OST
Ostschweizer
Fachhochschule

Python for Machine Learning Übungen

Part 1

OST
Ostschweizer Fachhochschule

Oberstrasse 10
8640 Rapperswil
Switzerland

T +41 79 350 24 36
michael_inden@hotmail.com
ost.ch

Michael Inden

1 Aufgaben Quick Start & Grundlagen

Aufgabe 1: Mathematische Berechnungen

Nehmen wir an, wir hätten eine Spedition. Wir bekommen einen Großauftrag und müssen 1.000 Bücherkisten ausliefern. In unseren Lkw passen pro Fahrt jedoch nur 75 Kisten. Berechnen Sie, wie oft wir fahren müssen und wie viele Kisten in der letzten Fahrt transportiert werden. Verwenden Sie sprechende Variablennamen.

Aufgabe 2: Schleifen mit variabler Schrittweite

Nutzen Sie eine Schleife, die beim Wert 0 und mit einer Schrittweite von 1 startet. Bei jedem Schleifendurchlauf soll der Wert um die Schrittweite erhöht werden und die Schrittweite wird jeweils um eins erhöht. Geben Sie die beiden Werte aus, solange die Schleifenvariable kleiner als 60 ist.

Aufgabe 3: Teiler

Schreiben Sie eine Funktion `find_proper_divisors(num)`, die alle echten Teiler einer natürlichen Zahl berechnet, also diejenigen Zahlen ohne die Zahl selbst. Für die Zahl 12 würden als echte Teiler die folgenden Werte ermittelt: 1, 2, 3, 4 und 6.

Aufgabe 4: Verschachtelte Schleifen – Variante 1

Schreiben Sie eine Funktion `print_number_triangle(row)`, die eine mehrzeilige Ausgabe bis zur übergebenen maximalen Zeilenanzahl wie folgt erzeugt:

```
1
1 2
1 2 3
1 2 3 4
```

KÜR: Als Kür soll eine Funktion geschrieben werden, deren Ausgabe wie folgt aussieht:

```
1
2 3
4 5 6
7 8 9 10
```

Aufgabe 5: Verschachtelte Schleifen – Variante 2

Implementieren Sie eine Funktion, um Buchstaben in folgendem Muster auszugeben:

```
A
BB
CCC
DDDD
```

Tipp: Einzelne Zeichen kann man mit der Funktion `ord()` in eine Zahl wandeln und mit `chr()` eine Zahl in einen Buchstaben:

```
>>> ord("A")
65
>>> chr(77)
'M'
```

Aufgabe 6: Befreundete Zahlen

Zwei Zahlen n_1 und n_2 heißen befreundet, wenn die Summe ihrer Teiler der jeweils anderen Zahl entspricht:

$$\text{sum}(\text{divisors}(n_1)) = n_2$$
$$\text{sum}(\text{divisors}(n_2)) = n_1$$

Schreiben Sie eine Funktion `calc_friends(max_exclusive)` zur Berechnung aller befreundeten Zahlen bis zu einem übergebenen Maximalwert.

Tipp: Nutzen Sie die in Aufgabe 3 erstellte Funktionalität. Was wird dabei für den Import von Modulen deutlich?

2 Aufgaben zu Strings

Aufgabe 1: Länge, Zeichen und Enthaltensein

In dieser Aufgabe geht es darum, grundlegende Funktionalität der Strings anzuwenden. Sie sollen die Länge eines Texts abfragen, dann ein Zeichen an einer beliebigen Position, etwa der 13., ermitteln und schließlich prüfen, ob ein gewünschtes Wort im String enthalten ist.

Aufgabe 2: Zeichen wiederholen

Bei dieser Aufgabe geht es darum, die Buchstaben eines Worts gemäß ihrer Position zu wiederholen, aus ABC wird dann ABBCCC. Schreiben Sie dazu eine Funktion `repeat_chars(input)`. Schreiben Sie einen Unit Test, idealerweise einen Parameterized Test. Infos finden Sie unter https://docs.pytest.org/en/6.2.x/index.html und https://docs.pytest.org/en/6.2.x/parametrize.html.

Aufgabe 3: Vokale raten

Bei dieser Aufgabe werden einige etwas ältere Leser sich vielleicht an die Sendung »Glücksrad« erinnern, bei der es genau um das Erraten von Wörtern, Sätzen oder Redewendungen ging, in denen Vokale fehlten.

a) Als Erstes sollen in einem gegebenen String mithilfe der selbst geschriebenen Funktion `remove_vowels(input)` alle Vokale entfernt werden.

b) Als Zweites soll eine Funktion `replace_vowels(input)` implementiert werden, die einen Vokal durch ein '_' ersetzt, damit wir für ein kleines Ratespiel einen Hinweis auf einen entfernten Vokal bekommen.

```
remove_vowel("Es gibt viel zu entdecken!") => s gbt vl z ntdckn!
replace_vowel("Es gibt viel zu entdecken!")=> _s g_bt v__l z_ _ntd_ck_n!
```

Aufgabe 4: Reverse String

Schreiben Sie eine rekursive Funktion `reverse_string(input)`, die die Buchstaben des übergebenen Eingabetexts umkehrt. Es darf nicht auf die Standardfunktionalität `[::-1]` zurückgegriffen werden.

Eingabe	Resultat
"A"	"A"
"ABC"	"CBA"
"abcdefghi"	"ihgfedcba"

Tipp: Zur Erinnerung: Rekursion meint Selbstaufrufe, etwa wie folgt: fac(n) = n * fac(n) – 1

Aufgabe 5: Palindrom-Prüfung

Schreiben Sie eine Funktion `is_palindrome(input)`, die überprüft, ob ein gegebener String unabhängig von Groß- und Kleinschreibung ein Palindrom ist. Erinnern wir uns: Ein Palindrom ist ein Wort, das sich von vorne und von hinten gleich liest.

Eingabe	Resultat
"Otto"	True
"ABCBX"	False
"ABCXcba"	True

Aufgabe 6: Doppelte Buchstaben entfernen

Schreiben Sie eine Funktion `remove_duplicates(input)`, die in einem gegebenen Text jeden Buchstaben nur einmal behält, also alle späteren doppelten unabhängig von Groß- und Kleinschreibung löscht. Dabei soll aber die ursprüngliche Reihenfolge der Buchstaben beibehalten werden.

Eingabe	Resultat
"bananas"	"bans"
"lalalamama"	"lam"
"MICHAEL	"MICHAEL"

Aufgabe 7: Anagramm

Als Anagramm bezeichnet man zwei Strings, die dieselben Buchstaben in der gleichen Häufigkeit enthalten. Dabei soll Groß- und Kleinschreibung keinen Unterschied machen. Schreiben Sie eine Funktion `is_anagram(str1, str2)`.

Eingabe 1	Eingabe 2	Resultat
"Otto"	"Toto"	True
"Mary"	"Army"	True
"Ananas"	"Bananas"	False

Aufgabe 8: Pattern Checker

Schreiben Sie eine Funktion `matches_pattern(pattern, text)`, die einen mit Leerzeichen separierten String (2. Parameter) auf die Struktur eines Musters hin untersucht, das in Form einzelner Zeichen als erster Parameter übergeben wird.

Eingabe Muster	Eingabe Wörter	Resultat
"xyyx"	"tim mike mike tim"	True
"xyyx"	"tim mike tom tim"	False
"xyxx"	"tim mike mike tim"	False
"xxxx"	"tim tim tim tim"	True

Aufgabe 9: Lineal

In dieser Aufgabe wollen wir ein Lineal im englischen Stil nachahmen. Dabei wird ein Bereich von einem Inch in 1/2 und 1/4 sowie 1/8 unterteilt. Dabei nimmt die Länge der Striche jeweils um eins ab.

---- 0

-

--

-

-

--

-

---- 1