◆ PGZ - Hangman

Skrevet av: Ole Kristian Pedersen, Kodeklubben Trondheim

Oversatt av: Stein Olav Romslo

Kurs: Python
Tema: Tekstbasert, Spill
Fag: Programmering
Klassetrinn: 5.-7. klasse, 8.-10. klasse

Introduksjon

I denne oppgåva skal me lage vårt eige Hangman-spel. Me har laga litt kode allereie for å hjelpe deg på veg. Den kan du laste ned her (./hangman.py). Lagre den der du brukar å lagre Python-koden din.

Det kan sjå ut som mykje kode, men det er berre den øvste delen du skal endre på. Eit lite stykke ned er det ei overskrift som ser slik ut:

```
################################################################################
########
# Library code
################################################################################
########
```

All koden under denne overskrifta kan du berre sjå bort frå. Dette er koden som hjelper deg med å teikne figurar og hente input frå brukaren.

I koden over overskrifta har me laga nokre variablar:

- Variabelen `TRIES` seier kor mange forsøk brukaren har. Sidan det berre er teikna figurar for 7 feil er det lurt om du let denne vere slik den er.
- Lista `WORDS` inneheldt ord som kan gjettast, men ingen av desse orda kan innehalde æ, ø eller å.
- Tilstanden til spelet er gitt i ordboka `state`.

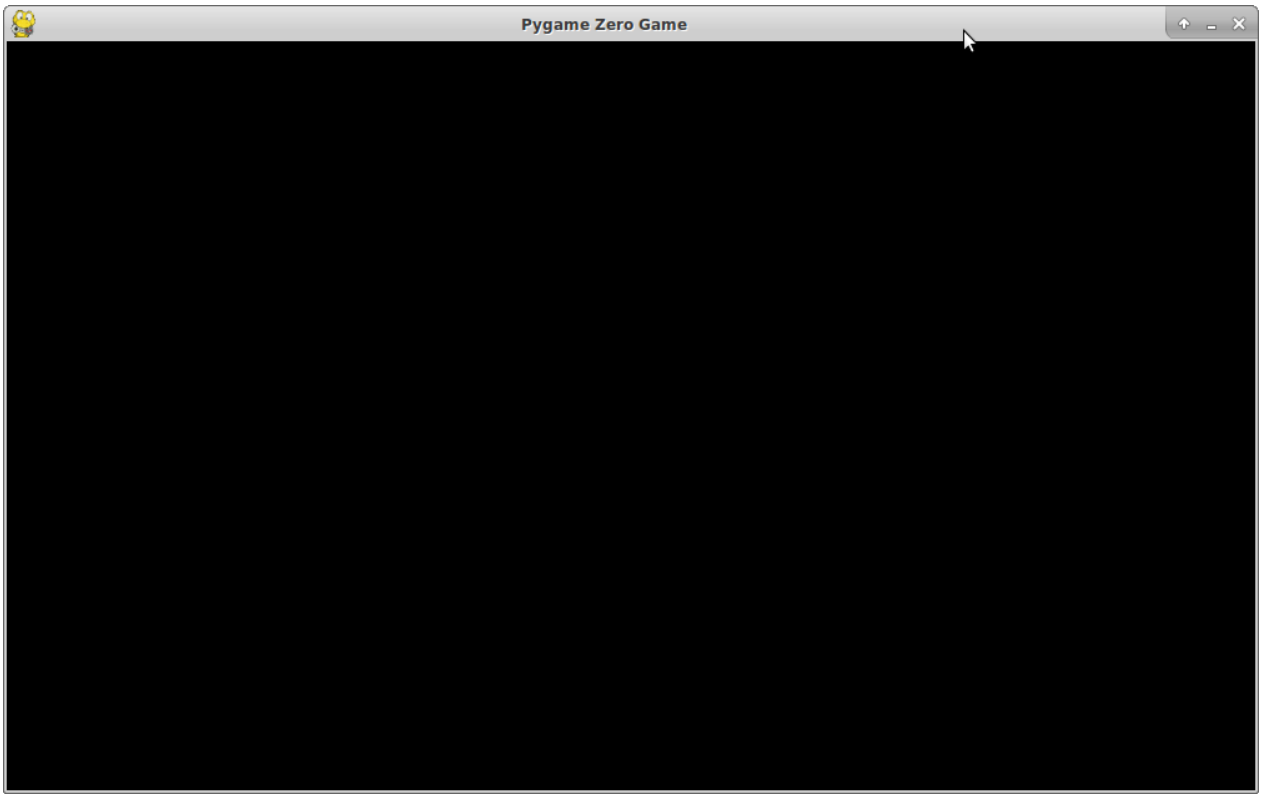
Me har begynt å lage fire funksjonar for deg. Desse skal du gi innhald gjennom steg 1 til 4.

Test at alt fungerer

Ved å køyre kommandoen

```
pgzrun hangman.py
```

skal du få opp eit vindauge som ser slik ut:



Feilsøking

Viss du ikkje får opp eit vindauge som beskrive over, pass på at du har installert Python på riktig måte, lasta ned hangman.py (./hangman.py) og er i same mappe som fila.

Viss du framleis står fast bør du prate med ein rettleiar.

Steg 0: Installere Pygame Zero

For å gjere denne oppgåva må du installere Pygame Zero (https://pygame-zero.readthedocs.io/en/latest/installation.html). Start med å sjekke at du har installert Python 3, altså at Python-versjonen din er nummerert på forma 3.X.X.

Åpne kommandolinja (engelsk: command prompt) på datamaskina di. Brukar du Windows kan du åpne start-menyen og skrive cmd (eventuelt *Ledetekst*, som er det norske namnet på programmet som skal køyre). På Mac og Linux åpnar du terminalvindauget. Skriv inn følgjande:

Windows og Mac:

```
pip install pgzero
```

Linux:

```
sudo pip install pgzero
```

Nokre Linux-system kallar den `pip3` , i så fall må du skrive det i staden for `pip` i koden over. Viss pip ikkje er installert kan du prøve å skrive

```
sudo python3 -m ensurepip
```

før du prøver `sudo pip install pgzero` att.

Steg 1: Lag det hemmelege ordet

No skal me skrive koden som skal køyrast når `create_display_string()` blir kalla. Denne funksjonen har to parametrar - `secret_word` (det hemmelege ordet) og `remaining_letters` (ei liste med dei bokstavane brukaren ikkje har gjetta).

Her er eit døme på korleis funksjonen kan fungere:

```
>>> remaining_letters =['b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
'k', 'l',
                        'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',
'v', 'w',
                        'x', 'y', 'z']
# alle bokstaver fra b til z
>>> secret_word = 'banan'
>>> create_display_string(secret_word, remaining_letters)
'_ a _ a _'
>>> remaining_letters.remove('b')
>>> create_display_string(secret_word, remaining_letters)
'b a _ a _'
>>> create_display_string(secret_word, [])
'b a n a n'
```

Her er alle bokstavar i `secret_word` som òg er i `remaining_letters` erstatta med `_` . Me har dessutan lagt inn eit mellomrom mellom kvart teikn.

For å lage denne funksjonen treng me nokre av tekstfunksjonane me lærte i Tekst ABC (../tekst_abc/tekst_abc_nn.html). Viss du ikkje hugsar desse kan det vere lurt å gå attende og lese gjennom alle dei gule boksane i oppgåva.

No skal du skrive kode i funksjonen. Dette må du gjere:

- [ ] For å konvertere `secret_word` til ei liste må du bruke funksjonen `list()`. Du kan gi ein streng som argument til `list()`, og så blir det returnert ei liste med bokstavar.

  ```
  >>> list("ord")
  ['o', 'r', 'd']
  ```

- [ ] Slå saman bokstavane til ein streng att, med eitt mellomrom mellom kvar bokstav.

  Hint: `s.join(lst)`

- [ ] Byt ut kvar bokstav som er i det nye ordet vårt og i `remaining_letters` med `_`, slik som i dømet over.

  Hint: `s.replace()`

- [ ] Hugs å returnere det hemmelege ordet.

Steg 2: Å starte spelet

No skal me kode funksjonen `start_game()`. Denne funksjonen lagar nøkkel/verdi-para i `state`. Viss du ikkje hugsar korleis ordbøker fungerer kan det vere lurt å repetere oppgåva om ordbøker (../ordboeker/ordboeker_nn.html).

Nøklane i `state` er som følgjer:

- `"running"` fortel hjelpefunksjonane om spelet køyrer
- `"used_tries"` er kor mange forsøk brukaren har brukt
- `"secret_word"` er det ordet brukaren skal gjette
- `"pressed_button"` er den siste knappen som vart trykka av brukaren

- `"help_text"` er hjelpetekst for brukaren som forklarar kva brukaren skal gjere
- `"remaining_letters"` er ei liste over dei bokstavane brukaren ikkje har gjetta
- `"display_string"` er det ordet som visast til brukaren (etter at me har sett inn `_`).

Dette må du gjere:

- [ ] Lag følgjande nøkkel/verdi-par:
  - Nøkkel: `"running"` Verdi: `True`
  - Nøkkel: `"used_tries"` Verdi: `0`
  - Nøkkel: `"pressed_button"` Verdi: `""` (Ein tom streng)
  - Nøkkel: `"help_text"` Verdi: `"Guess a letter!"`

- [ ] For å velje eit tilfeldig ord skal me bruke `random.choice()`. Ved å gi denne funksjonen ei liste som argument returnerast eit tilfeldig element i lista.
  - [ ] Gi `WORDS` som argument og bruk det returnerte ordet som verdien som høyrer til nøkkelen `"secret_word"`.

- [ ] No skal me lage lista over bokstavar som brukaren ikkje har gjetta. For å gjere det brukar me dei 26 fyrste bokstavane i `string.ascii_letters`, og legger til desse i lista.
  - [ ] La den tilhøyrande verdien til `"remaining_letters"` vere ei tom liste.
  - [ ] Bruk `enumerate()` og gå gjennom dei 26 fyrste bokstavane i `string.ascii_letters`, og legg til desse i `state["remaining_letters"]`.

Hint: Bruk `lst.append(elm)`

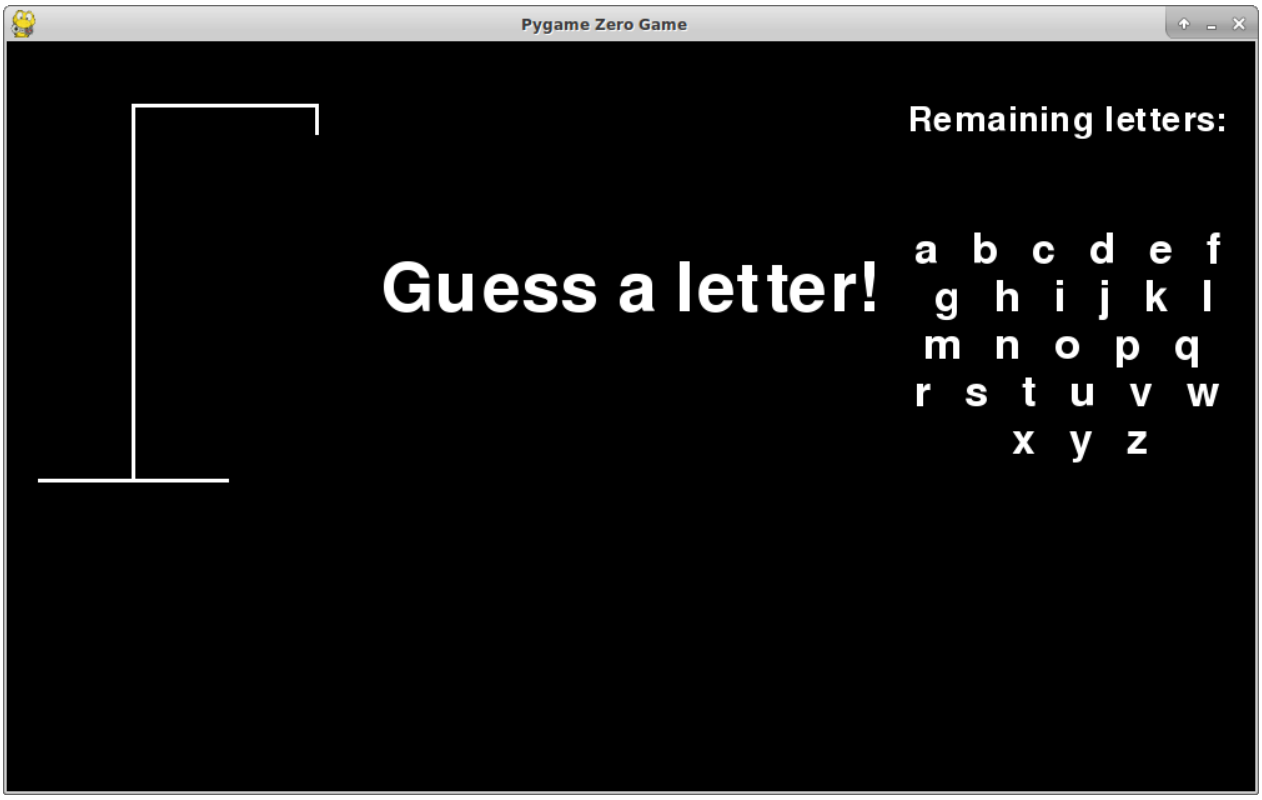
- [ ] For å lage verdien som høyrer til `"display_string"` må me bruke funksjonen me laga i steg 1. Kva to argument skal me gi til funksjonen?

Test spelet ditt

Lagre endringane dine i `hangman.py`, og køyr spelet ditt:

```
pgzrun hangman.py
```

No skal du få opp eit vindauge som ser slik ut:



Steg 3: Å avslutte spelet

I funksjonen `game_over()` skal me endre `state` slik at både brukar og hjelpefunksjonar får beskjed om at spelet er over.

Dette må du gjere:

- [ ] Endre `state["running"]` til `False`.

- [ ] Endre verdien til `state["display_string"]` slik at brukaren kan sjå løysinga, sjølv om brukaren ikkje vann. Dette kan du gjere ved hjelp av funksjonen frå steg 1.

  Hint: Kva skal listen me gir som argument innehalde når me ynskjer å vise alle bokstavane?

Steg 4: main()

I dette steget skal me skrive hovudfunksjonaliteten til spelet. Me skal skrive koden i `main()`, som køyrer kvar gong spelet blir oppdatert. Her skal me finne ut kva bokstav brukaren trykka på, og så sjekke om bokstaven er inneheldt i ordet vårt. Me må oppdatere talet på brukte forsøk, og til slutt sjekke om brukaren har brukt opp alle forsøka sine eller gjetta riktig ord.

Dette må du gjere:

- [ ] Du finn bokstaven brukaren trykka på i `state["pressed_button"]`. Det kan vere lurt å lagre denne i ein eigen variabel så du slepp å skrive `state["pressed_button"]` mange gonger.

- [ ] Viss bokstaven finst i `state["remaining_letters"]` må programmet gjere det følgjande:

  - [ ] Fjern bokstaven frå lista

  - [ ] Viss bokstaven finst i det hemmelege ordet må du oppdatere `state["display_string"]`. Viss ikkje har brukaren brukt opp eitt forsøk, og du må legge til 1 til `state["used_tries"]`.

- [ ] Sjekk om brukaren har brukt opp alle forsøka sine. Viss det er tilfelle må du endre på `state["help_text"]` til `"You lost!"`. Til slutt må du kalle funksjonen `game_over()`.

  Hint: Sjekk om `state["used_tries"]` er større enn eller lik `TRIES`.

- [ ] Me må sjekke om brukaren har gjetta ordet. Ein måte det kan gjerast på er å sjekke kor mange `"_"` det er i `state["display_string"]`. Bruk `s.count("_")` for å telje antal understrekar. Dersom det ikkje er fleire understrekar har brukaren vunne spelet, og du må endre `state["help_text"]` til `"You won!"` og kalle `game_over()`.

Test spelet ditt

Lagre endringane dine i `hangman.py`, og køyr spelet ditt:

```
pgzrun hangman.py
```

Spelet skal no fungere fullt og heilt. Det er nokre ting me må teste:

- [ ] Slik kan det sjå ut når spelaren vinn.

- [ ] Når spelaren tapar kan det sjå ut som i biletet under. Pass på følgjande:
  - [ ] Det skal ikkje vere mogleg å taste inn fleire bokstavar. Viss det går an har du gløymt å setje `state["running"] = False` i `game_over()`.

- [ ] Pass på at du viser løysingsordet nedst.