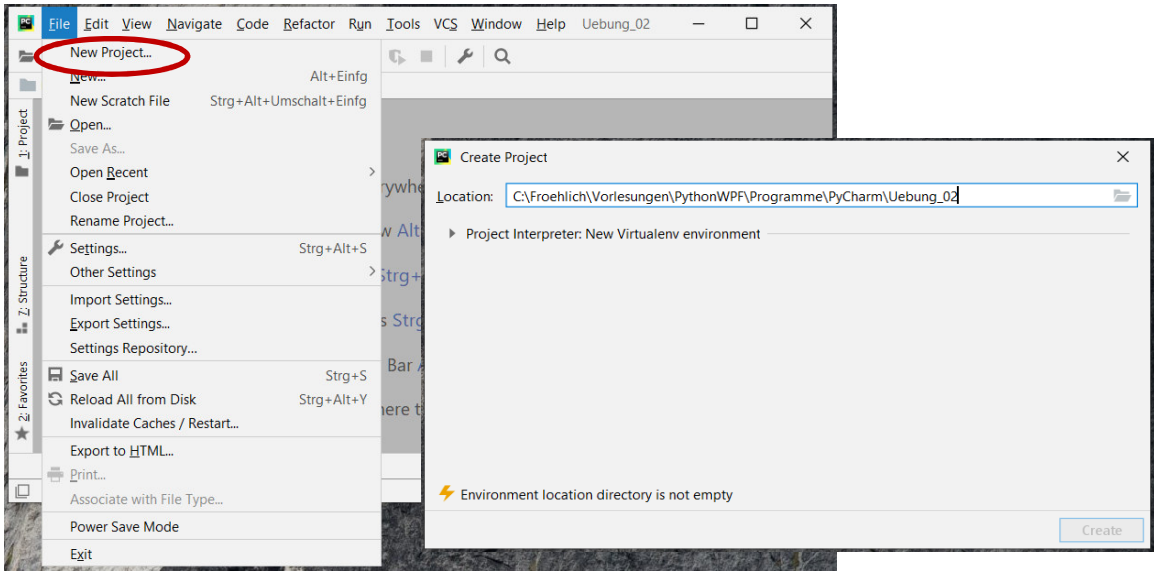
Übungsblatt 2 (Hallo Python)

Aufgabe 1:

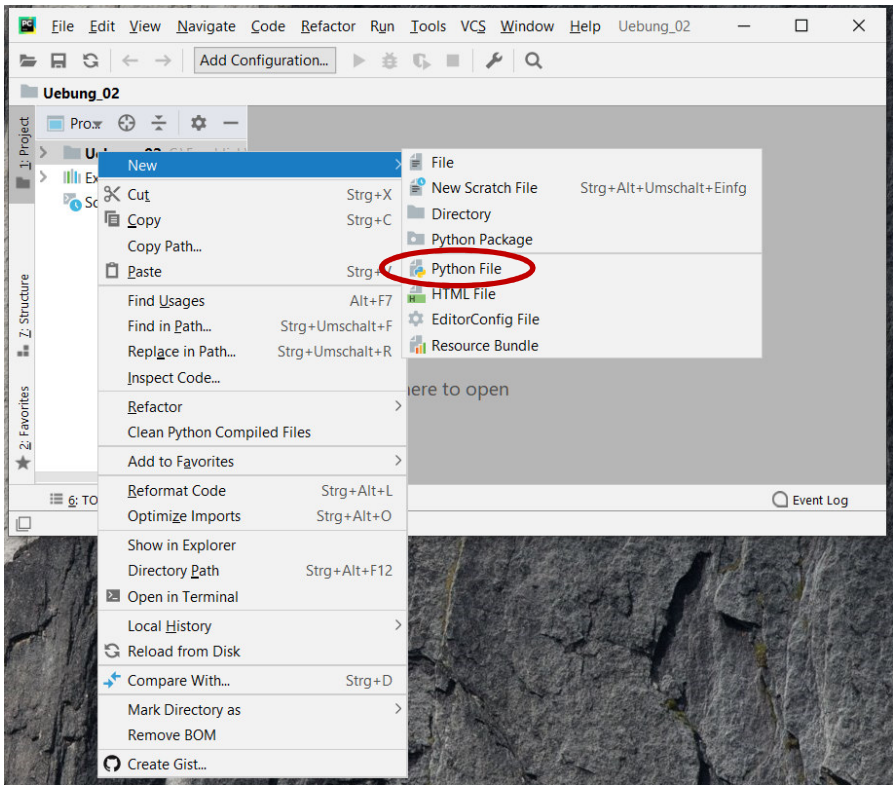
Für alle weiteren Aufgaben ist es sinnvoll eine komfortablere Entwicklungsumgebung zu installieren!

a) Laden Sie sich zunächst den Installer der Community-Version von PiCharm der Firma JetBrains herunter und installieren Sie PiCharm auf Ihrem Rechner:
https://www.jetbrains.com/pycharm/download/#section=windows

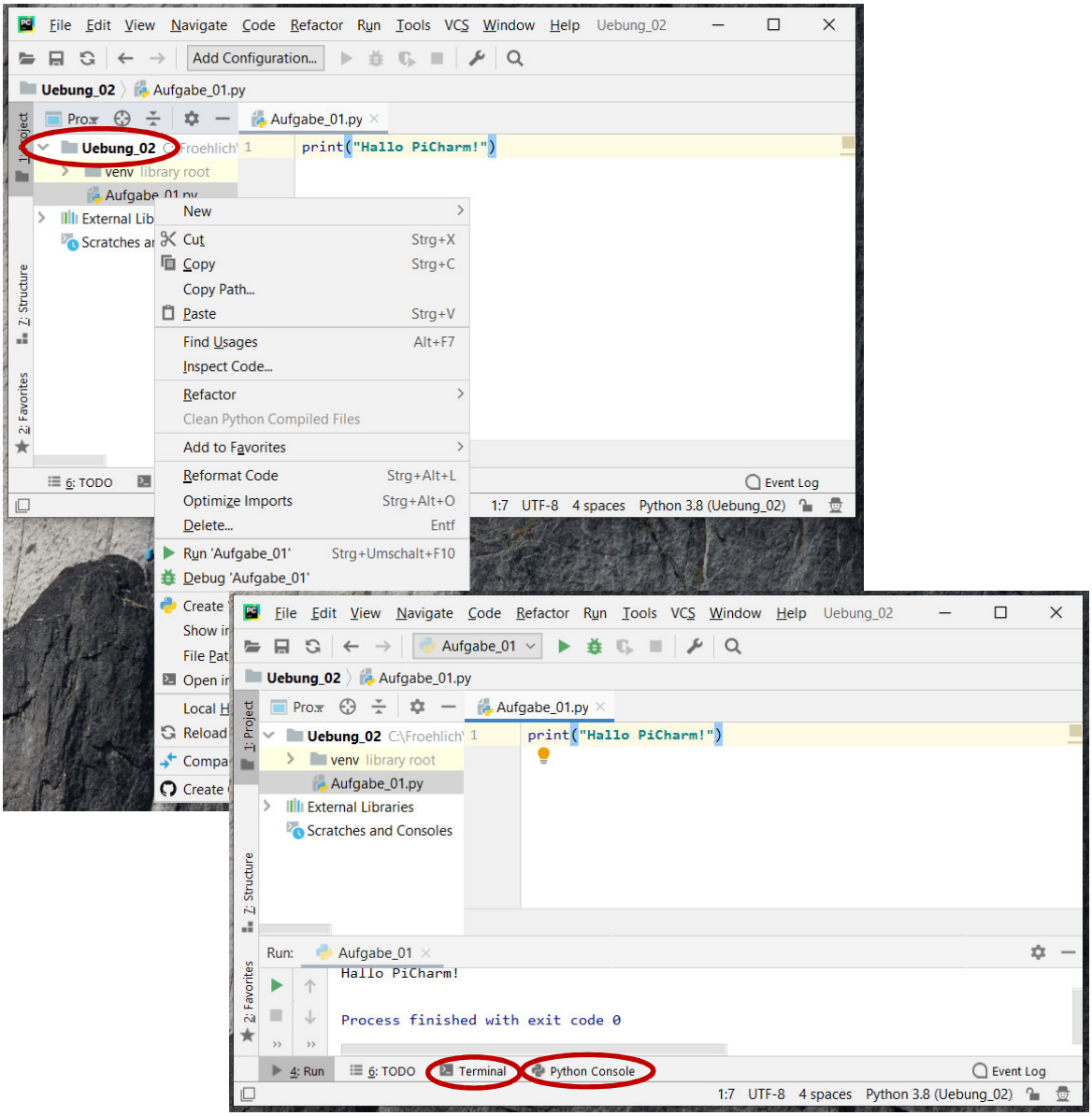
b) Starten Sie PiCharm und erzeugen Sie ein neues Projekt z.B. „Uebung_02“ unter New Project:



c) Erzeugen Sie ein Python-File und schreiben Sie z.B. print("Hallo PiCharm!") in die 1. Zeile.



d) Öffnen Sie den Folder von Uebung_02, öffnen Sie das Menue durch Rechtsklick auf Aufgabe_01 und starten Sie den Interpreter durch Run 'Aufgabe_01'. Die Ausgabe wird emuliert:



e) Öffnen Sie die Python Console, Markieren und kopieren Sie Ihre Zeil(en) im Editor und fügen Sie diese in der Console ein. So können Sie Code-Sequenzen testen und verändern.

f) Öffnen Sie das Terminal und aktualisieren Sie für Ihre Arbeitsumgebung pip:
`<venv> C:\...\Uebung_02> python -m pip install --upgrade pip`
Aus dem Terminal können Sie auch Ihr Programm starten: python Aufgabe_01.py.

g) Testen Sie die anderen Möglichkeiten das Programm zu starten und zu debuggen
Hinweis: Um von der Anzeige der Variablen auf die Ausgabe zu wechseln klicken Sie auf Console!

h) Öffnen Sie eine Eingabeaufforderung außerhalb von PiCharm, gehen Sie in den Pfad des Projekts und führen Sie python Aufgabe_01.py dort aus. Wenn Sie so weit gekommen sind funktioniert Ihre Entwicklungsumgebung einwandfrei und Sie können mit Aufgabe 2 fortfahren!

Aufgabe 2:

In dieser Aufgabe sollen Sie sich mit den Built-In Funktionen vertraut machen (siehe Anhang der Vorlesung) bzw.: https://docs.python.org/3/library/functions.html

a) Scheiben Sie ein Python-Script, dass eine beliebige Zahl (als String) einliest und dann in der hier angegebenen Reihenfolge in folgende Datentypen umwandelt und die gewandelten Werte mit Typangabe ausgibt:

- Komplexe Zahl aus dem String (evtl. verwenden Sie hier auch try/except um das Script mit exit() zu verlassen falls kein sinnvolles Zahlenformat eingegeben wurde).
- Reelle Zahl aus dem Betrag (der Länge) der Komplexen Zahl, auf 4 Stellen gerundet.
- Ganze Zahl aus der Reellen Zahl
- Hexadezimalzahl und Binärzahl
- Die ganze Zahl Als Schriftzeichen (Achtung die Obergrenze ist 0x110000)

b) Schreiben Sie ein Python-Script, das eine Textzeile einliest und danach folgende Ausgaben generiert:

- Die Zeile als Liste
- Die Liste als Tupel
- Die Zeile als sortierte Liste
- Die Länge der Zeile
- Den Minimal- und Maximalwert der Zeile als Zahlenwert
- Für jedes Zeichen der Zeile entsprechend dem Beispiel für „Test“ folgende Angaben:

```
T    84      'T' b'T'
e    101     'e' b'e'
s    115     's' b's'
t    116     't' b't'
```

Benutzen Sie dazu ord(), ascii() und bytes(). Wählen Sie für bytes() eine Codierung aus:

https://docs.python.org/3/library/codecs.html#standard-encodings

Aufgabe 3:

In dieser Aufgabe beschäftigen Sie sich mit den Methoden, die für Sequenzen zur Verfügung stehen! Diese sind für die unterschiedlichen Sequenzen (String, List, Tupel…) verschieden!

a) Obwohl Strings unveränderlich sind gibt es sehr viele Methoden zum Umgang mit Strings. Dabei existieren auch Methoden, die scheinbar den String verändern, jedoch wird der ur-sprüngliche String nicht verändert, sondern die Methode gibt eine veränderte Kopie des Strings zurück! Machen Sie sich mit einigen der Methoden vertraut:

https://docs.python.org/3.8/library/stdtypes.html#string-methods

Schreiben Sie ein Python-Script das eine Texteingabe entgegennimmt, um dann folgende Methoden des Strings anzuwenden:

- Untersuchen Sie die Unterschiede von casefold(), lower(), upper() und title()
- Ersetzten Sie mit replace() alle Leerzeichen durch Komma, Tabulator bzw. Zeilenvorschub
- Was unterscheidet split() und splitlines()? Zerlegen Sie den String unter deren Verwendung
- Machen Sie sich mit strip() vertraut und machen Sie eine Beispiel

b) Schreiben Sie ein Python-Script in dem Sie die Methoden center(), join() und expandtabs() verwenden, um eine Tabelle zu realisieren, die folgendes Aussehen hat:

```
____Name___|___Ort____|___Land___
Thomas     |Berlin    |Germany
Britney    |Sydney    |Australia
```

c) Da Listen veränderlich sind wirken viele Methoden der Liste direkt auf die Liste selbst! Machen Sie sich mit möglichst vielen Methoden zu Listen vertraut:

https://docs.python.org/3/tutorial/datastructures.html

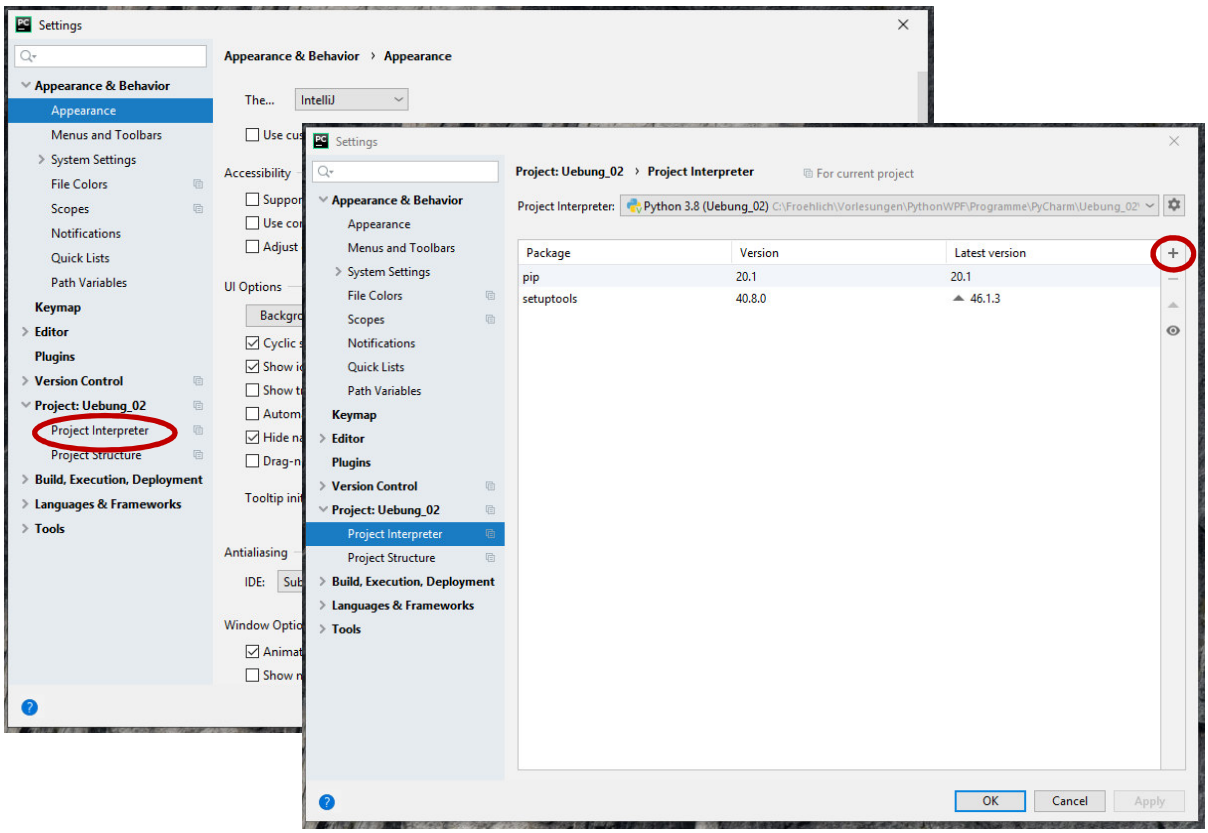
- Erzeugen Sie eine Liste mit der Zahl 0.
- Hängen Sie an die Liste mit append() die Zahlen 1,2,3 an.
- Erweitern Sie die Liste mit extend() um die Zahlen 1,2,3. Was ist der Unterschied?
- Entfernen Sie die Liste [1,2,3] aus der Liste mit remove[].
- Weisen Sie die Liste der Variablen rListe zu.
- Legen Sie eine Kopie der Liste mit copy() in der Variablen cListe an.
- Hängen Sie am Anfang der Liste die Zahlen -3, -2, -3 mit insert() an (keine Liste in Liste!).
- Ermitteln Sie den Index der Zahl 0 mit index(), um die Zahl 0 mit pop() zu entfernen.
- Geben Sie alle drei Listen aus.
- Löschen Sie die originale Liste mit clear().
- Geben Sie alle drei Listen aus. Was haben Sie beobachtet?

d) Legen Sie ein Wörterbuch an, mit dem Sie ein paar deutsche Worte ins Englische übersetzten können.
Versuchen Sie für das Wörterbuch automatisch auch die Rückübersetzung zu generieren.
Schreiben Sie ein Skript mit dem Sie ein deutsches oder englisches Wort eingeben können, um dann die Übersetzung auszugeben.
Versuchen Sie das Programm so zu ändern, dass ein ganzer Satz übersetzt werden kann, für den alle Worte in Ihr Wörterbuch vorkommen!

Aufgabe 4: Wer sind meine Nachbarn?

In dieser Aufgabe geht es darum, eine Matrix der Größe NxN zu erzeugen und mit zufälligen Werte zu füllen, um dann die Summe aller Nachbarn zu einer gegebenen Position zu ermitteln.

Zur Vorbereitung müssen Sie in Ihrer Arbeitsumgebung das Modul **numpy** einbinden! Gehen Sie dazu über File in die Settings-Maske Ihres Projektes und Öffnen Sie den Projekt Interpreter:



Klicken Sie auf das Pluszeichen, suchen Sie nach **numpy** und installieren Sie dieses Package.

a) **Jetzt kann es losgehen:**

- Importieren Sie random, numpy und math
- Erzeugen Sie eine leere Liste
- Fragen Sie den Benutzer nach der NxN Größe der Matrix
- Hängen Sie mit random.randint() NxN-Mal eine zufällige Zahl zwischen -9 und 9 an die Liste an
- Machen Sie aus der Liste mit numpy.array() ein Array
- Nutzen Sie die Methode reshape(n,n) um aus dem Array eine NxN-Matrix zu machen
- Geben Sie die Matrix mit print(Matrix) aus
- Nach Eingabe einer Position ist nun die Summe der Nachbarn zu ermitteln:
    i. Prüfen, ob die Position keine Randposition ist!
    ii. Die Zahlenwerte aller 8 Nachbarn zusammenzählen
    iii. Die Summe ausgeben
- Evtl. ermöglichen eine neue Position zu wählen

b) Machen Sie sich Gedanken darüber, wie Sie es ermöglichen können, für jede beliebige Position in der Matrix die Summe der Nachbarn zu ermitteln, also auch alle Position ganz am Rand (die Linken Nachbarn der 1. Spalte wären dann die Felder der letzten Spalte, die oberen Nachbarn der ersten Reihe wären dann die Felder der letzten Reihe und umgekehrt)!