Miroslav Luňák

Automatizace měření se zaměřením na Python, přístroje Agilent, sběrnici GPIB a knihovnu VISA, v prostředí Windows XP

Obsah

Instalace softwarových součástí 3

Vytvoření jednoduchého automatizovaného měřícího celku PC-ovládací karta-měřící přístroj 4

Práce v PythonWin 4

Propojení USB-měřící karta GPIB - přístroj 5

Vytvoření programu (skriptu) pro ovládání jednoduchého automatizovaného měřícího celku s možností „odlaďování doma“ 6

Simulace měření 6

Vytvoření „exe“ souboru z „pythonovského“ programu 9

Ukládání dat do souboru a cyklus 11

Aplikace pro měření VA charakteristiky diody 13

Instalace softwarových součástí

Rozhodli jsme se sestavit ovládání přístrojů prostřednictvím sběrnice GPIB(IEEE 488) a za použití programovacího jazyka Python. Začneme stáhnutím a nainstalováním základních součástí.

Upozornění na začátek: pokud vám nepůjde některá část nainstalovat, je možné, že budete potřebovat jinou verzi konkrétní aplikace, například pro Windows Vista, nebo jiný než 32-bitový systém.

Dotazy můžete směřovat na Mgr. Miroslav Luňák, Ph.D. (lunak@dp.fce.vutbr.cz).

1. Začneme stažením instalačního souboru jazyka Python (ver.2.7.1) a nainstalujeme.
2. Stáhneme pomocnou aplikaci-prostředí pro pohodlnou editaci programů v jazyce python, PythonWin-pywin a nainstalujeme.
3. Je doporučeno (ale není nutné) stáhnout vědeckou knihovnu pro python, má název Scipy a nainstalovat. Bude se jednou hodit pro případ použití složitějších matematických funkcí (FFT).
4. Zájemci o práci s databázemi (není náplní kurzu) ocení pysqlite, tento krok však můžeme vynechat.
5. Pro komunikaci s přístroji od firmy Agilent pomůže balíček pyvisa,více se dočteme zde.
6. Kdo chce své programátorské počiny převádět do „*.exe“, může využít Py2exe, více zde.
7. Poslední, ale pro skutečně zapojená zařízení jako budou: USB-GPIB karta pro ovládání přístrojů Agilent a nebo samotné přístroje Agilent, je velmi vhodná instalace ovladačů komunikačních karet, shrnutých v trochu objemnější instalačce IOLibSuite (Input/Output knihovny) od firmy Agilent.

Nyní máme vše podstatné nainstalováno.

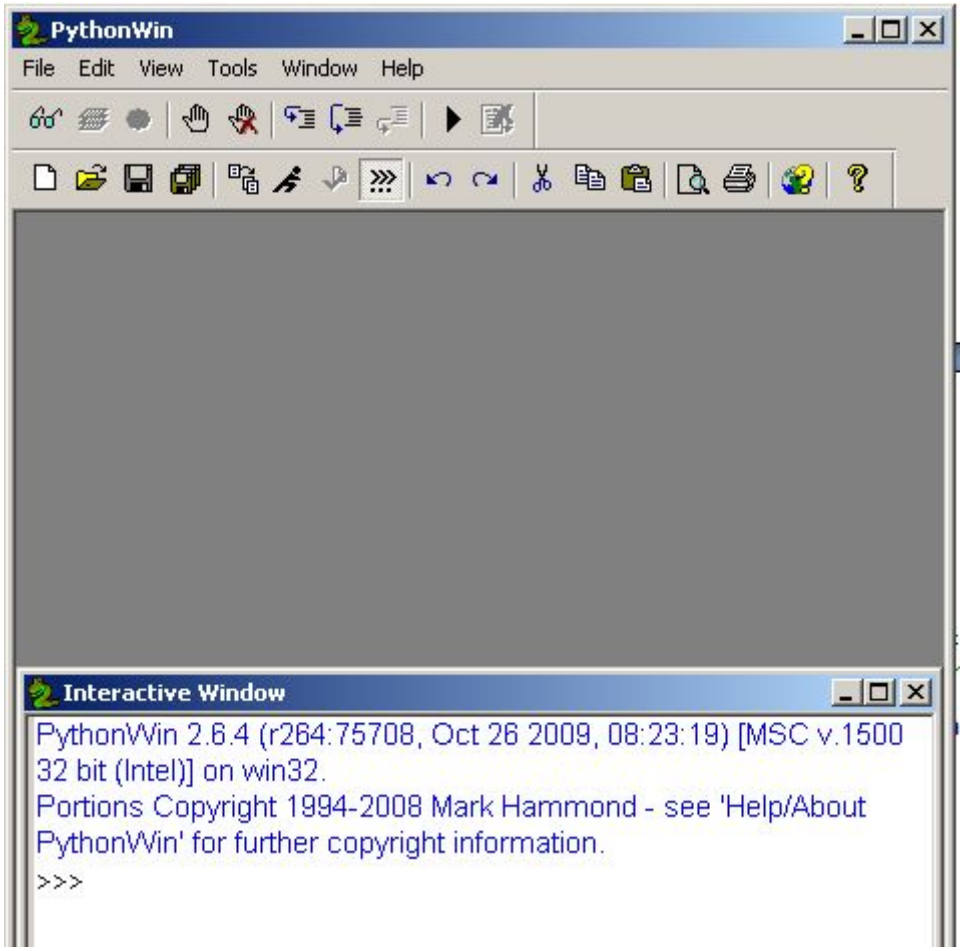
Vytvoření jednoduchého automatizovaného měřícího celku PC-ovládací karta-měřící přístroj

Předpokládá se, že máme nainstalovány součásti podle návodu na první straně. Zapojení měřící(ovládací)karty a měřícího přístroje provedeme později.

Další kroky předpokládají nejzákladnější znalosti programování v jazyku Python.

Práce v PythonWin

Spustíme aplikaci PythonWin. Objeví se okna podle obrázku 1.



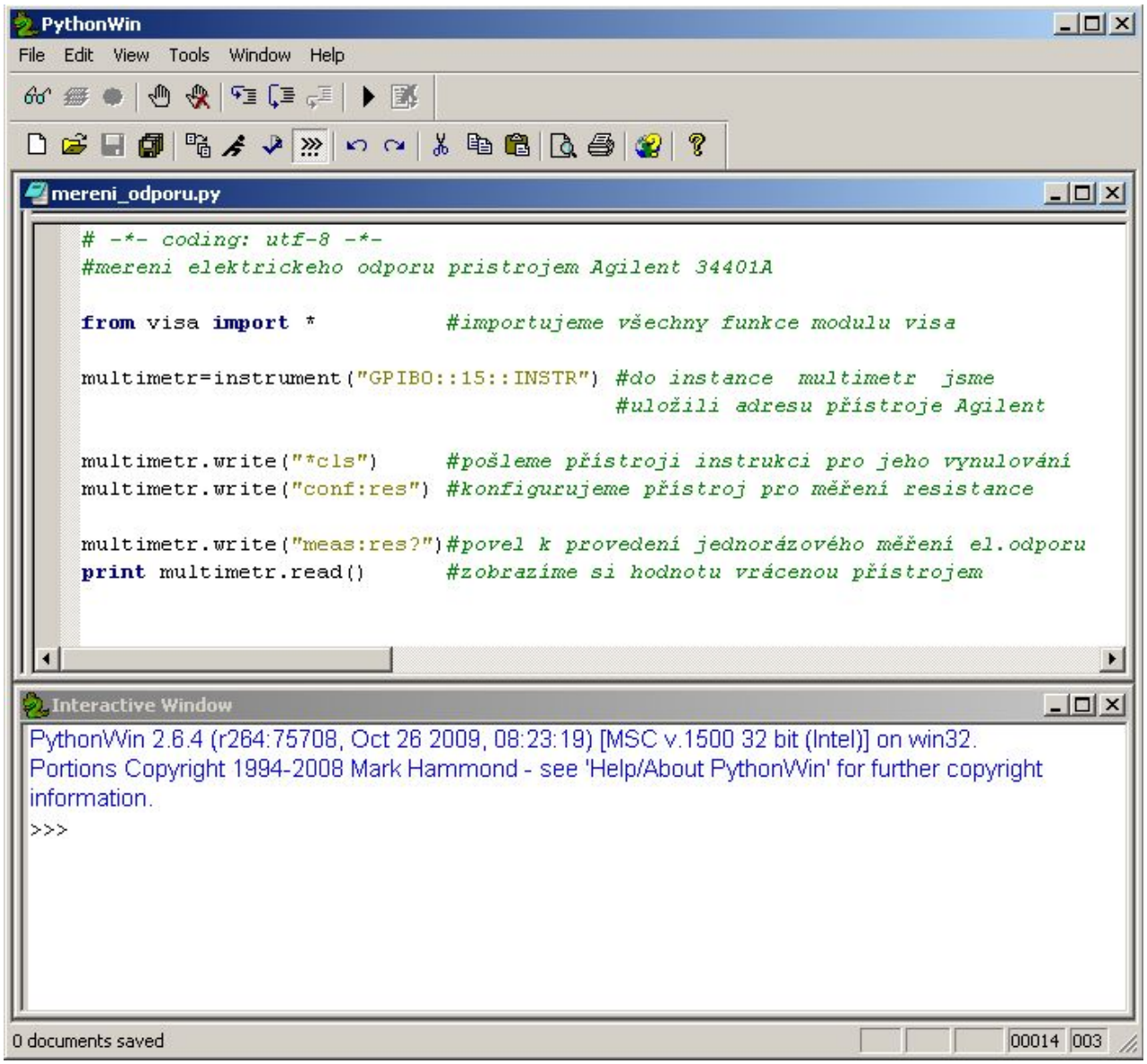
Obrázek 1 Okno PythonWin po spuštění

V Interactive Window se už dá programovat v souladu s návodem ze stránek Programujte.com. My však chceme vytvářet programy, které budeme chtít ukládat na harddisk a nebo ještě víc, šířit je jako spustitelné exe soubory u kohokoliv jiného.

Klikneme na ikonu vypadající jako bílý list (New) a dáme v dialogovém okně OK-souhlas s vytvořením nového skriptu(volba Python Script). V novém okně můžeme začít psát první řádky našeho programu, správně skriptu. Je dobré vyzkoušet si funkci různých prvků v nástrojové liště,

stačí třeba maximalizovat nové okno skriptu a hned například v menu Windows klikneme na Tile a máme okna zarovnána přehledně nad sebou.

Nyní vepíšeme do okna skriptu řádky programu, stejně jako jsou uvedeny na obrázku 2. Pozor, text za mřížkou (#) nemusíme vpisovat, na běh programu nemá vliv, jedná se pouze o textové poznámky.



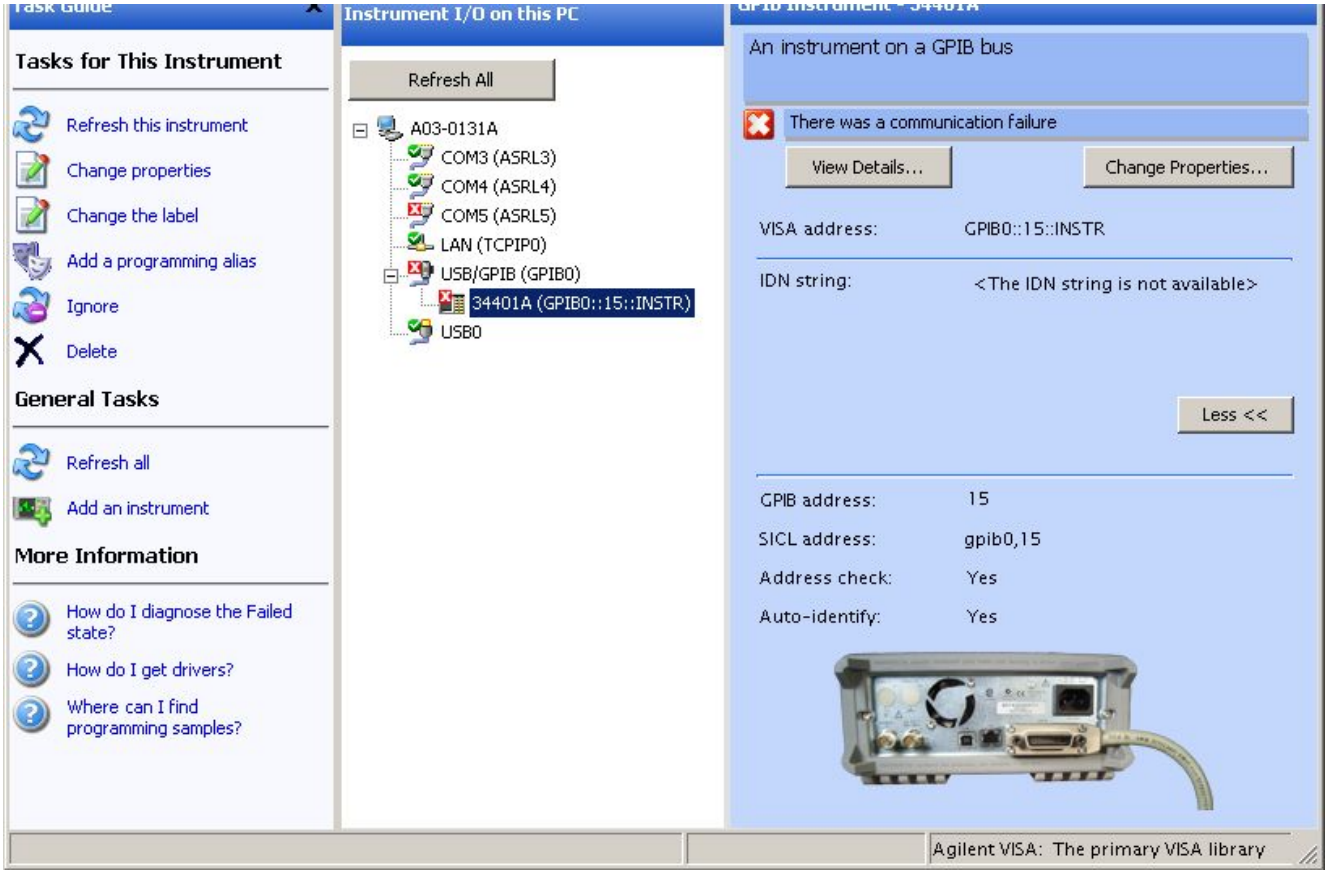
Obrázek 2 PythonWin a skript mereni_odporu.py

Uložíme náš program pod libovolným názvem, v našem případě byl použit název mereni_odporu. V adresáři pak vidíme celý název „mereni_odporu.py“.

Propojení USB-měřící karta GPIB - přístroj

Než spustíme program, musíme ještě propojit přes USB vstup měřící kartu Agilent (podle typu karty je možné i jiné zapojení) a kartu spojit přes GPIB sběrnici (rozhraní) s přístrojem Agilent

34401A. Přístroj zapneme a měřící karta musí svítit po automatické inicializaci zeleně. O správné konfiguraci přístroje a správné inicializaci se přesvědčíme v aplikaci Agilent Connection Expert. Obrázek 3 ukazuje **neinicializovaný** přístroj Agilent s adresou 15. Pro pokus o inicializaci provedeme „Refresh“ zařízení.



Obrázek 3 Agilent Connection Expert

Pokud je u přístroje ikonka („fajfka“) zelené barvy, můžeme spustit program kliknutím na ikonku černého běžce (závodníka) v nástrojové liště. V okně Interactive Window se zobrazí v řádku číselný údaj naměřené hodnoty elektrického odporu. Pokud se ozvalo pípnutí přístroje, nebo se v okně zobrazilo hlášení o chybě, je něco špatně.

Vytvoření programu (skriptu) pro ovládání jednoduchého automatizovaného měřícího celku s možností „odlaďování doma“

Simulace měření

Protože ne vždy bude možnost tvořit aplikaci za použití měřících přístrojů, nahradíme si přístroj jednoduchou softwarovou funkcí, která nám bude posílat smyšlené údaje. Provedeme několik úprav v našem programu. Použijeme skript (program) z předchozí kapitoly (obrázek 2), upravíme zdrojový kód podle obrázku 4 a uložíme pod novým názvem.

mereni_odporu_virtualne2.py

```
# -*- coding: utf-8 -*-
#měření elektrického odporu virtuálně
#from visa import instrument
from visa_simul import instrument

multimetr=instrument("GPIB0::15::INSTR")
multimetr.write("*cls")
multimetr.write("conf:res")
multimetr2=instrument("GPIB0::16::INSTR")
multimetr2.write("*cls")
multimetr2.write("conf:volt")

print multimetr.read()      #zobrazíme si hodnotu vrácenou přístrojem
print multimetr2.read()
```

Obrázek 4 Zdrojový kód "mereni_odporu_virtualne2.py"

Změny oproti předchozímu programu pečlivě zkontrolujte.

Nyní nebudeme nic zavírat a otevřeme nový skript. Nazveme jej „visa_simul.py“ a vepíšeme do něj řádky podle obrázku 5. Uložíme.

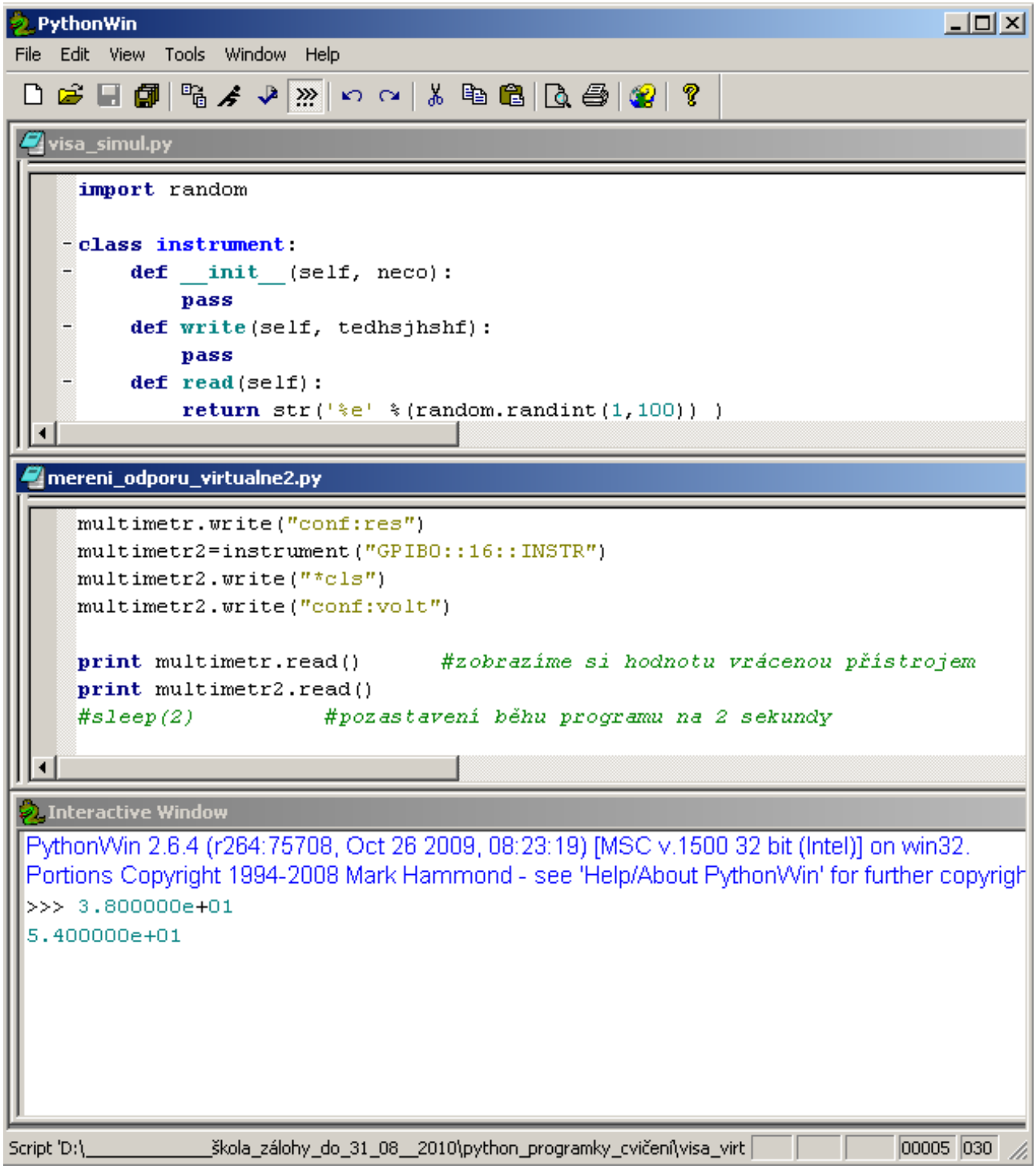
visa_simul.py

```
import random

class instrument:
    def __init__(self, neco):
        pass
    def write(self, tedhsjhshf):
        pass
    def read(self):
        return str('%e' %(random.randint(1,100)) )
```

Obrázek 5 Zdrojový kód modulu "visa_simul.py"

Po zarovnání oken bude vypadat PythonWin okno třeba jako na obrázku 6.



Obrázek 6 PythonWin – třída, skript a interaktivní okno

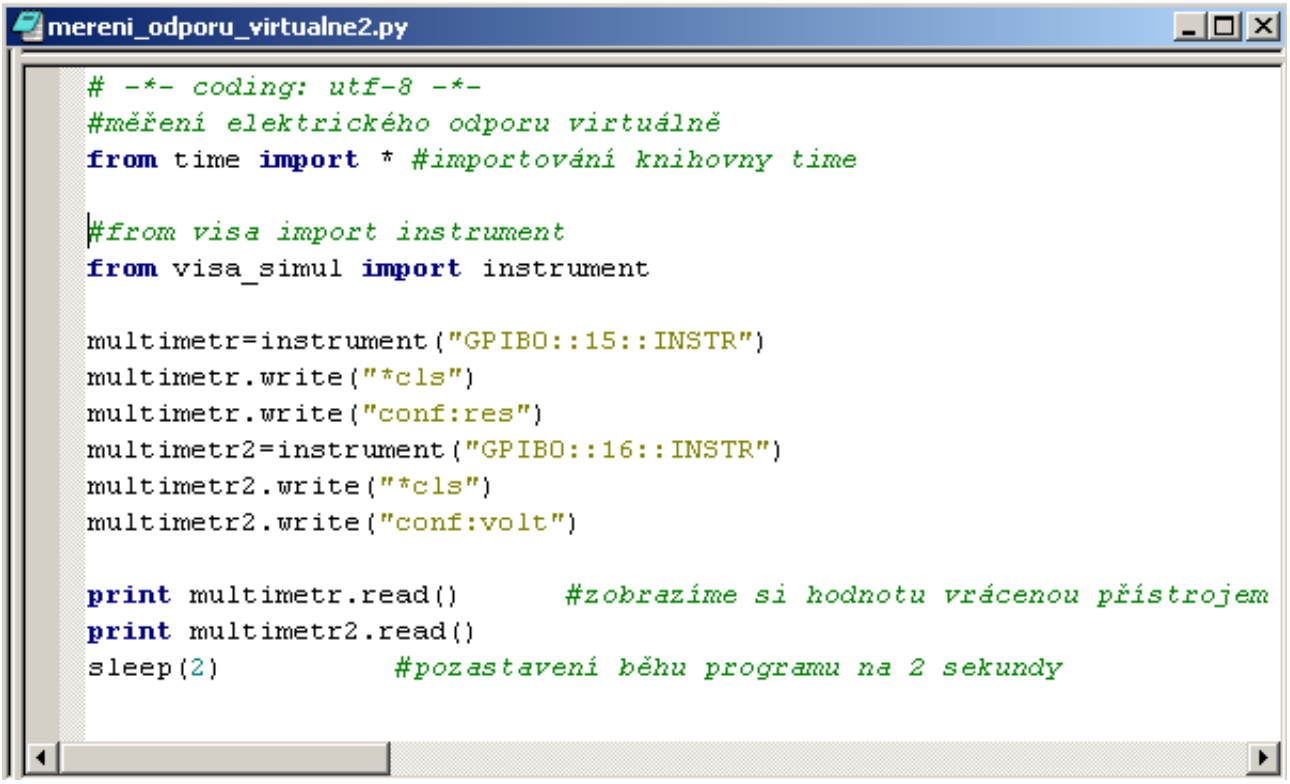
Nyní se přesuneme zpět do okna skriptu "mereni_odporu_virtualne2.py" a spustíme náš program. Pokud se program vykonal úspěšně, uvidíme v Interactive Window jako poslední záznam dvě náhodná přirozená číslo od 1 do 100. Jsme hotovi.

Navíc nám v místě uložení našeho programu přibyl soubor s názvem „visa_simul.pyc", který bude třeba v následující kapitole.

Vytvoření „exe“ souboru z „pythonovského“ programu

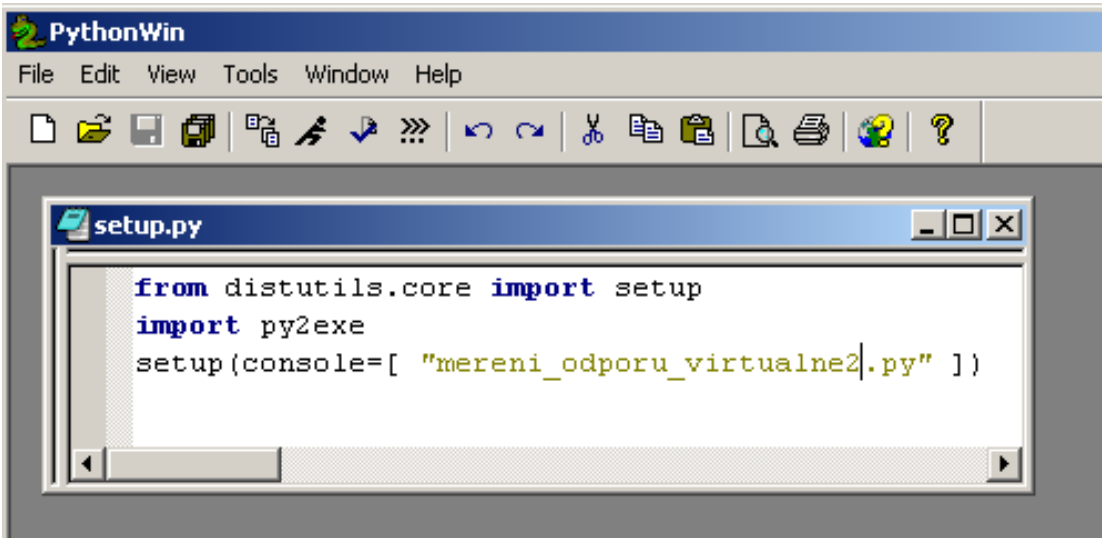
Předpokládáme, že máme funkční předešlý program s názvem „mereni_odporu_virtualne2.**py**“. Nyní si ukážeme, jak transformovat náš program na program „mereni_odporu_virtualne2.**exe**“, který si bude moct spustit i uživatel, který nemá na svém počítači nainstalovaný Python. Použijeme k tomu už nainstalované **py2exe**.

Nejprve doplníme program „mereni_odporu_virtualne2.**py**“ o dva řádky (v obrázku 7 jsou to třetí a poslední řádek), otestujeme a pokud pracuje správně, uložíme.



Obrázek 7 Doplnění hlavního souboru

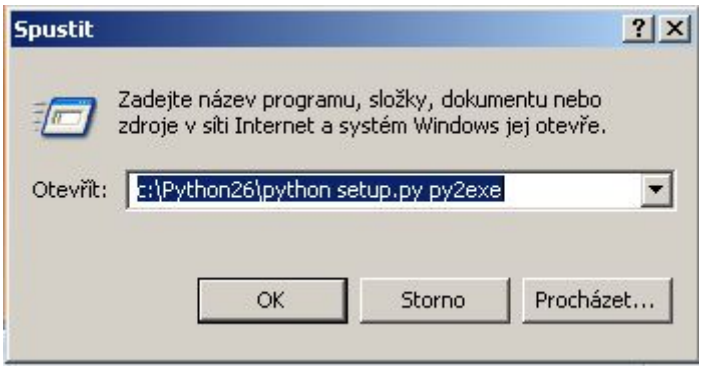
Zkopírujeme hotový program „mereni_odporu_virtulane2.py“ do adresáře Python26 na disku C. (C:\Python26\). Na stejné místo zkopírujeme soubor „visa_simul.pyc“. V prostředí PythonWin zavřeme okna skriptů a otevřeme zcela nový skript, do kterého vepíšeme řádky podle obrázku 8.



Obrázek 8 Obsah setup.py souboru

Skript uložíme pod názvem „setup.py“ také do C:\Python26\. Je důležité správně vyplnit název vašeho souboru uvnitř uvozovek. PythonWin můžeme zavřít.

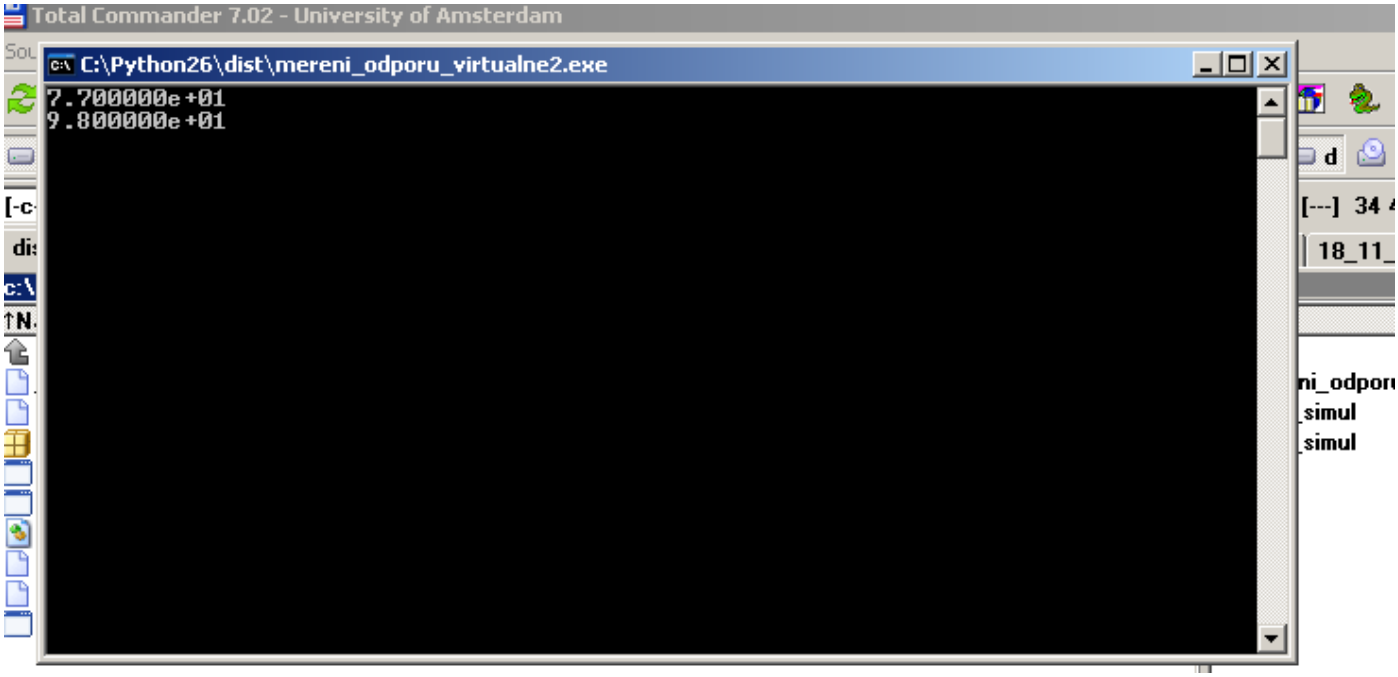
V prostředí Microsoft Windows klikneme na Start-spustit a do řádku vepíšeme text „c:\Python26\python setup.py py2exe“. Viz obrázek 9.



Obrázek 9 Spuštění tvorby exe souboru

Stlačíme klávesu Enter nebo klikneme na OK. Pokud je všechno v pořádku, začne se nám v černém konzolovém okně velmi rychle vypisovat různý text, informující o probíhající tvorbě „exe“ souboru. Po skončení uvidíme v adresáři Python26 dvě nové složky „build“ a „dist“. Uvnitř složky „dist“ nalezneme vygenerovány různé soubory, mezi kterými je očekávaný soubor „**mereni_odporu_virtualne2.exe“**. Tento soubor zkopírujeme do libovolné prázdné složky, abychom ověřili, zda samostatný exe soubor funguje. Po spuštění exe souboru v samostatné složce zaregistrujeme chvilkové probliknutí nějakého černého okna-program nepracuje, nebo nepracuje správně. Zkopírujeme ze složky „dist“ ještě další soubor s názvem „**python26.dll**“ do složky, která dosud obsahovala pouze nový exe soubor. Pokusíme se znovu spustit exe soubor. Opět nepracuje. Zkopírujeme další podpůrný soubor a to „**library.zip**“. Znovu se pokusíme o spuštění. Vidíme, že černé konzolové okno se zviditelnilo na delší dobu, stačíme přečíst údaje -výsledek chodu programu a po 2 sekundách se okno bez našeho souhlasu zavřelo. Program funguje!

V případě, že se opět nepodařilo spustit program, zkopírujeme do adresáře další podpůrné soubory.



Obrázek 10 Konzolové okno

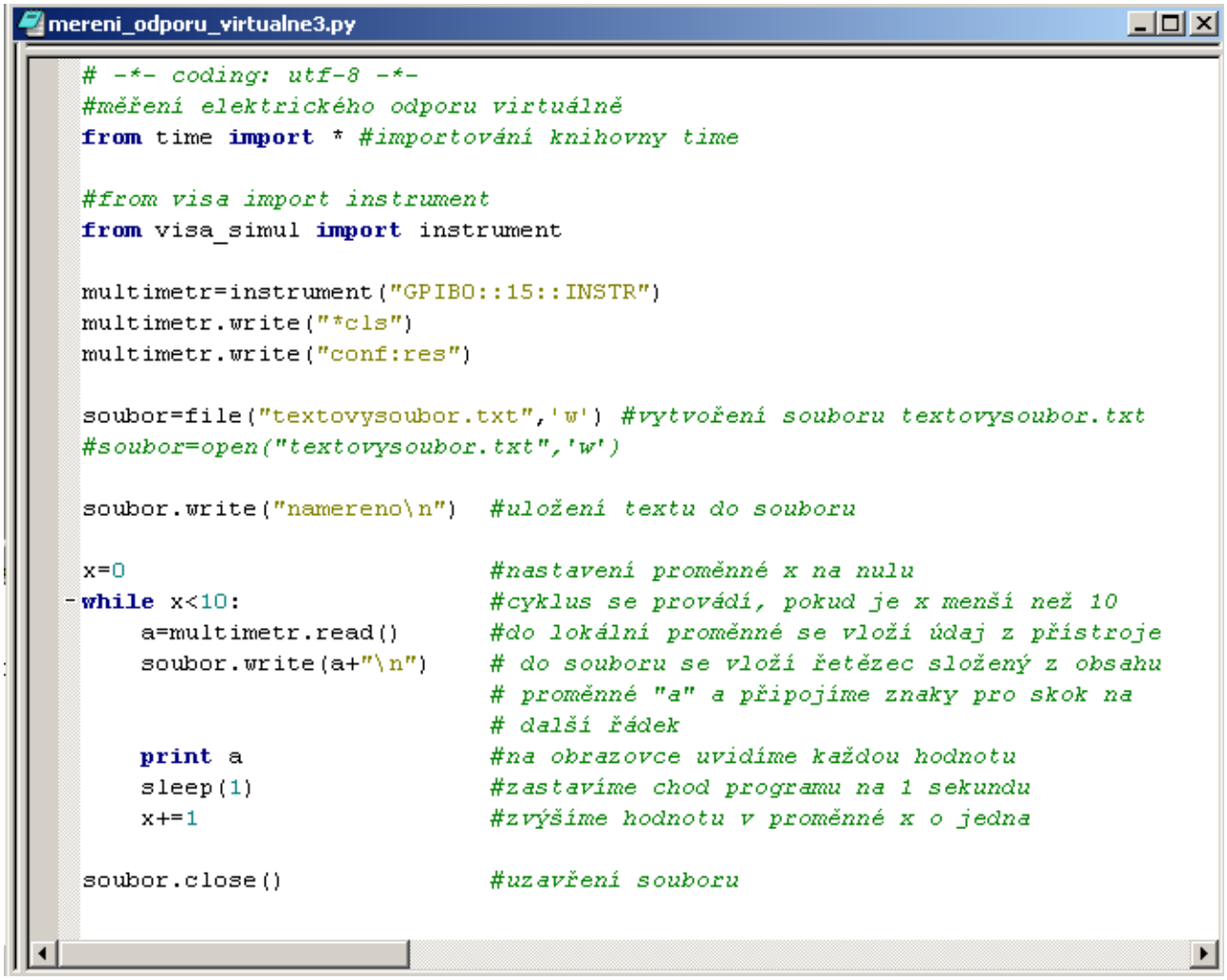
Text předchozího odstavce měl za cíl ukázat, že existují podpůrné soubory, nezbytné k chodu naší aplikace (exe). Množství takových souborů a jejich celková velikost se odvíjí od používaných knihoven. Samozřejmě by bylo nejsnazší vzít všechny nově vygenerovaná soubory a ty šířit spolu s novým exe souborem. Některé jiné programátorské nástroje (jiné jazyky a prostředí) mají propracovanější funkce tvorby výsledného exe souboru, který by v sobě už obsahoval všechny používané knihovny. Není vyloučeno, že lze na internetu nalézt jinou formu „setup.py“ skriptu, který umožní vytvořit soběstačný exe soubor.(http://www.py2exe.org/). Pro seznámení s metodami automatizace za použití moderních programovacích jazyků však uvedený postup stačí.

(Poznámka pro uživatele Total Commander: spuštění tvorby exe souboru lze provézt i v příkazovém řádku v adresáři C:\Python26\ napsáním „python setup.py py2exe“.)

Ukládání dat do souboru a cyklus

Už umíme vytvořit samostatný funkční program, nyní jej obohatíme o ukládání naměřených dat do textového souboru. A abychom mohli uložit efektivně víc než jednu sadu dat, seznámíme se s cyklem, přesněji jedním z možných cyklů.

Otevřeme nový skript nebo upravíme některý dřívější podle programu na obrázku 11. Samozřejmostí by mělo být uložení nového programu pod jiným názvem, než byl předchozí.



Obrázek 11 Program zahrnující ukládání do souboru a cyklus

Na 9. Řádku programu (prázdné řádky nezapočítány) se nachází syntaxe *soubor=file("textovysoubor.txt",'w')*, která vytváří soubor s příponou txt a do kterého se bude pouze zapisovat, proto je vedle názvu souboru uveden příznak w. Tento řádek lze nahradit použitím pouze syntaxe *soubor=open("textovysoubor.txt",'w')*, která otevírá soubor s daným názvem pro zápis do souboru. V případě, že soubor neexistuje, vytvoří se nový soubor. Po otevření souboru se začne zapisovat na začátek souboru. V našem případě se jako první zapíše slovo *namereno\n*, jenže poslední dva znaky nikdy v souboru běžně neuvidíme. Jedná se o formátovací znaky, které zajistí, aby se příští zapsané slovo (číslo) umístilo na nový řádek souboru.

Může se vám stát, že budete chtít otevřít soubor s daty, aniž byste chtěli přepisovat už uložená data. Zkrátka potřebujete pouze přidávat na konec souboru další a další údaje. K tomu slouží atribut append, který uvedete v podobě „a“ místo „w“ v řádku: *soubor=open("textovysoubor.txt",**'a'**)*

Ještě nutná poznámka k práci se soubory: vždy po skončení používání souboru pro čtení nebo zápis je nezbytné zavřít soubor. To je provedeno posledním řádkem našeho programu na obrázku 11.

Nyní se zaměříme na blok, uvozený syntaxí *while x<10:* . Než jsem k tomuto bloku přišli, nadefinovali jsme si proměnnou x a vložili do ní hodnotu „nula“. Pomocí syntaxe *while x<10:*

říkáme programu, aby obsah bloku prováděl opakovaně tak dlouho, dokud bude hodnota proměnné menší než 10. Co provádí další řádky bloku-cyklu, je popsáno v obrázku. Pro lepší pochopení je zde také výpis hodnot od multimetru na obrazovku (print) a zpomalení chodu programu (sleep) aby bylo patrné opakované provádění bloku. Na konci bloku je provedena inkrementace (navýšení hodnoty) proměnné x o „jedničku“. Kdybychom tento řádek vynechali, cyklus by se prováděl donekonečna a my bychom museli přerušit chod programu (*vpravo dole v oznamovací oblasti systému Windows rozevřít případnou roletku a na ikoně aplikace PythonWin kliknout pravou myší a vybrat možnost Break into runnig code.*)

Existují ještě další příkazy typu cyklus (např *for i in range(10):*) a podmínkové příkazy (*if podmínka1:*), kterým souhrnně říkáme příkazy řídící běh programu. Těm se budeme věnovat později.

Celý výsledek úspěšně spuštěného programu je pak uložen v souboru s názvem „textovysoubor.txt“. Název si může zvolit každý sám. Do budoucna je dobré nepoužívat pro názvy souborů mezery ani znaky diakritiky, vyhnete se pak komplikacím s načítáním jména souboru.

Aplikace pro měření VA charakteristiky diody

Ukázali jsme si, jak lze ovládat přístroje agilent za pomocí knihovny visa a jak provést libovolné množství operací zadáním několika příkazových řádků. V této kapitole si ukážeme a krátce vysvětlíme kód (jednoduchý příklad) aplikace, která bude ovládat zdroj elektrického napětí Agilent E3631 a 2 multimetry agilent 34401A nebo 34410A, a která naměřená data uloží do souboru.

Praktické zapojení přístrojů a polovodičové diody je triviální. Na kladnou svorku výstupu zdroje napětí s rozsahem 25V připojíme kladný vstup ampermetru, záporný napojíme na diodu (kladný pól). Druhý vývod diody přivedeme na COM svorku zdroje napětí. Na diodu připojíme paralelně voltmetr.

Při programování vystačíme s virtuální knihovnou.

Otevřeme nový nebo použitý skript a vepíšeme kód podle obrázku 12. Části kódu jsou detailně popsány, přibyly některé instrukce pro nastavení zdroje napětí, které lze spolu s dalšími nalézt v manuálu přístroje.

Zvláštní pozornost věnujme z programátorského hlediska novému řádku, který obsahuje syntaxi *zdroj.write("volt "+str(Uz)+"\n")*. V závorce jsou zde uvedeny tři řetězce, první obsahuje povel pro nastavení napětí zdroje, poslední obsahuje ukončovací znaky řetězce, posílaného přístroji přes knihovnu (třídu) visa. Prostřední zápis provádí převod hodnoty proměnné Uz (napětí zdroje, které požadujeme nastavit) na string-česky řetězec. Všechny řetězce pak laicky řečeno sečteme, spojíme do jednoho řetězce. Zde se ukazuje další výhoda jazyka python při práci s řetězci. V ostatních programovacích jazycích je práce s řetězci komplikovanější.

Podobně se zachováme i v řádku se syntaxí *soubor.write(U+" "+I+"\n")*, kde stejným způsobem spojíme naměřená data do jednoho většího slova, určeného pro zápis do souboru i s oddělovačem dvou hodnot, kterým je „mezera“. Všimněme si, že v tomto případě jsme nemuseli převádět hodnoty číselných proměnných na řetězce. Vysvětlení tohoto rozdílu není v tuto chvíli podstatné.

mereni_VA_diody.py

```
# -*- coding: utf-8 -*-
from time import *
from visa_simul import instrument
#from visa import instrument
zdroj=instrument("gpib0::8::instr")
zdroj.write("*cls")        #vymazání registrů přístroje
zdroj.write("inst p25v")   #výběr druhu výstupu zdroje
zdroj.write("volt 0")      #nastavení nulového napětí zdroje
zdroj.write("outp on")     #otevření výstupu zdroje
ampermetr=instrument("GPIB0::15::instr")
ampermetr.write("*cls")
ampermetr.write("conf:curr:DC") #konfigurace multimetru 1
voltmetr=instrument("GPIB0::7::instr")
voltmetr.write("*cls")
voltmetr.write("conf:volt:DC")  #konfigurace multimetru 2
soubor=open("VA_char.txt",'w')  #otevření souboru pro zápis
soubor.write("namereno\n")      #vložení textu do souboru
x=0                             #deklarace proměnné x
Uz=0                            #deklarace proměnné napětí zdroje
while x<(10):
    Uz+=1                       #přičtení jedničky do proměnné Uz
    zdroj.write("volt "+str(Uz)+"\n")#zaslání řetězce zdroji
    sleep(1)                    #počkáme až zdroj nastaví napětí
    voltmetr.write("read?")     #pokyn změř napětí na diodě
    U=voltmetr.read()           #vložení změřené hodnoty do proměnné
    ampermetr.write("read?")    #pokyn změř proud diodou
    I=ampermetr.read()          #vložení hodnoty proudu do proměnné
    soubor.write(U+" "+I+"\n")  #zápis napětí a proudu do souboru
    print U," ",I               #tisk napětí a proudu na obrazovku
    sleep(1)                    #zpomalení běhu programu
    x+=1                        #inkrementace proměnné x
zdroj.write("outp off")         #odpojení výstupu zdroje od obvodu
soubor.close()                  #uzavření souboru dat
```

Obrázek 12 Aplikace mereni_VA_diody.py

Druhým závažným případem v našem programu je zpomalení chodu běhu programu po odeslání instrukce „nastav napětí zdroje“ *sleep(1)* *#počkáme až zdroj nastaví napětí* – v reálných situacích není zdroj schopen nastavit napětí okamžitě, proto musíme před zahájením měření vyčkat několik sekund a až pak provádět měření. V opačném případě bychom naměřili nesmyslné hodnoty měřené veličiny. Prakticky zatím nekomunikujeme se zdrojem, zda už nastavil napětí, pouze chvíli přerušíme vykonávání programu.

Třetím závažným povelem je „outp off“ na konci programu. Odpojíme výstupní obvod zdroje od elektrického obvodu. Je to z ohledu na bezpečnost obsluhy. Pozor, některé elektrotechnické součástky, zejména tenkovrstvé polovodičové součástky jsou choulostivé na prudké změny

elektrického potenciálu, které by nastaly odpojením od zdroje, také přechodové změny při odpojování elektrické součástky mohou citlivé lasery nebo čipy zničit.

Za zmínku stojí také způsob výpisu naměřených dat na obrazovku. *print U," ",I #tisk napětí a proudu na obrazovku* –zápis je velmi jednoduchý oproti jiným programovacím jazykům.

Na závěr nesmíme zapomenout na uzavření souboru před skončením vykonávání programu.

Nyní máme dostatečný základ pro vytváření aplikací pro automatizované měřící celky. Další nástavbou by byly aplikace okýnkového typu (použití TKinter, Gtk), či vykreslování grafů v okně aplikace (použití Matplotlib).