MOONHACK 2020

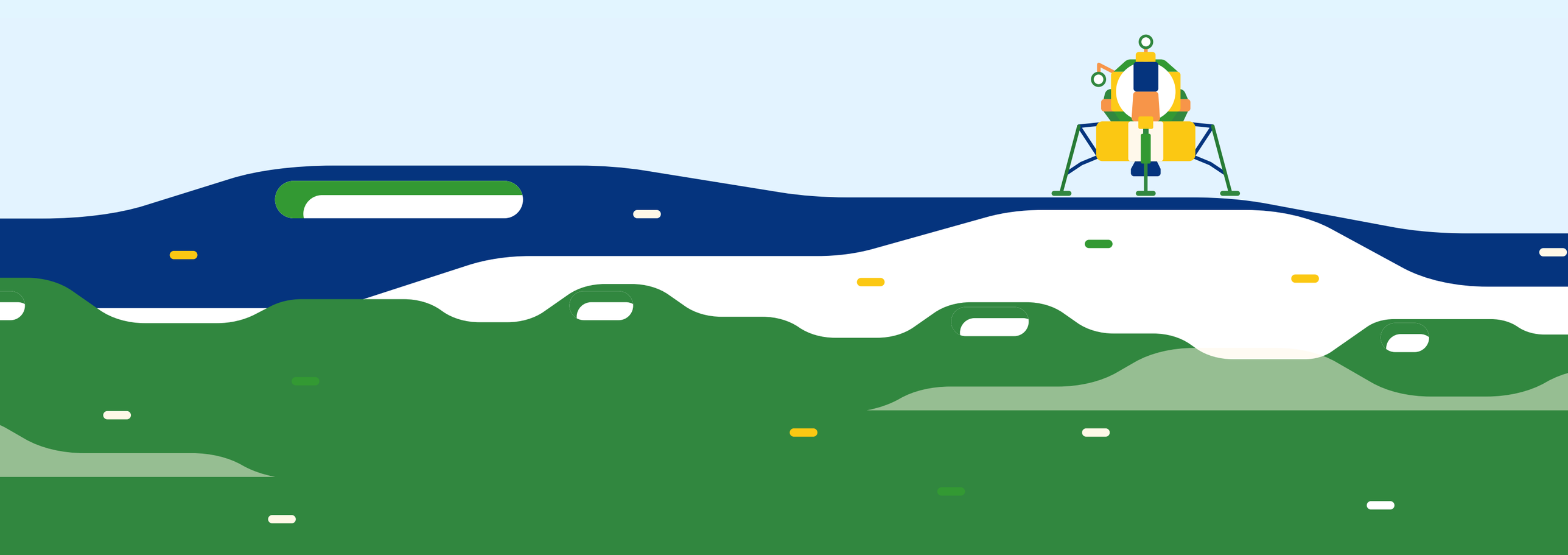
PYTHON WATER ON MARS

CROATIAN

BROUGHT TO YOU BY CODE CLUB AUSTRALIA
POWERED BY TELSTRA FOUNDATION

code club / AUSTRALIA | POWERED BY TELSTRA FOUNDATION

SUBMIT AND BE COUNTED AT
MOONHACK.COM



Voda na Marsu - Moonhack Python Projekt

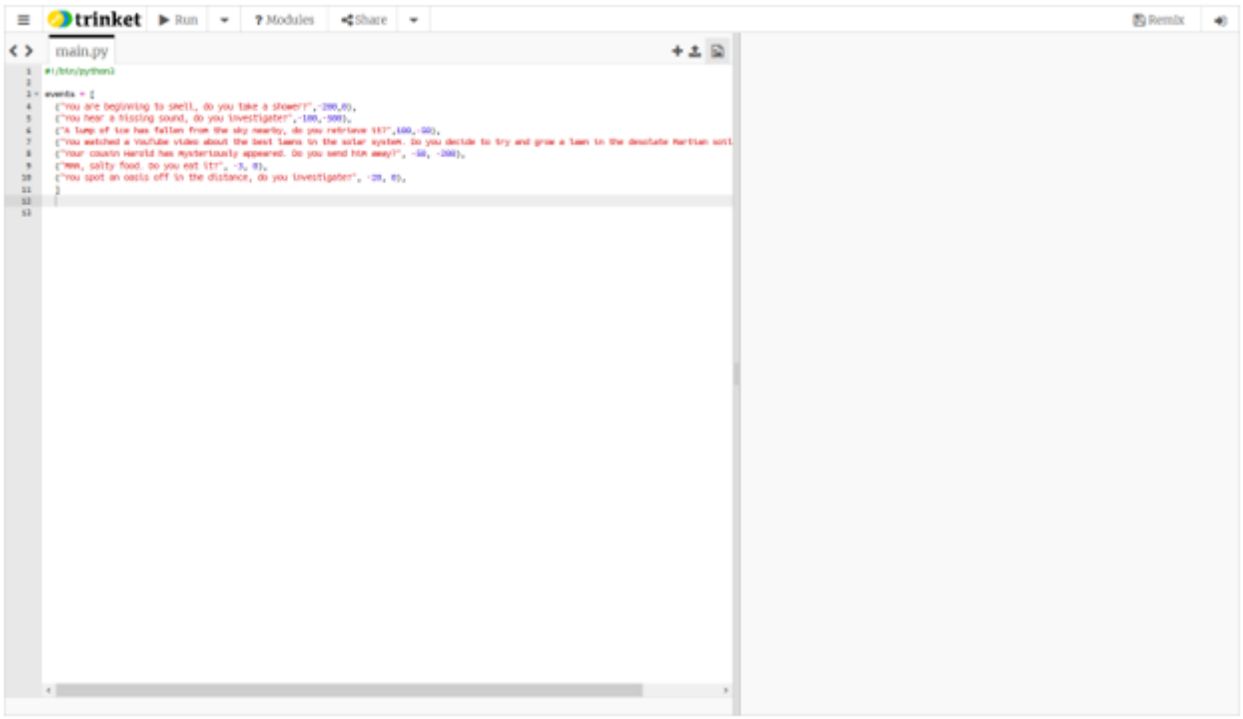
Uvod

Voda je nevjerojatno dragocjen resurs koji je ključan za život. Očuvanje vode važno je i postati će važnije tek kada naše vrste odu u Svemir. Danas ćemo napraviti tekstualnu igru za preživljavanje, u kojoj igrač mora donositi svjesne odluke kako bi opstao što duže, a da na Marsu ne ostane vode.

1. korak: Praćenje vode

U ovom prvom koraku započeti ćemo s našom igrom i postaviti varijablu pomoću koje možemo pratiti koliko vode imamo.

• Otvorite projekt na bit.ly/pyth onwater. Vaš bi prozor trebao izgledati ovako:



S lijeve strane je prozor vašeg koda; primijetit ćete da ovdje već postoji neki kod (na to ćemo se vratiti kasnije). S desne strane je vaš prozor za izvršenje; ovdje će se prikazati ishod vaše igre. Na vrhu je gumb za pokretanje; pomoću ovog gumba pokrenut ćete svoj kod.

Moramo dati našem igraču malo vode za početak. Krenimo s lijepim okruglim brojem od 1000 L Dodajte sljedeći označeni kod na dnu programa:

```
10    ("You spot an oasis
11    ]
12
13  water = 1000
```

• Želimo da naš korisnik nastavi igrati igru dok god ima vode. Kad im ponestane vode, igra će završiti. Kreirajmo našu glavnu petlju igre, koja će se nastaviti sve dok korisnik ima vode:

```
13  water = 1000
14
15  while water > 0:
```

Primijetite debelo crijevo (:) na kraju našeg retka. Ovaj se mali znak može činiti beznačajnim, ali ako ga zaboravimo, Python će se žaliti!

• Zatim dodajmo malo upotrebe vode u naš glavni krug za igru. Ovo će biti količina koja se oduzima svaki dan, neovisno o igračevom izboru, jer igrač treba piti.

Dodajte sljedeći kod unutar petlje:

```
15  while water > 0:
16      water -= 5
```

Primjetite dva prostora ispred retka 16 To nazivamo uvlakom. Ovo govori pythonu da je upute na toj liniji unutar petlje. Drugim riječima, to se događa svaki put kad obilazimo glavnu petlju igara (ako ste koristili Scratch, slična je narančastim blokovima).

• Sada je vrijeme za neki izlaz. Recimo korisniku da je igra završila kada im ponestane vode. Dodajte sljedeći kod izvan vaše glavne petlje igre (ne zaboravite da biste se riješili uvlačenja!):

```
15  while water > 0:
16      water -= 5
17
18  print ("Game Over :(")
```

• Pokrenite svoj kod. Što se događa?

• Python će jednostavno pokušati pokrenuti naš kod što je brže moguće. Bez ikakvog izlaza u glavnoj petlji, ne možemo znati što se događa dok ne dođemo do kraja. Dodajmo neki izlaz u našu glavnu petlju.

```
15 while water > 0:
16   water -= 5
17   print("A day has passed.")
18
19 print ("Game Over :(")
```

• Ponovno pokrenite svoj kod. Sada biste trebali vidjeti sve dane koji prođu prije nego što igra završi.

• Promijenimo zadnji redak koda koji smo umetnuli da nam kaže koliko smo vode dodali. Vrijednost naših varijabli možemo kombinirati s tekstom koji ispisujemo pomoću znaka +.

```
15 while water > 0:
16   water -= 5
17   print("A day has passed. Your Water " + water + "L")
18
19 print ("Game Over :(")
```

• Ponovno pokrenite svoj kod. Jeste li dobili pogrešku?

• Veliki dio programiranja pokušava otkriti zašto radimo pogreške koje dobijamo. U ovom slučaju, pogreška nam govori da "ne možemo objediniti" str "i" int "predmete". Prilikom programiranja često ćete morati razraditi što znače ove neobične poruke o pogrešci.

Ova nam greška govori da pokušavamo kombinirati dva različita tipa podataka koje Python ne zna kombinirati, tekst (string (skraćen na str)) i broj koji koristimo za praćenje naše vode (cijeli broj, skraćeno na int). Da bismo to riješili, možemo reći Pythonu da pretvara naš cijeli broj u niz sa sljedećim kodom:

• Pokrenite svoj kod. Trebali biste dobiti izlaz koji će nam reći koliko vode koristimo svaki dan prije nego

```
15 while water > 0:
16   water -= 5
17   print("A day has passed. Your Water " + str(water) + "L")
18
19 print ("Game Over :(")
```

što liku igrača ponestane vode.

Izazov: Promijenite svakodnevno korištenje vode

Trenutno koristimo 5 litara vode dnevno. Što mislite, koji je apsolutni minimum koji je čovjeku potreban da bi preživio? Možete li promijeniti kod tako da igrač koristi tu količinu vode?

2. korak: praćenje dana

Do sada smo pratili količinu vode koju je igrač upotrijebio, ali da bismo vidjeli koliko je igrač dugo ostao na Marsu, pratit ćemo broj dana koji su prošli.

• Prvo ćemo postaviti početnu vrijednost varijable day koje predstavlja broj dana, počevši od prvog dana.

```
13  water = 1000
14  day = 1
15
16  while water > 0:
```

• Zatim, svaki dan koji prođe, želimo povećati vrijednost vrijable day za jedan. Dodajte sljedeći kod u svoj glavni krug igre:

```
16 while water > 0:
17   water -= 5
18   print("A day has passed. Your Water " + str(water) + "L")
19   day += 1
20
21 print ("Game Over :(")
```

• Na kraju, izvijestit ćemo igrača koliko je dugo ostao na Marsu. To ćemo učiniti zamjenom konačne izjave, izjavom koja uključuje broj dana koji su prošli.

```
19    day += 1
20
21  print("you lasted " + str(day) + " days")
```

Primjetite da smo za pretvorbu broja u tekst morali ponovo koristiti str (day).

• Pokrenite svoj kod. Trebali biste dobiti izlaz za svaki dan, a zatim konačni rezultat koji kaže koliko dana je igrač ostao na Marsu.

Izazov: Svakodnevno upišite broj dana

Trenutno koristimo samo varijablu day na samom kraju. Možete li promijeniti kod tako da svakodnevno šaljete broj dana?

Trenutno će izlaz reći:

A day has passed. Your Water 950L.

Možete li to promijeniti tako da kaže:

Day: 10. Your water 950L.

Korak 3: Vaš svakodnevni događaj

Sada ćemo dodati malo igranja. Svakodnevno ćemo dati igraču izbor, a taj će izbor odrediti koliko vode izgube ili dobiju.

• Vjerojatno ste primijetili da na početku postoji čitava gomila koda koje niste napisali. Ovo je popis događaja koji bi se mogli dogoditi na Marsu. Dodajmo još jednu izjavu o ispisu u našu glavnu petlju. Ovo će pristupiti našem popisu događaja i odabrati nasumično.

```
18    print("A day has passed. Your Water " + str(water) + "L")
19    print(random.choice(events))
20    day += 1
```

O ne! Još jedna pogreška. Ova kaže da "ime" random "nije definirano". To je zato što random nije dio Pythona, već je dio modula. Moramo ga uvesti da bismo ga mogli koristiti.

```
1   #!/bin/python3
2   import random
3
4   events = [
```

• Pokrenite svoj kod. Svakog dana trebali biste ispisati slučajni događaj.

• Morat ćemo taj dan upotrijebiti više puta. Ako bismo koristili random.choice (događaje) svaki put kada bismo željeli upotrijebiti događaj, dobili bismo drugačiji događaj! Umjesto toga, pohranimo taj događaj u varijablu zvanu događaj, koju možemo koristiti više puta bez brige.

```
19    print("A day has passed. Your Water " + str(water) + "L")
20    event = random.choice(events)
21    print(event)
22    day += 1
```

• Ponovno pokrenite svoj kod. Trebali biste dobiti sličan rezultat kao prethodni put.

• Vjerojatno ste primijetili da je događaj u našem izlazu okružen hrpom dodatnih informacija i simbola. To je zato što spremamo pitanje i vrijednosti za količinu vode koja se može dobiti ili izgubiti ako igrač na isto mjesto odgovori da ili ne. Doista želimo postaviti pitanje igraču bez da im pokažemo dodatne informacije. Da bismo pristupili pitanju, možemo koristiti ono što se naziva indeksacija. Indeks je samo položaj na kojem element sjedi, počevši od 0. Promijenite liniju ispisa na sljedeće:

```
20    event = random.choice(events)
21    print(event[0])
```

• Ponovno pokrenite svoj kod. To je puno bolje, ali igrač ne može odgovoriti na pitanje.

• Umjesto upotrebe izjave o ispisu, koristimo ulaznu izjavu. To govori pythonu da očekuje od korisnika da nešto napiše.

```
20    event = random.choice(events)
21    input(event[0])
22    day += 1
```

Sada pohranimo taj unos u varijablu, jer ćemo ga kasnije htjeti upotrijebiti.

```
20    event = random.choice(events)
21    response = input(event[0])
22    day += 1
```

• Ponovno pokrenite svoj kod. Vaša bi igra sada trebala čekati da igrač reagira svaki dan.

Korak 4: Korištenje odgovora igrača

Igrač odgovara na pitanje, sad moramo upotrijebiti njihov odgovor da bismo prosuđivali njihovu potrošnju vode.

• Korisnik bi trebao odgovoriti yes ili no na pitanje. Ako odgovore sa "yes", trebali bismo dodati prvi broj događaja:

```
21    response = input(event[0])
22    if response == "yes":
23        water += event[1]
24    day += 1
```

Iako dodajemo, mnogi su događaji negativni. Baš kao u matematici, dodavanje negativnog broja isto je kao i oduzimanje pozitivnog broja.

• Ako igrač odgovori s "no", u tom slučaju trebamo dodati drugi broj.

```
22  if response == "yes":
23      water += event[1]
24  elif response == "no":
25      water += event[2]
26  day += 1
```

• Pokrenite svoj kod. Igrači bi sada trebali moći odgovarati na pitanja. Primijetit ćete da će se količina vode koju igrač mijenja prema njihovim odlukama. Što se događa ako igrač odgovori nešto drugo osim yes ili no?

Korak 5: Provjerite korisnički unos

Sada imate igru koja radi, ali igrač može odgovoriti na bilo koji način koji želi. Mi samo želimo prihvatiti odgovor "yes" i "no".

• Postoje samo dvije mogućnosti, yes i no, tako da možemo korisnike zatražiti da odgovore na način na koji želimo da odgovore. Izmijenite liniju odgovora na sljedeće:

```
20  event = random.choice(events)
21  response = input(event[0] + " (yes/no): ")
22  if response == "yes":
```

• To je bolje, ali igrač i dalje može odgovoriti na bilo koji način. Igrač može upisati nešto drugo kako bi prevario igru ili pogrešno napisao njihov odgovor. Dodajmo izjavu "else" da bismo pronašli bilo koji odgovor osim yes ili no.

```
24  elif response == "no":
25      water += event[2]
26  else:
27      print("Please answer yes or no!")
28  day += 1
```

• Pokrenite svoj kod. Što se događa ako igrač unese nešto drugo osim yes ili no?

• Sada tražimo od korisnika da ispravi svoj odgovor, ali tada im postavljamo sljedeće pitanje! Moramo im stalno postavljati isto pitanje dok nam ne daju odgovor koji želimo čuti. To možemo učiniti petljom.

Primijetite da smo razrezali liniju odgovora i izjavu if / else. To znači da je sve u petlji. To možete učiniti

```
20  event = random.choice(events)
21  while True:
22      response = input(event[0] + " (yes/no): ")
23      if response == "yes":
24          water += event[1]
25      elif response == "no":
26          water += event[2]
27      else:
28          print("Please answer yes or no!")
29  day += 1
```

tako da označite linije koje želite uvući i pritisnite "tab" na tipkovnici.

• Pokrenite svoj kod. Što se događa ako odgovorite na nešto drugo osim Yes ili No? Trebalo bi vam ponovno postaviti pitanje. Ali što se događa ako odgovorite sa yes ili no? To vam još uvijek postavlja pitanje! Oh moj!

• Petlja "while True" je petlja koja će se stalno petljati. One su vrlo korisne ako želite otići forever, ali mi ne želimo forever, samo dok korisnik ne odgovori yes ili no! Srećom, imamo posebnu naredbu koja nas može izbaciti iz petlje, čak i ako je ona drugačija koja bi išla zauvijek. Ta naredba je "break". Dodajmo naredbu break kada igrač odgovori yes ili no.

```
22      response = input(event[0] + " (yes/no): ")
23      if response == "yes":
24          water += event[1]
25          break
26      elif response == "no":
27          water += event[2]
28          break
29      else:
```

• Ponovno pokrenite svoj kod. Sada bi vam se opet trebalo postavljati isto pitanje ako ne odgovorite sa yes ili no.

Sada stvari dobro funkcioniraju, ali postoji još jedno malo pitanje, protiv kojeg će se neki igrači možda suočiti. Što se događa ako, umjesto da odgovorite sa „yes“, igrač odgovori „Yes“ s velikim slovom Y?

• Python razlikuje velika i mala slova. To znači da ako usporedite dvije žice, pomislit će da su različite žice samo zato što imaju različitu početnu upotrebu. Srećom, to je lako popraviti. Python ima naredbu koju možete koristiti na stringovima(tekst) kako biste sva slova pretvorili u mala. Promijenite liniju odgovora na sljedeće:

```
21  while True:
22      response = input(event[0] + " (yes/no): ").lower()
23      if response == "yes":
```

• Pokrenite svoj kod. Sada bi trebalo prihvaćati “Yes”, “YES”, “yes” i bilo koje druge kombinacije velikih slova.

Čestitamo!

Završili ste ovaj projekt. Dobro napravljeno! Pokušajte igrati svoju igru i vidite možete li preživjeti duže od svojih prijatelja. Želite više? Pokušajte s dolje navedenim izazovima ili pogledajte naše Scratch projekte ili projekte Moonhack iz prethodnih godina.

Izazov: Dodajte još događaja

Postoji nekoliko različitih događaja, ali možete li ih dodati više?

Savjet: ako imate poteškoća, pokušajte kopirati i urediti jedan od događaja koji je već tu.

Izazov: Igra sa zemljom

Možete li modificirati da se igra odnosi na Zemlju, umjesto na Mars? Razmislite o vrstama događaja koji se mogu dogoditi i izborima s kojima se igrač može suočiti.

Napredni izazov: Igrajte ponovo, Sam

Možete li dati igraču opciju da nastavi igrati nakon što su izgubili igru?

Savjet: Želite da cijela igra bude u petlji.