**Windows Subsystem Linux -WSL**

:

Test-Text

Erstes Gespräch mit Martin Elshuber

Gespräch war am 30.6. 2024

Ich habe großteils nur zugeschaut und versucht soviel wie möglichmitzubekommen

Dateien sind gemounted über /mnt/d/wj/pcharm1/testPdf.py

d.h. das System kriegt nichts mit wenn man Änderungen durchführt

Eswird diverses installiert, wie zuletzt

* sudo apt install xdg-utils
* sudo apt install evince

vorher diverses von derToolbox von jetbrains: (wieder Reihenfolge verkehrt!)

* .local/share/JetBrains/Toolbox/aexit
* .local/share/JetBrains/Toolbox/aquit
* .local/share/JetBrains/Toolbox/apycharm
* .local/share/JetBrains/Toolbox/a.local/share/JetBrains/Toolbox/apps/pycharm-community/bin/pycharm.sh
* .local/share/JetBrains/Toolbox/arm -rf .venv
* .local/share/JetBrains/Toolbox/acd pcharm1/
* .local/share/JetBrains/Toolbox/acp -rv /mnt/d/wj/pcharm1 .

Dazwischen diverses und vorher

* .local/share/JetBrains/Toolbox/asudo apt install python3-pip
* .local/share/JetBrains/Toolbox/acd pcharm1/
* .local/share/JetBrains/Toolbox/awsl

Ist nicht mehr so ganz klar

* .local/share/JetBrains/Toolbox/apycharm
* .local/share/JetBrains/Toolbox/arm -rf .venv/
* .local/share/JetBrains/Toolbox/adu -sh .venv/
* ….
* .local/share/JetBrains/Toolbox/adu -sh /mnt/d/wj/pcharm1/
* .local/share/JetBrains/Toolbox/acp -r /mnt/d/wj/pcharm1 .
* .local/share/JetBrains/Toolbox/als /mnt/d/wj/pycharm1/pythonProject/
* .local/share/JetBrains/Toolbox/als /mnt/d/wj/pycharm1

Jetzt wird wieder vom Systeminstalliert:

* .local/share/JetBrains/Toolbox/asudo apt install tree
* .local/share/JetBrains/Toolbox/arm -rf pythonProject/.venv
* …
* .local/share/JetBrains/Toolbox/acp -r /mnt/d/wj/pycharm1 .
* .local/share/JetBrains/Toolbox/a~/.local/share/JetBrains/Toolbox/apps/pycharm-community/bin/pycharm.sh
* ??? anano .bahsrc
* jetbrains-toolbox
* sudo apt install libfuse2 libxi6 libxrender1 libxtst6 mesa-utils libfontconfig libgtk-3-bin
* nano .profile
* mv jetbrains-toolbox-2.3.2.31487/jetbrains-toolbox .local/bin/
* mkdir -p .local/bin
* tar -xzvf jetbrains-toolbox-2.3.2.31487.tar.gz
* mv /mnt/c/Users/weiser/Downloads/jetbrains-toolbox-2.3.2.31487.tar.gz .
* rm -rf pycharm-community-2024.1.4.tar.gz
* tar -xzvf pycharm-community-2024.1.4.tar.gz
* …
* mv pycharm-community-2024.1.4.tar.gz ~
* sudo apt install libfuse2
* sudo apt install xterm
* man -k apt-get
* man -k man
* man man
* apt-get –h
* help apt-get
* sudo apt -y –h
* sudo -h –y
* sudo apt install –y
* sudo apt –y
* sudo apt -y –h
* sudo apt list

Fragen:

* Letzte 50 Kommandos auflisten?
* Was ist **".local/share/JetBrains/Toolbox/arm "**
* Wie kommst du zur Toolbox in Linux?

WSL-History.txt

Die letzten 50 Kommandos auflisten geht mit history Kommando

Einstieg in Linux

Windows Power Shell aufrufen

Dann wsl eingeben

User weiser, password "Pauli" oder "pauli"

Man muss zuerst die Toolbox aufrufen: **jetbrains-toolbox** steht in

/home/weiser/.local/bin/jetbrains-toolbox

Beschreibung der Rechnungsgenerierung

Source analyseData.py

Variable rowstr

Sollte eigentlich columnString heißen, List der Spaltenüberschriften, enthält allerdings nur 5 Namen

Methode createPart1

Eingabe ist eine Datei "…ÜbersichtV1.csv. Ausgabe ist eine Datei part1.csv. Die Übersichtsdatei aus EDA wird gespalten 2 Teile. Aus dem ersten Teil wird nur der Zeitraumgenomen. Im zweiten Teil ..part2.csv. warden die Daten für jeden Teilnehmer genommen.

Methode createPart2

Wie oben bei createPart beschrieben.

Methode readList1

readList1 wird gelesen (…part1). Es wird Zeitraum ermittelt.

Methode readList2

Es wird hier die **Variable res** erzeugt. Sie enthält für jeden Teilnehmer eine Liste der Komponenten der in der Zusammenfassung gespeicherten Daten.res wird zurückgegeben!

Methode saveInput

Es wird die Variable res weitergegeben und eswird eine ganz ordentliche erste Ausgabe prodiuziert mir den Daten von res.

In Methode saveInput warden die **Variablen tln, dictTlsn und interm** definiert, allerdings nicht weiter verfolgt.

Methode test()

Eigentlich das Hauptprogramm, funktioniert aber nicht mehr, leicht veraltet. Die test-Methode sollte allerdings nicht im Hauptprogramm stehen, da diese Methde dann beim Importieren ausgeführt wird . Das isrt aber nicht beabischtigt.

Souce csvTest1.py

Methode convertFileV2

wandelt eine .xlsx in eine .csv-Datei, allerdings wird da **Sheet 0** genommen (ausgewählt). Man kann keine Zeilen überspringen

Es wird da das **Paket spire.xls.common** genommen, ich weiß nicht,ob das eine gute Wahl ist.

Methode convertFileV1

Wie bei convertFileV1, allerdings kann man da eine Anzahl Zeilen überspringen (skiprows).

Methode singleReportLinux

Es warden eine Reihe von Methode aufgerufen (converFileV1, createPart1, createPart2, readList1, readList2, saveInput), zuletzt wird auch res zurückgegeben

Methode singleReport

Hernach sollte eine Methode single Report definiert sein, diese wurde aber vergessen. War ursprünglich definiert, aber dannzu singleReportLinux umgestaltet, aber das Original vergessen!

Methode startMethod1

Sollte für Methode startMethod1 Variable setzen und dann die Methode starten, wurde aber in Kommentar gesetzt

Methode startMethodLinux

Das ist eher aktuell (Variable setzen und Methode startReportLinux starten.

Methode createInvoiceList

Wandelt .xlsx-Datei in csv-Datei um (über Methode convertFileV1), ruft dann Methode listNames auf

Methode listNames

Zeile   
 print(p['Anrede'])   
dürfte fehlt am Platz sein.

Es liest die csv-Datei mit den Kunden zeilenweise ein konvertiert das die html-Vorlage mit dem Kommando format und schreibt dann die Daten in ein directory "invoicedir", wobei der Dateiname aus Famiien- und Vorname besteht.

Die Methode wurde vordenVersuchen mit jinja geschrieben.

Source jinjaTest2

Methode startJinja

Ein Beispiel vom Primer jinja Templating einfach abgeschrieben zu mDurchdenken

Methode listNames

Eine Methode analog zur **Source csvTest1.py**

Methode converFileV3

Noch ein converFile-Methode, ziemlich ähnlich zu convertFileV1

Die Datenstrukturen

Die Bankdaten

Hier sind die privaten Daten gespeichert.

Die Daten sind in der Mitgliederliste\_Reidlinger gespeichert

Die Daten werden in einem Array abgelegt, jedes Array zeigt aus ein Dictionary für alle Komponenten

Das Array heißt **privateArray**, das zugehörige Dictionary heißt **privateDict**. Beim Dictionary ist der Zählpunkt der Schlüssel.

Das privatArray entsteht direct beim Einlesen, das privatDict muss extra aufgebaut werden

Die EDA-Daten

Wiederum werden die Daten als .xslx-File gespeichert. Dann werden sie in ein csv umgewandelt und dann mit dem csv.DictReader zeilenweise eingelesen, sodass auch die einzelnen Spalten vorhanden sind.

Man kann die Daten etwas reduzieren, sodas nur mehr die notwendigen Spalten bleiben und dann die geschriebenen Texte hinzufügenoder eventuell spezielle Methoden für die Umwandlung einführen. Es muss jedenfalls sichergestellt sein, dass sich die geschriebenen Werte nicht ändern.

Es wird wiederum ein Dictionary namens **EDA\_Dict** eingeführt. Der Zugriff ist wiederum über die Zählernummer.

Bei diesen EDA-Daten gibt es zwei Arten:

* Zählernummern, die einfach nur bezahlt werden
* Zählernummern, bei denen von einem PV-Produzenten ein Rabatt gewährt wird. In diesem scheint der Geldbetrag 2 Mal auf nämlich PV-Produzenten, welcher den Rabatt gewährt, und dem Kunden, welcher den Rabatt erhält.

privateList und edaList

privateList:

eine Liste von privateElem, eine privateElem ist eine Zeile eines Kunden. Das ist also ein dictionary mit verschiedenen Einträgen für einen Kunden (Name, email usw.)

edaList:

Die edaList stellt die Daten aus EDA-Sicht da. Für jeden Kunden gibt es pro Monat eine Statistik über den Verbrauch. Das ist auch eine Zeile pro Kunde. edaElem ist wiederum ein Dictionary also edaElem.

EdaElem symbolisiert einen Zählpunkt. Wenn mehrere Zählpunkte pro Haushalt vorhanden sind, dann werden die Zählpunkte zu einer Liste von edaElem zusammengefasst.

Zusammen verbinden:

Komponente "edaData" in der privatList. Das ist eine Liste

**[]**

(Reihenfolge egal, eher gering). Wenn eine Komponente daher kommt, dann wird die Liste vergrößert.

edaData ist wiederum eine Komponente im dictionary

privateList[0] oder als dictionary privateList["ZA002…Zählernummer"]

kommt man auf die privateElem.

Dann gibt es dort die Komponente edaData, nämlich mit Forsetzung

privateList["ZA002…Zählernummer"]["edaData"]

Dort will ich ein edaElem eintragen.

* Wenn ich das privatelem habe ich ja die Zählernummer.
* Mit der Zählernummer kann ich das edaElem holen
* Das privateElem wird ja über die Methode genannt
* Es wird eine Liste (zunächst nur ein Element) gemacht
* Dies wird dann in die Komponente edaData gespeichert

Jetzt probiere ich einmal den Code dazu.

Rabatte eintragen

mainZp:

der Zählpunkt, bei welchem der Rabatt eingetragen wird (Also Markus), dieser wird unter dem Namen **'mainZP'** eingetragen

rabattZp:

der Zählpunkt, welcher vom Rabatt profitiert, dieser seht unter **'Rabatt-Zaehlernummer'**

Grundsätzliche Vorgehensweise

Man geht die Liste der Rabatt Zeilen (Kennzeichen "R") in der Mitgleiderliste-Reidlinger durch.

Nun macht man für jede Rabatt-Zeile zwei Einträge:einmal in der mainZp (Komponente

privateElemGeber = self.privateDict[rabattGeberZp]  
privateElemNehmer = self.privateDict[rabattNehmerZp]

Die Liste sind als Komponente rabattGeber beim privateElementNehmer bzw

if 'rabattGeber' not in privateElemNehmer:  
 privateElemNehmer['rabattGeber'] = []  
privateElemNehmer['rabattGeber'].append(edaElem)

als rabattNehmer beim privateElementGeber definiert

if 'rabattNehmer' not in privateElemGeber:  
 privateElemGeber['rabattNehmer'] = []  
privateElemGeber['rabattNehmer'].append(edaElem)

Die Listenelemente sind als edaElem "deklariert", es muss da noch die "R"-Zeile (Rabatt-Zeile ) hinzugefügt warden. Was da in der Listegenau hinzugefügt wird, muss noch analysiert warden.

Das passiert alles in der **Methode addRabatt1 (Klassse GenerationData)** x

Klasse GenerationData

Diese Klasse gehört zwar auch zur Source invgen.py, aber ist eine eigene Klasse und daher wird diese Klasse eigens behandelt

Datenstrukturen

edaList:

Die Einträge bestehen Großteils aus einer Zeile, die mittels csv.DictReader gelesen wurde. Es ist eine Liste dieser Daten.

edaDict:

Ein Dictionary, mit welchem man mittels des Zählpunktes direct auf die EDA-Daten zugreifen kann

privateList:

Eine Liste, welche die Mitglieder der Energiegemeinschaft repräsentiert.

privateDict:

Ein dictionary mit welchem man direkt auf die zugehörigen Mitglieder zugreifenkann

Methode addEdaElem

Fügt ein edaElem, also Zeile vom csvDictReader hinzu, in edaList und edaDict

Methode addPrivateElem

Fügt ein Mitglied (also eine entsprechende Zeile aus dem DictReader zur privateList und zum privateDict hinzu.

Die Zählpunkte, welche demselben Mitglied gehören, werden in der edaListe von der privateListe gesammelt.

Bei der Methode hier wird ein neues MMMitglied angelegt und entsprechend hat dieList am Ende nur ein Element

Methode addZaehlpunkt

Es wird angenommen, dass das edaElem schon im edaDict vorhanden ist. Daher wird das edaElem nur in der edaListe des Mitglieds hinzugefügt

Source invgen.py

Das ist die neueste Version der Software, da ist alles zu einer Source zusammengefasst.

Die obigen Bechreibungen passen großteils für die Source invgen.py ("InvoiceGeneration")

Methode readEda2

Liest mit Hilfe des csv.DictReader die Daten einer Zeile der eda\_Daten ein

Es wird unterschieden zwischen CONSUMPTION und Generation.

Bei CONSUMPTION warden die Daten Verbrauch, VerbrauchText, preiBrutto und preisBruttoText gespeichert.

Bei GENERATION warden die Daten Lieferung und LieferungText (statt Verbrauch) gespeichert. Am Ende der Methode readEda2 sind die Daten vorhanden.

Am Ende der Analyse einer Zeile wird entsprechend die **Methode addEdaElem aufgerufen**.

Methode creatPrivate

Die Mitgliederliste wird als csv-File durchgelesen mit csv.DictReader.

Elemente ohne Folgezeilenkennzeichnung werdn einfach als privateElem eingetragen

createHtmlInvoice(self)

Die Methode ist wichtig zum Verstehen,

for privateElem in self.gd.privateList:

Das ist eine Schleife über alle Teilnehmer der Energiegemeinschaft, alle Mitglieder kriegen eine Rechnung. Diese Methode erzeugt aus der Vorlage eine html-Version der Rechnung.

Zunächst wird der Name der neuen Datei festgelegt.

Dann kommen die eda-Daten ("edaData") dran. Diese kommen aus der EDA-Datenbank. Da gibt es eine Zeile für jeden Zählpunkt.

Dann kommt die Liste der Rabatteinträge dran. Da gibt es die Liste rabattGeber und rabattNehmer

Als Beispiel hier: rabattGeber ist Markus und rabattNehmer ist Leopold(Vater)

Die Elemente der List rabattGeber und rabattNehmer sind alle ein dict mit Komponenten rabattNehmer, rabattGeber, edaData (die Daten des Nehmers, aus welcher hervorgeht wie hoch sein Zahlungen ohne Rabattte sind) und rabattZeile (die Beschreibung des Rabattes in einer Folgezeile vom Typ "R"(Rabatt) in der Mitgliederliste).

Die Elemente der Liste rabattGeber und rabattNehmer sind definiert im Dictionary rabattElem (siehe Methode addRabatt3)

Jedes dict rabattElem kommt zwei Mal je Rabatt vor. Einmal als rabattGeber und einmal als rabattNehmer. Jeder Rabatt erscheint in 2 verschiedenen Rechnungen (nämlich beim rabattGeber und beim rabattNehmer). Der Betrag ist jeweils gleich. Der Betrag wird zwei Mal reduziert. Beim rabattNehmer werden die Zahlungen reduziert und beim rabattGeber werden die Gutschriften reduziert. Damit werden die Rabatte unabhängig von der Energiegemeinschaft.

Nun werden die Rabatte (das Geld und der Strom) als Komponenten 'Verbrauch' und 'VerbrauchText' als negative Werte eingetragen.

Die Listen eda, rabattGeber und rabattNehmer dürfen nicht None sein. Sie werden in den HTML-Vorlagen verwendet um die Zeilen der Rechnung zu erzeugen. Wenn die Listen None sind dann kommt es zu einer Fehlermeldung!

Jinja – Templates

Es gibt nur ein Template, nämlich **test2.html**. Dieses dient zum Generierendes Rechnungen im html-Format und wird hier im Folgendem beschrieben.

Daneben gibt es noch ein Template **"SparkaseBuchung-t.html"** und **"SparkassseLastschrift-t.html"**. Diese sind aber einfacher aufgebaut und werden eventuell später beschrieben.

Nun zum Template test2.html:

* Es besteht aus einer Einleitung (Verein, Anrede, Rechnungs-Nummer, Abrechnungszeitraum
* Am Ende ist eine List von Hinweisen hinzugefügt. Diese richten sich nach dem Typ des Kunden (privat, Kleinunternehmer, pauschalierter Landwirt, normaler Unternehmer)  
  x
* Dazwischen sind zwei Tabellen auf jeder Rechnung: nämlich
  1. Die Liste der Zählpunkte mit denen Strom verkauft wird (oder auch eingespeist wird),  
     edaListGeberFlag = True
  2. Die Liste der Zählpunkte mit denen Strom gekauft wird (d.h. Strom wird bezogen,  
     edaListNehmerFlag = True

Die beiden Listen werden im Moment nicht zusammengeführt. Es kommt immer nur eine Liste vor. Derzeit gibt es keine Möglichkeit, dass ein Kunde sowohl Lieferant und Bezieher von Strom ist. Es gibt da im Moment nur die Möglichkeit, dass ein Lieferant den Strom an einen Kunden weiterschenkt (wie z.B. Reidlinger Markus schenkt Strom an seinen Vater Reidlinger Leopold)

Wir gehen die beiden Listen nun durch:

Liste mit edaListeGeberFlag = True(Stromverkäufer):

* Headerzeile
* Zählpunkte an denen Strom eingespeist wird
* Es wird hier für jeden Zählpunkt ein eigener Mitgliedsbetrag berechnet.
* rabattNehmer Liste, also Liste der Zählpunkte, welchen Strom nicht vollständig zahlen müssen,  
  in der Rechnung von Markus R. steht Leopold R. als rabattNehmer
* Abschlusszeile mit Summe

Liste mit edaListeNehmerFlag = True (Stromkäufer):

* Headerzeile
* Zählpunkte an denen Strom bezogen wird
* Auch hier wird hier für jeden Zählpunkt ein eigener Mitgliedsbetrag berechnet.
* rabattGeber Liste, also Liste der Zählpunkte, welchen Strom nicht vollständig bezahlt bekommen  
  in der Rechnung von Leopold R. steht Markus R. als rabattGeber,
* Abschlusszeile mit Summen

Grundsätzlich gibt es dann die Gesamtsumme, nämlich die Gutschriften pro Stromverkäufer und die Rechnungen je Stromkäufer. Da müsste noch die Summe erstellt werden(+/-).

Eine Zusammenführung beider Listen ist im Moment nicht vorgesehen, da es derzeit keine Rechnung gibt, bei der es beide Listen gibt!

Beschreibung des Excel-Files mit den Mitgliederlisten

Im Moment gibt es nur eine Mitgliederliste, nämlich die

**Mitgliederliste-Reidlinger.xlsx**

Diese Mitgliederlisternbestehen aus folgenden Spalten:

lfd. Nummer:

Die laufende Nummer ist immer ganzzahlig, fortlaufende  
falls Teile in Folgenzeilen beschrieben werden, dann wird eine Kommastelle angefügt (.B.: 1, 1.1, 1.2 …)

Man unterscheidet also zwischen der Hauptzeilennummer (1, ,2, 3, …) und der Folgezeilennummer (1.1, 1.2, …)

Folgezeile:

Dies wird mit speziellen Zeichen markiert (bisher nur R (für Rabett), F (für weitere Zählpunkte)

* F: zusätzliche Zählernummer
* R: Rabatt in % oder Cent
* M: Rabatt des kompletten Mitgliedsbeitrages

Mandatsreferenz:

Eindeutiges Zeichen bei eine SEPA-Lastschrift

MandatsDatum:

Datum der Unterzeichnung der Mandatsreferenz

Ust:

Umsatzstuerkennzeichnung, nur für PV-Produzenten relevant, folgende Werte sind möglich:

* P-privater Lieferant oder Kleinunternehmer,
* PL –pauschalierter Landwirt (13 % Ust. selbst an das Finanzamt zu entrichten
* U – normaler umsatzsteuerpflichtiger Unternehmer (20 % Ust, aber reverse charging)

Anrede, Name, Vorname:

klar

Adresse, Stadt, Postleitzahl:

Adresse enthält nur Strass Hausnummer und eventuell Stiege und Türnummer

email, Tal-Nr:

auch klar

Konto:

IBAN Code des Kontos

Zählpunkt:

vollständige Zählpunktnummer

Rabatt-Cent:

Rabatt in Cent pro kWh

Sinnvoller wäre es gewesen statt Rabatt-Cent einen entsprechenden Prozentsatz anzugeben,  
z.B: bei eine Rabatt von 1 Cent /kWh wären das bei 12 Cent Normalpreis eine Reduktion um 8.33 % !!, das wäre einfacher als die momentane Lösung

Rabatt:

Rabat in Prozent des Gesamtpreises (100 Prozent für Schenkung), eine Leerspalte bedeutet auch 100% Rabatt

Rabatt-Zaehlernummer:

Zählernummer, welchem der Rabatt zugut kommt

Bei Zeilentyp F (Folgezeile) wird einfach nur die zusätzliche Zählernummer bei der Rechnung berücksichtigt.

Bei Zeilentyp M und R erhält die unter Rabatt-Zählernummer angegebene Zählernummer einen Rabatt auf Kosten des angegebenen Zählpunktes.

Mitgliedsbeiträge einsammeln

Hiermit wird für alle Mitglieder der Mitgliedsbeitrag eingesammelt. Dies erfolgt einmal jährlich

Es wird grundsätzlich folgendermaßen vorgegangen:

Klasse Configurration:

* Im Konstruktor der Klassse Configuration werden die allgemeinen Datengesetzt: Höhe des Beitrages, aktuelles Jahr (für den Beitrag), Flag zum Einfügen des Beitrages
* Später soll auch ein Flag für jedes Mitglied extra gesetzt werden, im Moment gibt es nur ein generelles Flag.

Methode createPrivate:

Hier wird das Excelfile eingelesen, es wird nur die Folgezeile Typ 'M' zusätzlich berücksichtig.

Allerdings wird hier auch die MethodeaddRabatt3 aufgerufen.

Methode GenerationData.addRabatt3:

Hier werden die Werte für den Rabatt eingetragen. Es sind im Moment 15.- (kompletter Mitgliedsbeitrag), da wird nicht s genaueres gemacht.

Es wird preisBrutto und preisBruttoText gesetzt.

Festlegen des preisBrutto:

Der preisBrutto ist immer negativ, derzeit € 15.-/Jahr

Wenn der Kunde keine PV-Anlage hat, dann ist der Preis positiv, da er in Rechnung gestellt wird, der Rabatt ist daher negativ.

Wenn der Rabattgeber eine PV-Anlage hat, dann ist der Preis negativ, da eine Gutschrift verkleinert wird.

Lediglich wenn der Kunde eine PV-Anlage hat (bzw. der Rabattgeber keine hat), dann sind Änderungen notwendig. Das lassen wir im Moment offen.

Datei test2.html:

Da wird die Schnittstelle von python zu jinja und html bedient.

Hier muss man die Liste der rabattNehmer und die Liste der RabattGeber etwas anpassen:

* Nachfragen ob es sich um einen Rabatttyp 'M' handelt
* Nachfragen ob da das berechneBeitrag – Flag überhaupt gesetzt ist
* Ein wenig anpassen der Zeile

ermittleRechnungssummen2:

Bei der Liste der rabttGeber und rabattNehmer etwas anpassen,

Liste von Büchern zum Thema

Bücher zu WSL

Pro Windows Subsystem for Linux (WSL):

Powerful Tools and Practices for Cross-Platform Development and Collaboration 1st ed. Auflage

von Hayden Barnes (Author)

Learn Windows Subsystem for Linux:

A Practical Guide for Developers and IT Professionals 1st ed. Auflage

von Prateek Singh (Author)

Bücher zu Visual Studio und Python

Visual Studio Code for Python Programmers

1. Auflage

von April Speight (Author)

Netzunterlagen zu Visual Studio, WSL und Python

Ich habe das heute mir die Dokumentation einmal angeschaut:

<https://learn.microsoft.com/en-us/windows/wsl/>

<https://learn.microsoft.com/en-us/visualstudio/ide>

<https://learn.microsoft.com/en-us/visualstudio/python/>

Das gibt es ganz allgemein:

<https://learn.microsoft.com/en-us/docs/>

<https://learn.microsoft.com>

Bei <https://learn.microsoft.com> kann maneinfach pdf-Dateien machen.

Gut sind auch die zugehörigen Docs:

<https://code.visualstudio.com/docs>

Da findet man Material zu Extensions, WSL unter <https://code.visualstudio.com/docs/remote>

Auch zu Python selbst

Python packages

Das sind so normale Erweiterugen,

es gibt dann auch noch extensions , die sind aber etwas anderes

Zu python gibt es den **python package index** (<https://pypi.org/> )

Python Extensions

Sie sind unter anderem in C++ geschrieben, siehe Kapitel "Create a C++ extension for Python in Visual Studio", siehe <https://learn.microsoft.com/en-us/visualstudio/python/>

Verschiedene Projekte

Projekte aus pycharm

Directory ~/python2:

Pycharm Projekt in wsl.Projekt python2

Directoy ~/pycharm1/pythonProject

Bedeutung unklar

Projekt testPdf.py

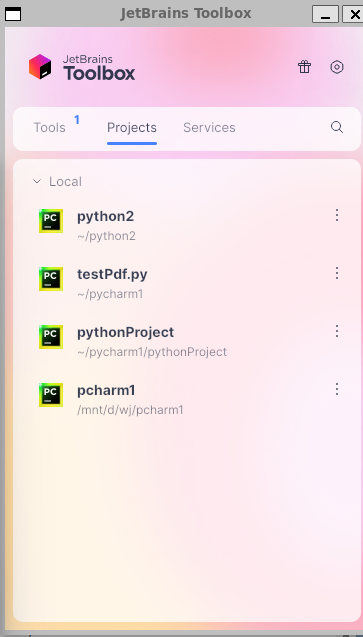
Directory ~/pycharm1

Bedeutung unklar

Bild der Toolbox von jetbrains

Man muss zuerst die Toolbox aufrufen: **jetbrains-toolbox** steht in

/home/weiser/.local/bin/jetbrains-toolbox



Folgender Aufruf hat zuletzt funktioniert

**/home/weiser/.local/share/JetBrains/Toolbox/bin/jetbrains-toolbox&**

Projekte aus Visual Studio Code

Eigentlich nur eines:

~/helloworld

Da ist alles drinnen, was man braucht! Da ist der editor in windows, Rest dann über WSL!!!

Das Projekt wird aufgerufen über die App Visual Studio Code!!!

Hinzufügen von packages

Dasi ist beschrieben in

<https://docs.python.org/3/tutorial/venv.html>

python ist relativ komplett beschrieben in

<https://docs.python.org/3>

<https://pypi.org/>

Der Python package index

Packages smtplib und email

Irgendwie gehört da IMAP und POP3 dazu

POP3 dürfte veraltet sein

Packages imaplib und poplib (veraltet!)

Aufruf von pycharm

/home/weiser/.local/share/JetBrains/Toolbox/apps/pycharm-community/bin/pycharm   
 /home/weiser/python2

Dieser Aufruf passt, ich habs wiederholt!

Wichtige packages

Email examples

<https://docs.python.org/3/library/email.examples.html>

Email Beispiele

<https://docs.python.org/3/library/email.html>

email — An email and MIME handling package

Mail utils

sudo apt install mailutils

powerShell

commands in Windows!

Details zum Senden vonMails

Die Drafts: b'(\\HasNoChildren \\Drafts) "." Drafts'

Aber anscheinend ist die Mail auch irgendwie gespeichert

Mails in der mailbox ablegen:

Mails erneut senden

Das ist zwar kompliziert, wie unten beschrieben, aber es geht nur für einzelne Mails, nicht für viele auf einmal!

* Neu empfangene Mail (in …J2024) anklicken
* Nachricht öffen
* "Aktionen" , dann "diese Nachricht erneut senden"

Parameter date\_time

Man muss ihn angeben entweder "None" oder "" oder beliebige Datum sonst gibt es Fehler

Mit date\_time=" " gabe es einen Fehler

Mit date\_time="" war es okay, aber es wurde die aktuelle Zeit eingefügt, also wie date\_time=None

Fehlermeldung beim setzen des \Draft-Flags!

imaplib.IMAP4.error: APPEND command error: BAD [b'Error in IMAP command APPEND: Invalid system flag \\DRAFTS (0.001 + 0.000 secs).']

imaplib.IMAP4.error: APPEND command error: BAD [b'Error in IMAP command APPEND: Invalid system flag \\DRAFTS (0.001 + 0.000 secs).']

Python installieren

sudo apt-get install python3.11

hier war es die Version3.11.

Interessant war aber das wsl gemeint hat 3.10 sei die aktuellste Version, aber3.12 bzw. 3.13 existiert schon, aber ich habe mit apt-get auch nur 3.11 erreicht!

Beginn der Gespräche mit Paul

Microsoft Teams:

Start mit Microsoft Teams

Das geht eigentlich ganz gut, ohne Konten , das kostet nichts!

Ich habe die email angegeben, sonst war eigentlich nichts notwendig. Vermutlich war eine Testmail (ob diese okay ist) dann dabei

**Meine email:** [**johann.weiser@aon.at**](mailto:johann.weiser@aon.at)

**Passswort (vermutlich nicht notwendig): Pauli2022**

WSL Kennung:

Die Kennung, mitwelche man anlogt ist **johann**

Das dazugehörige Passwort ist **pauli**

Mit zwei Befehlen habe ich die Datei und das dicrectory weggebracht:

johann@DESKTOP-DIJNNBL:~/wj1python$ sudo rm helloWorld.py  
[sudo] password for johann:

johann@DESKTOP-DIJNNBL:~/wj1python$ ls –l  
total 0

Dann wird das directory entfernt:

johann@DESKTOP-DIJNNBL:~$ sudo rmdir wj1python  
johann@DESKTOP-DIJNNBL:~$ ls  
wj2python

Jetzt bleibt nur mehrdas wj2python übrig

WSL Windows Subsystem Linux:

Das war auch ganz okay,

ich habe dann ein paar Details nachgelesen:

* **Die root-Kennung ist defaultmäßig nicht aktiviert**, man soll stattdessen **sudo** verwenden
* Aber ich habe zwei Dateien:  
  \wj1python und /helloWorld.py.  
  Beides haben root als Eigentümer! Das ist irgendwie komisch, ich weiß nicht wie die zu root als Eigentümer gekommen sind. Ich habe jetzt einfach das directory wj1python weggelassen11
* Ich habe jetzt nur wj2python und helloWorld.py. Die kann ich editieren, da ist alles okay!

Jetzthabe ich die zwei Dateien weggebracht (siehe oben).

Visual Studio Code

Ich kann mittlerweile auch den Code ausführen und testen (wiederum nur den Test helloWorld.py. Aber das nächste ist die letzte Version von meinem Code.

Nächste Pläne:

* Auf github oder so was ähnliches speichern – mit Paul reden.
* Ich habe da auf eine venv (heißt .venv) reduziert, nachdem ich zwischenzeitlich bis zu 5 hatte
* Dann müssen die Änderungen (so ca. seit Weihnachten nachgeholt werden) und dann sind noch ein paar offene Pläne.
* Es sieht grundsätzlich ganz okay aus, dies Geschichte mit den root-Eigentümern liegt mir ein wenig im Magen im Magen

ZoneIdentifier löschen:

Grundsätzlich geht das auch mit gpedit.msc nach Windows-R

Benztzerkonfigurateion/Administrative Vorlagen/ Windows Komponenten/ Anlagen Manager/  
Zoneninformationen in Dateianhängen nicht beibehalten

Daneben kann man auch **das Kommando find** benutzen:

find . -name "\*:Zone.Identifier" –delete (mit Vorsicht!!)