**Uzdevuma mērķis**:

Izveidot koordināšu minēšanas spēli, kurā lietotājs izvēlas grūtības līmeni un balstoties uz to tiek ģenerēts spēles laukums, kurā nejauši tiek izvietoti punkti, kuru koordinātas ir jāuzmin. Spēles mērķis ir uzminēt visu punktu koordinātas pēc iespējas mazāk minējumos.

**moduli.py:**

Šajā datnē ir visas programmā izmantotās funkcijas, izņemot main()

**main.py:**

Galvenā datne. Lai lietotu programmu, to vajag palaist no šī faila.

**rezultati.txt:**

Šeit tiek uzglabāti visu spēlētāju rezultāti.

**Koda skaidrojums:**

1.-4. rinda: tiek importētas koda bibliotēkas time, os un sys, kā arī modulis.

6.-8. rinda: tiek definēti mainīgie, kuri tiks izmantoti vēlāk, tiek definēts sets ar spēles noteikumiem.

10.-81. rinda: tiek definēta main() funkcija.

**main():**

11. rinda: izvada spēles noteikumus noņemot iekavas no set

12.-81. rinda: while cikls, kurš nodrošina iespēju atkārtot spēli bezgalīgu daudzumu reižu

13.-14. rinda: definē globālos mainīgos, kā arī atsāk punktu un minējumu skaitīšanu no 0

15.-32. rinda: Spēlētājam tiek dota izvēle sākt jaunu spēli vai pārstāt programmas darbību, ievadot 0 vai 1. Tas ir realizēts izmantojot while ciklu, kurš bezgalīgi pieprasa ievadīt vērtību, līdz tam, kad tā ir derīga. Darbības beigšana tiek nodrošināta izmantojot sys.exit() metodi, jo lai gan break metode radīja darbības beigšanas ilūziju, tā īstenībā vienkārši radīja vairāk problēmas.

33. rinda: tiek saukta funkcija laukuma\_izvele() no moduļa. Tā izmanto while ciklu līdzīgi kā iepriekš minētajā spēles sākšanas izvēlē. Pēc grūtības līmeņa izvēles tiek aprēķinātas spēles laukuma izmēri pēc formulas x = grūtības līmenis + 2 (tāpat aprēķina arī y vērtību).

34.rinda: tiek saukta funkcija laukuma\_izveide(), kurā izmantojot list comprehension metodi tiek izveidots divdimensionāls saraksts, kurš sastāv no “tukšiem” lauciņiem, kuri apzīmēti ar o. Arī tiek izveidota šī paša laukuma kopija, kura ir redzama spēles laikā un tiek atjaunota atkarībā no spēlētāja veiktajiem pareizajiem minējumiem.

35.rinda: tiek saukta funkcija merki(), kurā tiek izvēlēti punkti laukumā izmantojot random metodi. Tiek nodrošināts tas, ka 2 punkti netiks nolikti vienādās koordinātās.

36.-48. rinda: while cikls nodrošina spēles procesu. Viens cikla “aplis” izskatās šādi: tiek izvadīts laukums, izmantojot laukuma\_izvade(), tad tiek dota iespēja minēt x un y koordinātas, izmantojot minesana(), kurā pie katras ievadītās vērtības notiek pārbaude, vai vērtība ir derīga tāpat kā iepriekšējos datu ievades gadījumos. Visu laiku tiek uzskaitīti “derīgo” minējumu pāri. Tiek saukta funkcija parbaude(), kura pārbauda vai minējuma koordinātās ir punkts. Gadījumā, ja koordinātas ir pareizi uzminētas, tiek pieskaitīts 1 punkts pie spēlētāja punktu skaita, un tiek atjaunots gan redzamais laukums, gan arī laukums, kuru pārbauda kods. Atsevišķa laukumu atjaunošana ir vajadzīga, lai spēlētāji nevarētu iegūt teorētiski bezgalīgu punktu skaitu vienkārši ievadot identiskas koordinātas n reizes. Cikls beidzas tad, kad spēlētājs uzmin visu punktu koordinātas.

49.-55. rinda: Tiek pieprasīts ievadīt vārdu, kurš tālāk kopā ar grūtības līmeni, punktu skaitu un minējumu skaitu tiek apkopots dictionary struktūrā.

56. rinda: Iepriekš minētā dictionary struktūra tiek izvadīta.

57.-60. rinda: Rezultātu apkopojums tiek pārveidots string un tiek saglabāts iepriekš minētajā failā rezultati.txt

61.-81. rinda: Lietotājam tiek dota izvēle izvadīt visu spēlētāju rezultātus no faila rezultati.txt. Tāpat kā iepriekš, atbildē ievadītie dati tiek pārbaudīti un tiek noteikts, vai tie ir “derīgi”. Gadījumā, ja spēlētājs izvēlas izvadīt rezultātus, tie tiek nolasīti no faila un izvadīti. Pirms rezultātu izvadīšanas tiek pārbaudīts, vai failā ir dati, izmantojot os.path.getsize() metodi no os bibliotēkas. Tā kā visa darbība ir while ciklā, pēc 81. rindas, cikls atsāk darbību no spēles sākšanas izvēles.

83. rinda: tiek saukta main\_darbiba(), kas nodrosina programmas darbību.

**Ekstra piezīmes par kodu:**

Visas rindas, kuras nebija minētas koda darbības skaidrojumā ir tukšas strukturēšanas un pārskates iemeslu pēc.

Time bibliotēka tika izmantota, lai starp teksta izvadēm varētu ielikt nelielu pauzi.

**Palīgi:**

Bruno palīdzēja saprast time bibliotēku, kad veidoju šīs programmas agro versiju. Bruno arī ieteica izmantot sys.exit() metodi, lai pilnībā apstādinātu programmas darbību.