Gymnázium, Praha 6, Arabská 14
Arabská 14, Praha 6, 160 00

Casino

ROČNIKOVÝ PROJEKT

Předmět: Programování
Téma: Casino
Autor: Mariia Gavrylenko

Třída: 2. E
Školní rok: 2022/2023
Vyučující: Mgr. Jan Lána
Třídní učitel: Mgr. Blanka Hniličková

Čestné prohlášení:

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené.

Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze ……..... .....……..

Mariia Gavrylenko

Ráda bych poděkovala panu profesoru Mgr. Janu Lánovi, který byl vedoucím ročníkového projektu. Dále bych chtěla poděkovat svému otci, který mi vždycky byl rád poradit s kódem a kamarádům, které podporovali mě v průběhu mé ročníkové práce.

Anotace

Tato ročníková práce je na téma, které je velmi popularní a zajímavé v dnešní době. Je to casino. Opravdový casinový program v pythonu, který simulujet hru flip coin, ruletu a slot machine. Aplikace začíná přihlašovací obrazovkou, kde se uživatel může zaregistrovat nebo použít své přihlašovací údaje k přihlášení do platformy. V poslední sekci uživatel může změnit informace o svém účtu a kouknout se na svoji statistiku. Při psání jsem používala multiplatformní sadu modulů jazyka Python – Pygame.

Annotation

This year assignment is on a topic that is very popular and interesting nowadays. It is a casino. A real casino program in python that simulates the game of flip coin, roulette, and slot machine. The application starts with a login screen where the user can register or use their login credentials to log into the platform. In the last section, the user can change their account information and look at their statistics. I used a multiplatform set of Python modules - Pygame.

Zadání projektu

Téma: Casinový program který simulujet hru flip coin, ruletu a slot machine.

Upřesnění zadání:

- Hesla při ukladání se musejí šifrovat.
- Když se uživatel přihlašuje nebo registruje, heslo není vidět, je označeno hvězdičkami.
- Ani heslo ani jméno nesmí byt víc, než 12 znaků.
- Při registraci musí uživatel napsat heslo dvakrát.
- Ukladání hráčů do databaze
- Nakreslit animaci rulety, slot machine a flip coinu.
- Vysvětlení her a sázek
- U rulety je možnost vsadit víc než 1 sázku.

Obsah

Úvod

Pro svoji ročníkovou práci jsem si vybrala téma Casino. Můj program spočívá v tom, že uživatel bude moct si zahrát opravdové casino. V programu si můžete zahrát ty nejpopularnější hry. V poslední sekci si může provádět operace vkladu.

Cílem mého ročníkového projektu bylo udělat nejvíc podobnou reálnému casinu hru, která bude simulovat ty nejvíc oblíbené hry. Tato dokumentace vám poskytne přehled všech funkcí, které můj casino program nabízí, a pomůže vám s jeho používáním. Moje casino hry jsou navrženy tak, aby byly snadno ovladatelné a zábavné pro hráče všech úrovní. V dokumentaci naleznete informace o registraci do casina, o vkladech a výběrech peněz. Dále se dozvíte, jak používat hry.

Můj projekt řeší problematiku, kterou je riziko vzniku závislosti na hazardních hrách, které může mít negativní dopad na psychické i finanční zdraví jednotlivců a jejich rodin. Můj program ale měl by uživatelům přinést spoustu pozitivních emocí, protože peníze si může vložit vždycky a hlavně kolik bude chtít!

Pro napsání jsem zvolila multiplatformní sadu modulů jazyka Python – Pygame.

1. Použité technologie

Pygame je knihovna pro programování her a multimediálních aplikací v jazyce Python. Tato knihovna poskytuje nástroje pro práci s grafikou, zvukem a vstupem z klávesnice a myši. Je navržena tak, aby byla snadno použitelná pro začátečníky v programování her, ale zároveň poskytuje dostatek možností pro pokročilé programátory.

Pygame je open-source software, což znamená, že je k dispozici zdarma. Knihovna obsahuje mnoho příkladů a tutoriálů, které pomáhají novým uživatelům v rychlém zvládnutí základních funkcí.

Mezi výhody Pygame patří také multiplatformnost, díky čemuž může být použita na různých operačních systémech včetně Windows, Linux a Mac OS X. Knihovna je také snadno rozšiřitelná pomocí různých modulů a pluginů, což umožňuje programátorům vytvářet pokročilé hry a aplikace.

Celkově lze říci, že Pygame je skvělou volbou pro programování her a multimediálních aplikací v jazyce Python. Poskytuje snadné použití pro začátečníky, ale zároveň dostatek funkcí pro pokročilé programátory.

Pro instalaci knihovny Pygame do Visual Studio Code, jak jsem to udělala já, si musite v terminálu napsat příkaz „ pip install pygame “. Když ne, tak pro stažení a instalaci knihovny musite jít na webovou stránku Pygame[1]. Vybrat verzi Pygame pro váš operační systém Python verzi, kterou používáte a pak si to stáhnout.

2. Problematika a technologie programu

[1] http://www.pygame.org/download.shtml.

2.1. Database

V teto podkapitole bych si venovala Database a funkcim, ktere jsem pouzivala v procesu programovani.

Zacala bych funkci *add_user()*. Funkce má tři parametry: *name* (jméno uživatele), *password* (heslo) a *sec_passwor*d (potvrzení hesla). Umožňuje registraci hráče. Funkce nejprve ověřuje, zda se heslo a jeho potvrzení shodují a zda je heslo dostatečně dlouhé (alespoň 5 znaků a maximálně 12). Pokud tyto podmínky nejsou splněny, funkce vrátí False a vypíše odpovídající chybové hlášení pomocí funkce *showError().* Poté se funkce pokusí otevřít soubor users.json a načte jeho obsah do proměnné data. Pokud soubor neexistuje, vytvoří prázdný seznam a uloží ho do souboru. Následně funkce projde seznam registrovaných uživatelů a zkontroluje, zda již existuje uživatel se zadaným jménem. Pokud ano, funkce vrátí False a vypíše chybové hlášení. Pokud se nové jméno uživatele ještě nevyskytuje, funkce vytvoří nový uživatelský objekt, který obsahuje informace o uživateli, jako je id, jméno, zašifrované heslo pomocí *"hashlib"*. Je to knihovna jazyka Python, která poskytuje funkce pro generování bezpečných hashovacích funkcí. Jednou z hash funkcí, které lze pomocí *hashlib* generovat, je SHA-256, což je široce používaná kryptografická *hash* funkce. Výsledná hodnota hash bude 64znakový hexadecimální řetězec, který představuje hash SHA-256 vstupních dat. Hašovací funkce je navržena tak, že i malá změna vstupních dat vede ke zcela jiné hodnotě haše, což je užitečné pro ověření integrity dat a zajištění, že s nimi nebylo manipulováno. Poté přidá nový objekt do seznamu data a zapíše celý seznam do souboru *users.json.*

Dálé bych pokračovala funkci *authorize().* Tato funkce slouží k ověření identity uživatele, který se pokouší přihlásit do aplikace. Funkce má dva parametry: *name* (jméno uživatele) a *password* (heslo). Funkce nejprve pokusí otevřít soubor users.json a načte jeho obsah do proměnné data. Pokud soubor neexistuje, funkce vypíše chybové hlášení a vrátí False pomocí funkce *showError()*. Následně funkce projde seznam registrovaných uživatelů a zkontroluje, zda existuje uživatel se zadaným jménem a heslem. Pokud ano, nastaví proměnnou *is_in_base* na True. Poté funkce ověří, zda proměnná *is_in_base* je True, což znamená, že uživatel s daným jménem a heslem je v seznamu registrovaných uživatelů. Pokud je to pravda, funkce vrátí True, což značí, že uživatel byl úspěšně ověřen. Pokud je proměnná *is_in_base* False, funkce vypíše chybové hlášení, že uživatelské jméno nebo heslo jsou nesprávné, a vrátí False, což znamená, že ověření bylo neúspěšné.

```python
def add_user(name,password,sec_password):
    if password != sec_password:
        showError("Passwords do not match")
        return False
    if (len(password) < 5):
        showError("Your password must be between 5 and 12 characters")
        return False
    try:
        with open('users.json') as json_file:
            data = json.load(json_file)
    except:
        json_object = json.dumps([], indent=4)
        with open("users.json", "w") as outfile:
            outfile.write(json_object)
        data = []
    for user in data:
        if(user['name'] == name):
            showError("This name is taken")
            return False
    user ={
    "id": len(data)+1,
    "name": name,
    "password": sha256(password.encode('utf-8')).hexdigest(),
    "money": 1000,
    "roulete_wins": 0,
    "slot_wins": 0,
    "coin_wins": 0
    }
    data.append(user)
    json_object = json.dumps(data, indent=4)
    with open("users.json", "w") as outfile:
        outfile.write(json_object)
    return True
```

Screenshot 1 – funkce add_user().

```python
def authorize(name,password):
    try:
        with open('users.json') as json_file:
            data = json.load(json_file)
    except:
        showError("You need to sign up")
        return False
    is_in_base = False

    for user in data:
        if(user['name'] == name and user['password'] == sha256(password.encode('utf-8')).hexdigest()):
            is_in_base = True

    if is_in_base:
        return True
    else:
        showError("Your username or password is wrong")
        return False
```

Screenshot 2 – funkce authorize().

Teď bych chtěla řict něco o funkci *showError()*. Tato funkce slouží k zobrazení chybového hlášení uživateli v grafickém uživatelském rozhraní. Funkce má jeden parametr text, který obsahuje text chybového hlášení, které má být zobrazeno uživateli. Funkce vytvoří nové okno s názvem "Error" pomocí modulu *Tkinter*. Do tohoto okna funkce umístí Label, což je widget pro zobrazení textu. Label bude obsahovat text chybového hlášení a bude mít nastavenou velikost písma na 30 bodů. Také bude mít nastavenou barvu pozadí na tmavě šedou a barvu textu na světle béžovou. Nakonec funkce spustí hlavní smyčku okna pomocí metody *mainloop()*, aby uživatel mohl vidět chybové hlášení. Po zavření okna se funkce ukončí.

```
6   def showError(text):
7       window = Tk()
8       window.title("Error")
9
10      lbl = Label(window, text=text, font=("Arial Bold", 30),bg = '#222222', fg="#F3EFE0")
11      window.configure(bg='#222222')
12      lbl.grid(column=0, row=0)
13
14      window.mainloop()
```

Screenshot 3 – funkce showError().

Teď se vrhnem na funkci *load_data()*. Tato funkce slouží k načtení dat uživatele z databáze uložené v souboru users.json. Funkce má jeden parametr name, který obsahuje jméno uživatele, pro kterého se mají data načíst. Funkce otevře soubor users.json a načte jeho obsah pomocí modulu json. Následně funkce projde seznam uživatelů v proměnné data a hledá uživatele se zadaným jménem. Pokud takový uživatel existuje, funkce vrátí seznam obsahující některé informace o uživateli (jméno, peníze, počet výher v různých hrách). Pokud uživatel s daným jménem neexistuje, funkce vrátí prázdný seznam. Tato funkce může být použita v různých částech aplikace, kde se potřebují načíst data o uživateli, například v menu uživatele nebo v samotné hře.

```
78  def load_data(name):
79      with open('users.json') as json_file:
80          data = json.load(json_file)
81      for user in data:
82          if(user['name'] == name):
83              return [user['name'], user['money'], user["roulete_wins"],  user["slot_wins"],user["coin_wins"]]
84      return []
```

Screenshot 4 – funkce load_data().

Poslední funkce, které bych si věnovala je *update()*. Tato funkce slouží k aktualizaci dat o uživateli v databázi uložené v souboru users.json. Funkce má pět parametrů: *name* - jméno uživatele, *money* - množství peněz uživatele, *roulete_wins* - počet výher uživatele v ruletě,

slot_wins - počet výher uživatele na automatech a *coin_wins* - počet výher uživatele v hře s mincemi. Funkce otevře soubor users.json a načte jeho obsah pomocí modulu json. Následně funkce projde seznam uživatelů v proměnné data a hledá uživatele se zadaným jménem. Pokud takový uživatel existuje, funkce aktualizuje jeho informace včetně počtu peněz a počtu výher v jednotlivých hrách. Poté se zapisují aktualizovaná data zpět do souboru users.json. Tato funkce může být použita v různých částech aplikace, kde se potřebují aktualizovat data o uživateli, například po dokončení hry nebo po přidání peněz na účet uživatele.

```
87  def update(name,money,roulete_wins,slot_wins,coin_wins):
88      with open('users.json') as json_file:
89          data = json.load(json_file)
90          for user in data:
91              if(user['name'] == name):
92                  user["money"] = money
93                  user["roulete_wins"] = roulete_wins
94                  user["coin_wins"] = coin_wins
95                  user["slot_wins"] = slot_wins
96                  break
97
98      with open("users.json", "w") as outfile:
99          outfile.write(json.dumps(data, indent=4))
```

Screenshot 5 – funkce update().

2.2. Main nebo-li „rocnikovka.py“

Jak už jsem si psala před tím, tento kód je v jazyce Python, který používá knihovnu Pygame k vytvoření jednoduchého grafického uživatelského rozhraní. Aplikace se skládá z několika her, jako je ruleta, hrací automaty a přehazování mincí, a také z funkcí pro správu účtu, jako je přihlašování, registrace, vkládání a vybírání peněz. Skript inicializuje několik globálních proměnných pro uložení informací, jako je jméno uživatele a aktuální fáze aplikace. Definuje také několik funkcí pro zpracování uživatelských akcí, jako je klikání na tlačítka pro přechod mezi obrazovkami a hrami a vkládání nebo vybírání peněz. Skript také využívá několik externích modulů, například "*Database*" pro ukládání uživatelských dat, "descriptions" pro zobrazování herních instrukcí a "*Classes*" pro definování různých herních objektů, jako jsou tlačítka a textová pole. Celkově se tento kód jeví jako jednoduchá implementace herní aplikace, ale je důležité si uvědomit, že hraní může být návykové a potenciálně škodlivé. Proto je důležité být při používání takových aplikací opatrný a hrát pouze v rámci svých možností.

2.2.1. Scény

Jedná se o blok kódu Pythonu, který definuje několik funkcí, jež vytvářejí scénu pro různé části hry nebo aplikace. Každá funkce nastavuje globální proměnnou "stage" na určitou hodnotu, která slouží k určení, která část aplikace nebo hry je právě aktivní. Například funkce *set_stage_roulette()* nastaví scénu na "*Roulette*" a vytvoří novou instanci třídy "*Roulette*". Funkce *set_stage_home()* nastaví scénu na "*Home*", zatímco funkce *set_stage_slot()* nastaví scénu na "*Slot*" a vytvoří novou instanci třídy "*SlotGame*". Kód obsahuje také funkce pro nastavení scény pro další části aplikace, jako jsou stránky účtu a přihlášení. Klíčové slovo "*global*" je použito k označení toho, že tyto funkce budou modifikovat globální proměnné, nikoliv vytvářet lokální proměnné. V každé z funkcí je také volána funkce *update_btns()*, která aktualizuje tlačítka zobrazená na obrazovce na základě aktuální fáze. Tento kód je jenom součástí většího projektu, který zahrnuje různé fáze nebo způsoby interakce.

```
36  #Funkce, která aktualizuje tlačítka zobrazená na obrazovce na základě aktuální fáze.
37  def update_btns(btns):
38      for btn in btns:
39          btn.stage = stage
40
41  def set_stage_roulette():
42      global stage, roulette
43      stage = "Roulette"
44      update_btns(nav_bar_btns)
45      roulette = Roulette(screen,log_user_name_txt.user_text)
46
47  def set_stage_home():
48      global stage
49      stage = "Home"
50      update_btns(nav_bar_btns)
51
52  def set_stage_slot():
53      global stage, slot
54      stage = "Slot"
55      slot = SlotGame(screen,log_user_name_txt.user_text)
56      update_btns(nav_bar_btns)
57
58  def set_stage_coinflip():
59      global stage, coin_flip
60      stage = "Coinflip"
61      coin_flip = CoinGame(screen,log_user_name_txt.user_text)
62      update_btns(nav_bar_btns)
63
64  def set_stage_account():
65      global stage
66      stage = "Account"
67      update_btns(nav_bar_btns)
```

```
70  def set_stage_about():
71      global stage
72      stage = "About"
73      update_btns(nav_bar_btns)
74
75  def set_stage_deposite():
76      global stage
77      stage = "Deposite"
78
79  def set_signin_page():
80      global stage
81      stage = "signin"
82
83  def set_stage_login():
84      global stage
85      stage = "login"
86      log_user_name_txt.user_text = log_user_name_txt.textholder
87      log_user_password_txt.user_text = log_user_password_txt.textholder
```

Screenshot 6 – funkce set_stage_...().

2.2.2. Login funkce a sign in funkce

Jedná se o dvě funkce související s ověřováním uživatelů a vytvářením účtů. První funkce, *login_function()*, kontroluje, zda se přihlašovací údaje uživatele zadané v přihlašovacím formuláři shodují s hodnotami uloženými v databázi, a to voláním metody *authorize()* třídy s názvem *Database*. Pokud je autorizace úspěšná, funkce načte údaje uživatele a změní proměnnou stage na "*Home*". V opačném případě uživatel není autorizován a nemůže k aplikaci přistupovat.

Druhá funkce, *sign_function()*, přidá nového uživatele do databáze voláním metody *add_user()* třídy *Database*. Pokud je uživatel úspěšně přidán, je proměnná stage nastavena na "*login*", aby byl uživatel přesměrován na přihlašovací stránku. Pokud se uživatele nepodaří do databáze přidat, vypíše se do konzoly chybové hlášení. Celkově tyto funkce zajišťují potřebnou funkčnost pro ověření uživatele a vytvoření.

Screenshot 7 – funkce login a sign in.

```
90  def login_function():
91      if Database.authorize(log_user_name_txt.user_text,log_user_password_txt.user_text):
92          global stage
93          load_data()
94          stage ="Home"
95          update_btns(nav_bar_btns)
96
97  def sign_function():
98      global stage
99      if Database.add_user(signin_user_name_txt.user_text, signin_user_password_txt.user_text,signin_user_sec_password_txt.user_text):
100
101         stage ="login"
102     else:
103         print("false")
```

2.2.3. Vklad a výběr peněz

Funkce *add_money()*. Tato funkce slouží ke zvýšení množství peněz v herním účtu. Nejprve se převádí uživatelův vklad na celé číslo a přičítá se k penězům na herním účtu. Poté se uživateli zobrazí úvodní obrazovka a aktualizují se údaje v databázi s uživatelskými daty. Pokud uživatel zadá neplatný vklad, funkce neudělá nic.

Funkce *wisdraw()*. Tato funkce se zřejmě stará o výběr peněz z účtu uživatele. Kontroluje, zda hodnota zadaná do textového pole "*dep_money_out_txt*" není výchozí hodnotou (což by znamenalo, že uživatel nic nezadal) a zda má uživatel dostatek prostředků k výběru. Pokud jsou tyto podmínky splněny, odečte částku od uživatelovy proměnné "*money*" a aktualizuje databázi. Nakonec nastaví scénu zpět na úvodní stránku.

```
145 def add_money():
146     global money
147     try:
148         money += int(dep_money_txt.user_text)
149         dep_money_txt.user_text = ""
150         set_stage_home()
151     except:
152         None
153 
154     Database.update(log_user_name_txt.user_text,money,roulete_wins,slot_wins,coin_wins)
155 
156 def wisdraw():
157     global money
158     if(dep_money_out_txt.user_text != dep_money_out_txt.textholder and money- int(dep_money_out_txt.user_text)>=0):
159             money -= int(dep_money_out_txt.user_text)
160 
161     dep_money_out_txt.user_text = dep_money_out_txt.textholder
162     set_stage_home()
163     Database.update(log_user_name_txt.user_text,money,roulete_wins,slot_wins,coin_wins)
```

Screenshot 8 – funkce add_money() a wisdraw().

2.2.4. Stránky

Funkce page. Tato funkce slouží k rozhodnutí, jakou stránku zobrazit v závislosti na aktuálním stavu hry. Jedná se o výpis stavových funkcí pro jednotlivé stránky, jako jsou například *loginPage*, *homePage*, *roulettePage*, atd. Pokud se aktuální stav rovná některému ze stavů, které se ošetřují v match, zavolá se odpovídající funkce pro tuto stránku.

```
343  def page(current_state):
344      match current_state:
345          case "login":
346              loginPage()
347          case "Home":
348              homePage()
349          case "signin":
350              signinPage()
351          case "Roulette":
352              roulettePage()
353          case "Slot":
354              slotPage()
355          case "Coinflip":
356              coinflipPage()
357          case "Account":
358              accountPage()
359          case "About":
360              aboutPage()
361          case "Deposite":
362              depositePage()
```

Screenshot 9 – funkce page().

Tyto funkce zřejmě definují jednotlivé stránky aplikace kasinové hry. Funkce *loginPage()* zřejmě zobrazuje přihlašovací formulář s tlačítky a textovými vstupními poli. *homePage()* zobrazuje hlavní stránku aplikace s navigačním panelem a sadou karet. *signinPage()* zobrazí přihlašovací formulář s tlačítky a textovými vstupními poli. *roulettePage()* zobrazí stránku pro hraní rulety s navigačním panelem a herní deskou rulety. *slotPage()* zobrazí stránku pro hraní slotu s navigační lištou a herní plochou pro výherní automaty. *coinflipPage()* zobrazí stránku pro hraní hry házení mincí s navigační lištou a herní deskou pro házení mincí. *accountPage()* zobrazí stránku uživatelského účtu s aktuálním zůstatkem a počtem výher v jednotlivých hrách. *aboutPage()* zobrazí stránku o hře s informacemi o kasinové hře a jejím tvůrci. *depositePage()* zobrazí formulář pro vklad s tlačítky a textovými vstupními poli. Tyto funkce jsou volány hlavní funkcí, která spravuje stav hry a zpracovává uživatelské vstupy. Funkce *nav_bar()* je také odkazována v několika funkcích stránky a zřejmě zobrazuje navigační panel, který umožňuje uživateli pohybovat se mezi různými stránkami.

2.2.5. Events textového pole

Jedná se o funkci *text_field_events()* , která zpracovává události související s textovými poli. Začíná deklarací několika globálních proměnných a přiřazením jejich hodnot na základě aktuální fáze hry. Proměnná *stage* slouží k určení, na které stránce nebo části aplikace se uživatel nachází. Funkce pak prochází ve smyčce každé textové pole na aktuální stránce a kontroluje, zda na něm uživatel provedl nějakou událost, například kliknutí nebo psaní. Pokud uživatel klikne na textové pole, funkce nastaví atributy *active* a *isSelected* na hodnotu *true* a vymaže textové pole, pokud obsahuje výchozí zástupný text. Pokud uživatel píše do textového pole, funkce připojí napsané znaky k atributu *user_text* textového pole, pokud nebylo dosaženo maximální délky. Pokud je textové pole aktivní, funkce nastaví barvu na aktivní barvu, pokud ne, nastaví barvu na pasivní barvu. Nakonec funkce zkontroluje, zda uživatel stiskl klávesu *enter*, a pokud ano, provede funkci přihlášení, pokud je aktuální fáze přihlášení. Celkově se tato funkce jeví jako důležitá součást aplikace, která zahrnuje uživatelský vstup a formulářová pole.

```
365  def text_field_events(event):
366      global stage, roulette, coin_flip, slot
367      if (stage =="login"):
368          page_txt = login_page_txt
369      elif(stage == "signin"):
370          page_txt = signin_page_txt
371      elif(stage == "Roulette"):
372          page_txt = roulette.bet_txt
373          roulette.bet_txt[0].draw()
374      elif(stage == "Deposite"):
375          page_txt = dep_txts
376      elif(stage == "Coinflip"):
377          page_txt = [coin_flip.text_number_bet]
378      elif(stage == "Slot"):
379          page_txt = [slot.text_number_bet]
380      else:
381          page_txt = []
```

Screenshot 10 – funkce events textového pole.

```
383     for text in page_txt:
384
385         if event.type == pygame.MOUSEBUTTONDOWN:
386             if text.textRect.collidepoint(event.pos) and text.static == False :
387                 text.active = True
388                 text.isSelected = True
389                 if(text.user_text ==text.textholder):
390                     text.user_text = ''
391             else:
392                 if(text.user_text == ''):
393                     text.user_text = text.textholder
394                 text.isSelected = False
395                 text.active = False
396
397         if event.type == pygame.KEYDOWN and text.isSelected:
398
399             if event.key == pygame.K_RETURN:
400
401                 if(stage == "login"):
402                     login_function()
403
404             if text.alt_color == "9" or text.alt_color == "36":
405
406                 if event.key == pygame.K_BACKSPACE:
407                     text.user_text = text.user_text[:-1]
408
409                 if(len(text.user_text)<text.max_length):
410
411                     if (event.unicode.isnumeric()):
412                         if(int(text.user_text+event.unicode)!=0):
413                             text.user_text += event.unicode
414                     else:
415                         None
416                     continue
417
418             # Kontrola backspace
419             if event.key == pygame.K_BACKSPACE:
420                 text.user_text = text.user_text[:-1]
421             elif(len(text.user_text)<text.max_length):
422                 text.user_text += event.unicode
423
424         if text.active:
425             text.color = text.color_active
426         else:
427             text.color = text.color_passive
```

Screenshot 11 – funkce pro texty na ruznych strankach, check pro backspace.

2.2.6. Pre_login

V průběhu psaní mé práce mě nakonec přestalo bavit neustále zadávat svoje uživatelské jméno a heslo, a tak jsem se rozhodla vytvořit funkci *pre_login()*, která po zapnutí aplikace vyplní moje udaje a vstoupí do *homePage*.

```
430  def pre_login():
431      global name, roulete_wins, money, slot_wins, coin_wins, stage
432      data = Database.load_data("")
433      name = data[0]
434      money = data[1]
435      roulete_wins = data[2]
436      slot_wins = data[3]
437      coin_wins = data[4]
438      nav_bar_account_btn = Button(1000, 0, 280, 60,f"{name}"+" "+f"{money}"+"$", set_stage_account, False, screen, stage, "Account")
439      nav_bar_btns.append(nav_bar_account_btn)
440      stage = "Home"
441      log_user_name_txt.user_text = ""
442      update_btns(nav_bar_btns)
443  pre_login()
```

Screenshot 12 – funkce pre_login()

2.3. Animace

2.3.1. Ruleta a ruletní míček

Jedná se o metodu s názvem *draw_circle*. Metoda přebírá dva parametry, *x* a *y*, které určují souřadnice levého horního rohu obdélníku, do něhož bude nakreslena elipsa. Elipsa se vykreslí pomocí funkce *pygame.draw.ellipse*, která jako argumenty přijímá obrazovku, na kterou se má kreslit, barvu elipsy, obdélník definující velikost a polohu elipsy a nepovinný parametr šířky, který je ve výchozím nastavení roven nule. Po vykreslení elipsy metoda vypočítá střed elipsy a poloměr, přičemž pro šířku a výšku elipsy použije pevnou hodnotu 400. Definuje také seznam čísel, která představují čísla na ruletě. Metoda pak iteruje přes seznam čísel a pro každé číslo nakreslí oblouk pomocí funkce *pygame.draw.arc*. Oblouky střídají dvě barvy: červenou ("0x9d0208") a černou ("0x000000"), s výjimkou prvního oblouku, který je zelený ("0x4f772d"). Pro každé číslo metoda vypočítá souřadnice textu pro toto číslo pomocí trigonometrie a poloměru elipsy. Poté vykreslí číslo jako text pomocí funkce *Pygame.render* s bílou barvou ("0xF3EFE0") a dá jej na obrazovku na vypočtené pozici. Celkově metoda *draw_circle* vykresluje ruletu s čísly, přičemž k vykreslení elipsy a oblouků používá funkce *Pygame* a k výpočtu polohy čísel trigonometrii.

Funkce *ball_animation()* animuje pohyb kuličky v ruletě. Nejprve definuje souřadnice středu kola, poloměr a vnořenou smyčku funkce, která je zodpovědná za animaci kola. Poté náhodně vybere číslo ze seznamu čísel na kole a uloží je do atributu *self.win_num*. Poté se zavolá funkce slow, která pomalu roztočí kolo, dokud se kulička nezastaví na náhodně vybraném čísle. To provede tak, že nejprve roztočí kolo konstantní rychlostí, dokud kulička nedosáhne sektoru, který obsahuje vybrané číslo. Poté kolo zpomalí, dokud kulička nedosáhne přesného čísla, aby simulovala zpomalení skutečné rulety.

2.3.2. Coin

Tato třída se věnuje reprezentaci mince v herní aplikaci. V konstruktoru třídy jsou inicializovány základní parametry mince, jako jsou její souřadnice na obrazovce, barvy atd. Metoda draw se pak stará o vykreslení mince na obrazovku pomocí funkcí knihovny Pygame. Mince je vykreslena jako kruh s vnitřním i vnějším eliptickým tvarem, přičemž vnitřní kruh má menší průměr a je vybarven zelenou barvou, zatímco vnější elipsa slouží jako pozadí a je vybarvena černou barvou. Funkce animuje otáčení mince ve hře. Funkce obsahuje dvě vnitřní funkce, *loop()* a *end()*, které slouží k definování chování animace mince v různých fázích. Funkce *loop*() řídí počáteční fáze animace mince a postupně zvětšuje velikost černé elipsy, která obklopuje minci, dokud není celá elipsa černá. Poté zmenšuje velikost černé elipsy a současně zvětšuje velikost zelené elipsy, která představuje barvu otáčející se mince. Funkce *end()* definuje závěrečnou fázi animace, kdy je barva mince určena náhodnou proměnnou *self.coin_random*. Pokud je hodnota *self.coin_random* False, funkce změní barvu mince na červenou a zmenší velikost zelené elipsy obklopující minci, dokud nezmizí. Pokud je *self.coin_random* True, barva mince zůstane zelená a velikost zelené elipsy se zmenší. Nakonec animační funkce několikrát zavolá funkci loop, pokaždé se snižující se rychlostí, a poté zavolá funkci end, aby animaci dokončila. Počet iterací smyčkové funkce je při každém volání animace náhodně generován.

2.3.3. Slots

Jedná se o metodu s názvem "*draw*". Metoda zřejmě nejprve iteruje seznam objektů tlačítek a volá jejich metodu "*process*". Poté vykreslí textový objekt související s počtem provedených sázek. Pokud byla provedena sázka, zavolá metodu "*process*" objektu tlačítka *play*. Metoda pak na obrazovku vykreslí několik obdélníků, které představují různé části hry

na výherním automatu. Každý obdélník má přiřazenou určitou barvu, která je určena slovníkem nazvaným "colors". Velikost a pozice každého obdélníku je také určena pomocí metody "*Rect*" z modulu pygame. Nakonec se používá metoda "*blit*" pro vykreslení skutečných čísel, přičemž používá objekt písma nazvaný "*base_font*" a pozici každého čísla na obrazovce. Funkce *animation()* je zodpovědná za animaci slotu. Nejprve nakreslí obdélník v barvě tlačítek. Poté vygeneruje náhodná čísla pro každý slot a nakreslí je na obrazovku. Animace se vytváří generováním nových náhodných čísel a překreslováním slotů po dobu 60 iterací se zpožděním určeným parametrem *speed*. Nakonec funkce animaci zastaví a zobrazí konečná čísla vykreslením slotů se správnými barvami a čísly. První smyčka zobrazí pouze první slot, druhá smyčka zobrazí první dva sloty a třetí smyčka zobrazí všechny tři sloty. Tato funkce používá *pygame* ke kreslení obdélníků, písem a k aktualizaci zobrazení pomocí *pygame.display.flip()*. Používá také metodu *clock.tick(speed)* k řízení rychlosti animace.

Závěr

Díky této práci jsem se naučila spoustu zajímavých věcí o Pythonu a zlepšila své dovednosti v tomto jazyce. Kdo ví, třeba v budoucnu budu v tomto jazyce programovat :).

Zpočátku se mi zdálo nereálné vytvořit krásnou herní animaci. Trvalo mi velmi dlouho, než jsem na code animaci přišla, nakonec jsem to po použití matematiky a několika minutách výpočtů zvládla. Vypočitala jsem všechny úhly, ve kterých se musí míč při kutálení zastavit. Naučila jsem se, jak udělat text přesně uprostřed, jen vlevo a jen vpravo. Správně se říká, že bez znalosti matematiky nelze programovat.

Samozřejmě si myslím, že k mé práci je co doplnit. Například 4. hru, pro kterou jsem si nechaal místo na úvodní stránce, a nějaké obrázky na kartách na, aby to nevypadalo tak nudně. Také například moje ruleta neni úplně stejná jako ta skutečná. Mohu vsadit jen na barvu a číslo, zatímco ta skutečná má spoustu dalších možností, takže bych je tam v budoucnu přidala.

Naprogramovat tuto práci mi zabralo dva týdny tvrdé práce a bezesných nocí, ale v podstatě jsem se svou hrou spokojená. Funguje, animace jsou pěkné, hesla jsou zašifrovaná. Dalo by se říct, že zadání projektu jsem vyplnila. Byla to velmi zajímavá zkušenost, která mi dala spoustu nových poznatků.

Seznam použitých webových zdrojů

1) https://www.geeksforgeeks.org/how-to-create-a-text-input-box-with-pygame/
2) https://www.geeksforgeeks.org/reading-and-writing-json-to-a-file-in-python/
3) https://www.freecodecamp.org/news/create-a-dictionary-in-python-python-dictmethods/
4) https://www.pygame.org/docs/ref/draw.html#pygame.draw.circle
5) https://www.youtube.com/watch?v=WIIf3WaO5x4
6) https://stackoverflow.com
7) https://www.w3schools.com
8) https://www.pygame.org/news

Seznam vyobrazení