladu s principima adaptivnog testiranja u psihologiji (Wainer, 2000; van der Linden, Glas, 2010), ispitanici koji sporije reagiraju dobivaju više vremena, dok se ispitanicima koji brzo i točno odgovaraju postupno smanjuje dostupno vrijeme, čime se povećava zahtjevnost zadatka. U ovakvoj adaptivnoj varijanti bilo bi posebno zanimljivo promatrati pokrete miša jer bi promjene u vremenskom pritisku mogle utjecati na brzinu, preciznost i strukturu kretanja, što omogućuje uvid u dinamiku donošenja odluka u uvjetima različite razine kognitivnog opterećenja.

Uvođenjem ove strukture, kao samo jednog od mogućih primjera proširenja eksperimenta, mijenja se i njegova priroda: umjesto fiksnog vremenskog prikaza ili neograničenog odgovaranja, dobiva se dinamički sustav koji se prilagođava ponašanju ispitanika tijekom izvođenja. Takve adaptivne varijante eksperimenta teško je realizirati isključivo u PsychoPy Builder okruženju bez dodatnog programskog kôda, jer Builder sam po sebi ne omogućuje jednostavno mijenjanje trajanja prikaza ili drugih parametara u stvarnom vremenu na temelju ishoda prethodnih pokušaja.

3. Zaključak

PsychoPy predstavlja moćan i fleksibilan alat za izradu psihologijskih eksperimenata, pri čemu PsychoPy Builder omogućuje jednostavnu implementaciju standardnih paradigmi bez potrebe za programerskim znanjem. Međutim, kod složenijih eksperimentalnih postupaka javlja se potreba za korištenjem programskog kôda u PsychoPy Coderu ili kroz Code Components unutar Buildera.

U radu je prikazano kako Python kôd i PsychoPy biblioteka omogućuju visoku fleksibilnost i kontrolu nad eksperimentalnim tijekom, što je posebno korisno kod istraživanja koja zahtijevaju nestandardne postupke prikupljanja podataka. Kod složenijih varijanti eksperimenata, poput detaljnog praćenja pokreta miša, dinamičke obrade podataka ili prilagodbe tijeka eksperimenta, ograničenja Builder okruženja postaju izraženija, pa je programski pristup često nužan.

Također, jedan od ciljeva ovog rada bio je kroz praktični primjer približiti izradu PsychoPy kôda korisnicima koji nemaju napredno programersko znanje te ponuditi jasnije i prilagodljivije upute na hrvatskom jeziku.

Literatura

[1] Ely, M. M., Ambrus, G. G. (2025): *Shared neural dynamics of facial expression processing. Cognitive Neurodynamics* Vol 19 br. 45 (1-18)

[2] Jogan, M., Stocker, A. A. (2014): *A new two-alternative forced choice method for the unbiased characterization of perceptual bias and discriminability. Journal of Vision* Vol 14 br. 3 (1-18)

[3] Kontsevich, L. L., Tyler, C. W. (1999): *Bayesian adaptive estimation of psychometric slope and threshold. Vision Research* Vol 39 br. 16 (2729-2737)

[4] Leontyev, A., Yamauchi, T. (2021): *Discerning Mouse Trajectory Features With the Drift Diffusion Model. Cognitive Science* Vol 45 br. 10:e13046

[5] Levitt, H. (1971): *Transformed up-down methods in psychoacoustics. Journal of the Acoustical Society of America* Vol 49 br. 2B (467–477)

[6] Macmillan, N. A., Creelman, C. D. (2005): *Detection theory: A user's guide (2nd ed.)*. Lawrence Erlbaum Associates Publishers

[7] Meyer, D. E., Schvaneveldt, R. W. (1971): *Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. Journal of Experimental Psychology*, Vol 90 br. 2 (227-234)

[8] NumPy developers (2025): *NumPy*, https://numpy.org (13.05.2026.)

[9] Peirce, J. W. (2007): *PsychoPy-Psychophysics software in Python. Journal of Neuroscience Methods* Vol 162 br. 1-2 (8-13)

[10] Peirce, J. W. (2024): *Build or code?* PsychoPy blog, https://blog.psychopy.org/psychopy/build-or-code (15.05.2026.)

[11] Peirce, J. W., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman, E., Lindeløv, J. K. (2019): *PsychoPy2: Experiments in behavior made easy. Behavior Research Methods* Vol 51 (195-203)

[12] Peirce, J. W., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman, E., Lindeløv, J., Hoggett, D. J. K. (2026): *PsychoPy Reference Manual (API)*, https://psychopy.org (13.05.2026.)

[13] Spivey, M. J., Grosjean, M., Knoblich, G. (2005): *Continuous attraction toward phonological competitors*. In: *Proceedings of the National Academy of Sciences of the United States of America*, 19 July, Vol 102 br. 29:10393-8

[14] van der Linden, W. J., Glas, C. A. W. (Eds.). (2010): *Elements of adaptive testing*. Springer.

[15] Wainer, H. (Ed.). (2000): *Computerized adaptive testing: A primer*. Lawrence Erlbaum Associates

[16] Wulff, D. U., Kieslich, P. J., Henninger, F., Haslbeck, J. M. B., Schulte-Mecklenbeck, M. (2025): *Movement tracking of psychological processes: A tutorial using mousetrap. Behavior Research Methods*, Vol 57

[17] Yeshurun, Y., Carrasco, M., Maloney, L. T. (2008): *Bias and sensitivity in two-interval forced choice procedures: Tests of the difference model. Vision Research* Vol 48 br. 17 (1837-1851)