Übungsblatt 3

Abgabe als ZIP-Datei **Deadline**: 23.10.2020 23:55 Uhr

Ablauf der Übung

Deadline: : 23.10.2020 23:55 Uhr

- Die finalen Python Dateien sollen am E-Campus in der Übungsabgabe als ZIP Datei hochgeladen werden.
- Jedes Python Programm enthält in den ersten Zeilen einen Header in dem der Name, Autor, eine kurze Beschreibung enthalten sind.
- Der Code soll sauber geschrieben und wichtige Code-Zeilen kommentiert werden. Der Output sollte genauso wie in der Angabe aussehen!

```
"""
<Name>
<Description>
<Mat-Nr>
"""
```

U3Bsp1

Gegeben ist eine Python Datei mit einer Funktionen `U3Bsp1_exception_handling.py`
Testen Sie die Robustheit des Programms (Falscheingaben, usw.). Bauen Sie Exception Handling in den Code ein. Verwenden Sie spezifische Exceptions (auch mehrere) für kritische Code-Teile (z.B. `ValueError` bei int Casting; oder `ZeroDivisionError` bei Division durch 0)
das Programm soll nicht mehr durch Falscheingaben oder andere Fehler abstürzen!

Output:

```
Please enter number: 5
Another number: 2
5 / 2 = 2.5
Please enter number:
Falsche Eingaben - Versuch gescheitert!
Please enter number: a
Falsche Eingaben - Versuch gescheitert!
Please enter number: 10
Another number: 0
Falsche Eingaben - Versuch gescheitert!
```

U3Bsp2

Bauen Sie in ihren Code aus der U2Bsp6 so um, dass Sie Lottoeingaben mit Fehlerbehandlung haben. Sobald eine Fehleingabe bei den 6 Userzahlen erfolgt, wird der Fehler abgefangen, wird eine Meldung ausgegeben und muss der User die Zahl erneut eingeben.
Überprüfen Sie ihr Programm mit folgenden Fehleingaben:
Geben Sie einen Buchstaben statt einer Zahl ein
wenn Enter gedrückt wird (Input leer ist), soll der User aufgefordert werden, die Zahl erneut einzugeben

Hinweis: Falls Sie die Fehlerbehandlung bereits im U3Bsp6 Code integriert haben, geben Sie ihr Programm trotzdem nochmals ab.

Übungsblatt 3

Abgabe als ZIP-Datei **Deadline**: 23.10.2020 23:55 Uhr

U3Bsp3

Ändern Sie U3Bsp2 wie folgt um

- Erstellen Sie eine Funktion, welche automatisiert (ohne Userinput) überprüft wie viele Versuche es benötigt einen Lotto-6er zu erreichen
- Dabei wird eine zufällige, statische Liste von Lottozahlen gegen zufällige, neue Lottozahlen verglichen. Die statische Liste wird bei jedem Funktionsaufruf neu erstellt (vgl. Benutzer wählt neue Zahlen)
- Die Vergleichsliste wird nach jedem Nicht-6er neu erstellt und verglichen. Hierbei soll ein Zähler bei jedem Versuch erhöht werden

Wie viele Versuche benötigt das Programm um einen Lotto-4er zu erreichen (da es bis zu 10 Minute oder länger dauern könnte einen Lotto-6er zu erreichen.

Tipp:

- Gezogene Zahlen werden solange neu generiert bis diese gleich der Userzahlen sind
- Counter pro gezogene Zahlen erhöhen

[Optional]: Berechnen Sie wie lange es dauert in Sekunden einen Lotto 4er zu erreichen.

Output

```
Ihre Gewinnerzahlen: [41, 34, 15, 3, 10, 44]    Versuche: 119859
Time: 1.35s
```

U3Bsp4

Generieren Sie ein sicheres Passwort, welches folgende Kriterien erfüllen muss:

- Passwortlänge mind. 10 Zeichen lang
- Es müssen mind.
  - zwei Großbuchstaben
  - zwei Zahlen
  - zwei Sonderzeichen vorkommen

Geben Sie abschließend das generierte Passwort und die Länge aus.

Hinweis: `random.choice()` https://docs.python.org/3/library/random.html#random.choice

Mögliche Varianten mittels string-Klasse:

```
import string

print(string.ascii_letters)    # abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
print(string.digits)           # 0123456789
print(string.punctuation)      # !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Output:

```
Passwort: U%"+9bE34W*
Länge: 11
```

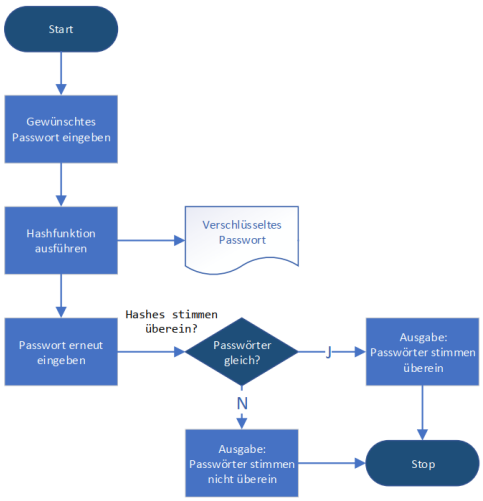
Übungsblatt 3

Abgabe als ZIP-Datei **Deadline**: 23.10.2020 23:55 Uhr

U3Bsp5

Programmieren Sie folgenden Ablauf nach (siehe Abbildung).
Es soll ein gewünschtes Passwort via Input eingelesen und gehahst werden. Danach soll das Passwort nochmals eingegeben und gehasht werden sowie gegen die erstellten Hashes auf Übereinstimmung überprüft werden.
Erstellen Sie dafür zwei Funktionen:

- hash_password (eingegebes Passwort wird gehasht mit SHA-256 und der Hash zurückgegeben)
- check_password (erneut eingegebenes Passwort wird ebenfalls gehasht und überprüft ob beide gleich sind)



Output

```
Please enter required password: Test12#
Hash: 99caf4ebc7778fb16c265c14d1a98e51910e971b18daf211d75fa2997c7a0e55:c902680a1b464300951d4095296ac9c3
Re-enter new password: Test12#
Success - the entered passwords match.
Please enter required password: Test123#
Hash: 95cd600a667d5a1275bc35c7616254cf7a729083743271d531851ddd1a9cdf2b:3d1f0187a15049ea82b85445c4d0f643
Re-enter new password: Test1234#
Error - The passwords do not match.
```

U3Bsp6

Gegeben ist die Klasse Kreis:

```
class Kreis:
    def __init__(self, radius):
        self.radius = radius
```

Fügen Sie folgende Funktionen (mit Ausgabe) hinzu

- Flächeninhalt berechnen
- Umfang berechnen

Erstellen Sie zusätzlich die Klassen „Rechteck“ und „Quadrat“ mit den dazugehörigen Funktionen (Umfang, Fläche)
Math Module verwenden:

```
import cmath
cmath.pi()
```

Testen Sie das Programm indem Sie mehrere Klassenobjekte erstellen, z.B.:

```
k1 = Kreis(7.1)
k2 = Kreis(4.8)
r1 = Rechteck(3.1, 7.4)
q1 = Quadrat(6)
```

Geben Sie von diesen Objekten die jeweiligen berechneten Inhalte aus

Output:

```
Kreis:      Fläche: 158.37, Umfang: 44.61
Rechteck:   Fläche: 22.94, Umfang: 21.0
Quadrat:    Fläche: 36, Umfang: 24
```

U3Bsp7

Erstellen Sie eine Klasse Person mit den Attributen (Name, Geb, PLZ, Alter)
Generieren Sie drei Instanzen dieser Klasse anhand folgender Aufrufe:

```
p1 = Person('Alfred Maurer', '08.06.2001', 3100)
p2 = Person('Lisa Winkler', '22.05.1992', 1100)
p3 = Person('Ernst Hindler', '01.10.1919', 1120)
```

Übungsblatt 3

Abgabe als ZIP-Datei **Deadline**: 23.10.2020 23:55 Uhr

Berechnen Sie das Alter (nur anhand Geburtsjahr) - erstellen Sie dafür eine Klassenfunktion calcAge(self)
Geben Sie anschließend alle Daten der Person via __str__ Methode aus

Tipps:
```
import datetime
today = datetime.date.today()
```

Output

```
Alfred Maurer, geboren am 08.06.2001, wohnhaft in 3100 ist 19 Jahre alt.
Lisa Winkler, geboren am 22.05.1992, wohnhaft in 1100 ist 28 Jahre alt.
Ernst Hindler, geboren am 01.10.1919, wohnhaft in 1120 ist 101 Jahre alt.
```

U3Bsp8

Ein Wellnesscenter verwendet eine ID-Card (Wertkarte mit Guthaben) um den Kunden die Verrechnung einzelner Leistung zu vereinfachen.
Erstellen Sie die Klasse `IDCard` - zu einer ID-Card sind folgende Informationen gespeichert:

- eindeutige (fortlaufende) Nummer (int)
- Name des Inhabers (str)
- Geburtstag (str)
- Guthaben (float)

Die Karten unterstützen folgende Funktionen:

Konstruktor

- Erzeugen der ID-Card mit der Nummer, einem Anfangsbetrag, sowie Name des/r InhaberIn und dem Geburtstag.

Methoden

- *chargeCredit(betrag)*
  - Aufladen eines Betrages; der Betrag wird zum aktuellen Guthaben dazu gezählt. Der Betrag muss mindestens 5€ sein und das Gesamtguthaben darf 200€ nicht übersteigen.
- *bookService(services, extraServices)*
  - Folgende Möglichkeiten (Services) der Abbuchungen gibt es:
    - Services:
      - ✓ 0: Tageskarte für Erwachsene (Alter >=18) kostet 25€
      - ✓ 1: Tageskarte für Studierende kostet 18€
      - ✓ 2: Tageskarte für Kinder kostet 12€
    - Extra Services
      - ✓ 0: kein extra Service
      - ✓ 1: Sauna zubuchbar kostet 5€
      - ✓ 2: Private SPA zubuchbar 10€

    z.B.: Tageskarte für Erwachsener mit Sauna -> `bookService(0, 1)`
    z.B.: Tageskarte für Studierende ohne extra Service -> `bookService(1, 0)`

  - Alle gebuchten Services werden in einer Liste als History hinzugefügt

- *__str__():*
  - Ausgabe aller Information
  - inkl. History der gebuchten Services (Liste)

Schreiben Sie eine Klasse IDCard. Legen Sie in der Main-Funktion mind. 5 Kunden in ihrem Testprogramm an und testen Sie entsprechend die Funktionalitäten.

Übungsblatt 3

Abgabe als ZIP-Datei **Deadline**: 23.10.2020 23:55 Uhr

z.B.: Kunde1

```
w = Wellnesscenter(1, 'Oliver', '19.10.1932', 150)
print(w)
w.chargeCredit(20)
w.bookService(1,0)
print(w)
```

Output

```
Name: Oliver, Geburtstag: 19.10.1932, Verfügbares Guthaben: 150.0€, gebuchte
Services: no services
Guthaben wurde aufgeladen - aktueller Stand: 170.0
Service Tageskarte Studierende wurde erfolgreich gebucht
Name: Oliver, Geburtstag: 19.10.1932, Verfügbares Guthaben: 152.0€, gebuchte
Services: Tageskarte Studierende
```