Moji začetki s Pythonom (Dejan Križaj, 2013)

Uporabo programskega jezika Python mi je svetoval kolega, ki dela v podjetju in s pomočjo njega oblikuje tako samostojne kot spletne aplikacije, ga uporabi za matematične operacije, grafične prikaze in podobno. S tem zapisom vas bom poskušal uvesti v prve korake uporabe Pythona v elektrotehniki, bolj specifično pri predmetu Osnove elektrotehnike. (Moram pa priznati, da sem tudi sam bolj začetnik pri uporabi tega jezika.)

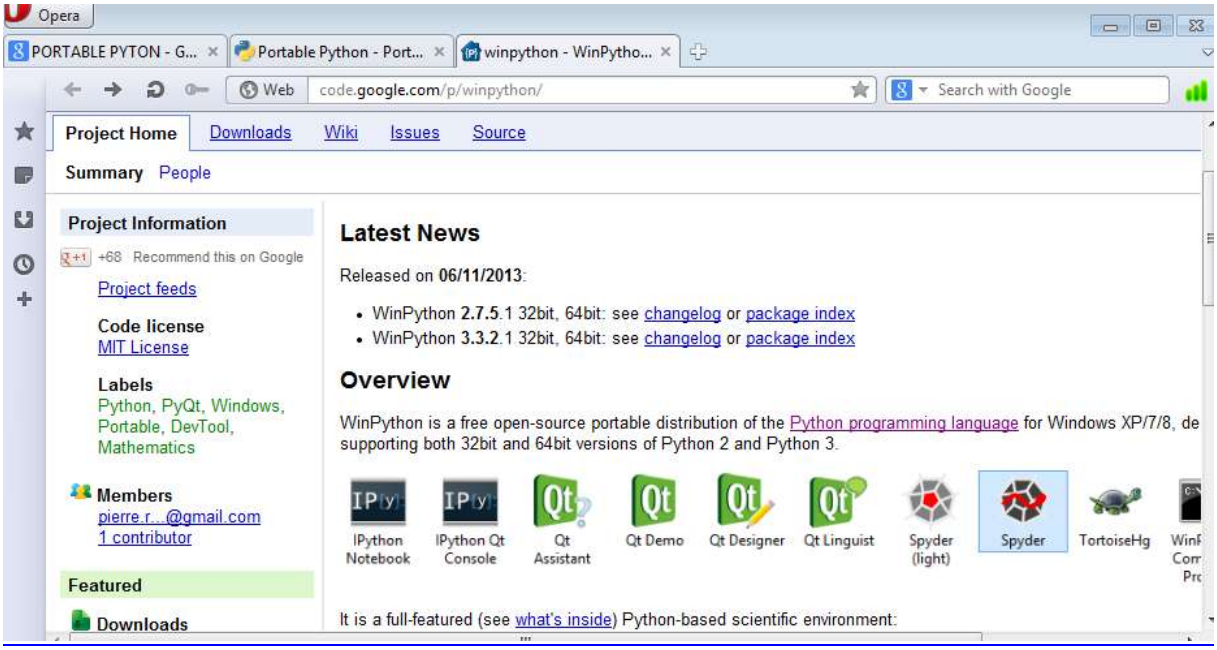
Python nas bo zanimal predvsem kot orodje za osnovne izračune in izrise funkcij, seveda pa je to le ena od mnogih možnosti njegove uporabe. Podobno kot Matlab je Python skriptni jezik, torej se izvaja vrstica za vrstico in ukaze lahko izvajate direktno iz ukazne vrstice. Python torej svetujem kot alternativo Matlabu, Octave ali drugim programskim orodjem. Ima vrsto prednosti pred drugimi programskimi jeziki, saj je njegova zasnova moderna, izvajamo ga lahko na zelo raznovrstnih platformah in omogoča integracijo vrste knjižnic, ki omogočajo delo s perifernimi enotami, matematičnimi operacijami in podobno. Poleg tega je zastonj, kar sicer v času študija ne izgleda tako pomembno (ker mnogi poznajo metode za brezplačno uporabo sicer plačljivih programov), postane pa pomembno, če želite narediti svojo aplikacijo, ki bo po možnosti tudi komercialno zanimiva. Da je Python uporabno programsko orodje se lahko prepričate tudi sami. Za začetek morda tako, da pogledate, kaj o tem pravijo na spletu:

http://pythoncard.sourceforge.net/what_is_python.html
http://en.wikipedia.org/wiki/Python_(programming_language)
http://mikelev.in/2011/01/python-programming-language-advantages/#
http://www.python.org/doc/essays/comparisons.html

Obstaja vrsta dobrih navodil kako začeti s Pythonom, kljub temu pa bom v tem zapisu opisal nekaj osnovnih korakov, ki vas hitro uvedejo v uporabo Pythona, še posebno v kontekstu njegove uporabe pri predmetu Osnove elektrotehnike. Na koncu bom navedel nekaj virov, ki jih velja uporabiti pri njegovi uporabi. Poleg tega bom ta zapis oblikoval tako, da vas bo tekst vodil tako, da vam bo predlagal nekaj lastnega eksperimentiranja po vsakem koraku in vas na ta način aktivno vodil do osnovne uporabe Pythona.

1 Korak: zagon in osnovne matematične operacije

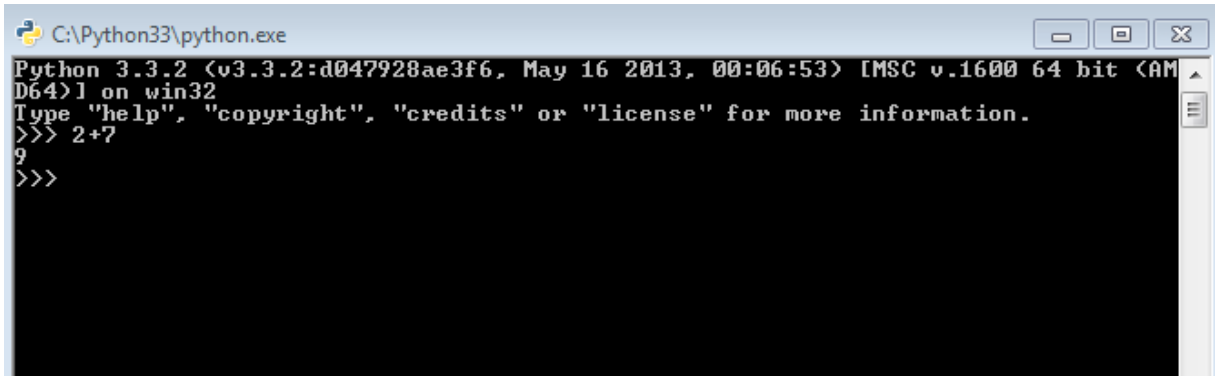
Naložite si eno od verzij Pythona, npr. iz http://code.google.com/p/winpython/



Dobra varianta se zdi inštalacija Pythona s pomočjo programa Anaconda. Prednost tega načina je, da se hkrati naložijo še vse potrebne knjižnice (moduli). Ta način priporočam tistim, ki so sicer manj vešči vzpostavitve programja: http://docs.continuum.io/anaconda/install.html,

Pogosto se poleg Pythona uporablja za izdelavo grafične podobe samostojnih aplikacij še program QT, ki je tudi dostopen na zgornji spletni strani. Obstaja verzija 2.* in 3.* Pythona. Obe sta aktualni, le da se v verziji 3 poskuša odpraviti nekaj nedoslednosti verzije 2. (Vsaj tako sem jaz razumel).

Program lahko zaženemo direktno s klikom na dadoteko *python.exe*. Opazimo »staro« DOS okno z



zapisom verzije Pythona in simbolom >>> za ukazno vrstico. Podobno kot v Matlabu.

Napišemo npr. 2+7 in program izpiše rezultat: 9. Sedaj lahko preizkusimo nekaj osnovnih operacij.

NAREDI SAM 1: zaženi inštaliran program in preiskusi nekaj matematičnih operacij. Pri tem uporabi matematične opracije iz spletne strani Wiki:
http://en.wikibooks.org/wiki/Python_Programming/Basic_Math

Zaplete se, ko želimo zapisati nekoliko bolj kompleksno enačbo, ki npr. vsebuje sinusno funkcijo. Če kot primer navedemo kar uporabo kosinusne funkcije in poskušamo izračunati cos(30°), kar bi v Matlabu dalo rezultat 0,5, bi v Pythonu dobili napako. Ki pa jo rešimo tako, da naložimo modul z matematičnimi funkcijami npr. z ukazom *import math*. Potem sinusno (in druge v modulu opisane funkcije) funkcije kličemo v obliki math.sin(pi/3). Žal tudi v tem primeru program javi napako, saj je funkcija pi definirana znotraj matematičnega modula. Zato je pravilen zapis *math.sin(math.pi/3)*. Sedaj bo rezultat enak 0,5. (Da se izognemo dolgemu pisanju lahko pri importiranju modula napišemo npr. import math as m in potem uporabimo ukaze m.sin ali m.pi. Alternativno bi lahko uporabili tudi ukaz *from math import **, kar bi omogočilo uporabo direktnih ukazov *sin* in *pi* vendar ta način ni priporočljiv, ker lahko pride do konfliktov pri definiranih imenih v različnih modulih)

Here are some commonly used mathematical operators

Syntax	Math	Operation Name
a+b	$a + b$	addition
a-b	$a - b$	subtraction
a*b	$a \times b$	multiplication
a/b	$a \div b$	division (see note below)
a//b	$a \div b$	floor division (e.g. 5//2=2) - Available in Python 2.2 and later
a%b	$a \bmod b$	modulo
-a	$-a$	negation
abs(a)	$\|a\|$	absolute value
a**b	a^b	exponent
math.sqrt(a)	$\sqrt{a}$	square root

NAREDI SAM 2: Določi npr. vrednosti izrazov $\sin(30^o)$, $\arctan(1)$, e^{-2}, $\ln(2)$, 2^{-3} **, pri čemer si pomagaj s pomočjo na spletu, na strani http://docs.python.org/2/library/math.html. Če si delal prav si moral dobiti (zaokroženo na dve decimalki): 0.87, 0.97,0.14, 0.96 in 0.125. Če želiš dobiti rezultat v stopinjah, je potrebno rezultat deliti s pi in pomnožiti s 180! Namesto tega lahko uporabite tudi ukaz math.degrees . Poskusite sami!**

POMNI: Običajno lahko v ukaznem oknu »prikličemo« prejšnje ukaze s pritiskom na puščico gor (↑), kar pohitri popravljanje napačno zapisanih ukazov.

NAREDI SAM 3: V določenem obdobju bomo pri Osnovah elektrotehnike (OE2, izmenični signali) potrebovali kompleksni račun. Računanje s kompleksnimi števili je v Pythonu enostavno, saj kreiramo imaginarno število preprosto tako, da dodamo črko j na koncu števila (npr. 2j). Kot primer pomnožimo 2j s samim seboj. Dobiti moramo -4, saj velja $j^2 = -1$ **. Poskusite zapisati nekaj računov s kompleksnimi števili.**

2. korak: Izvajanje skript

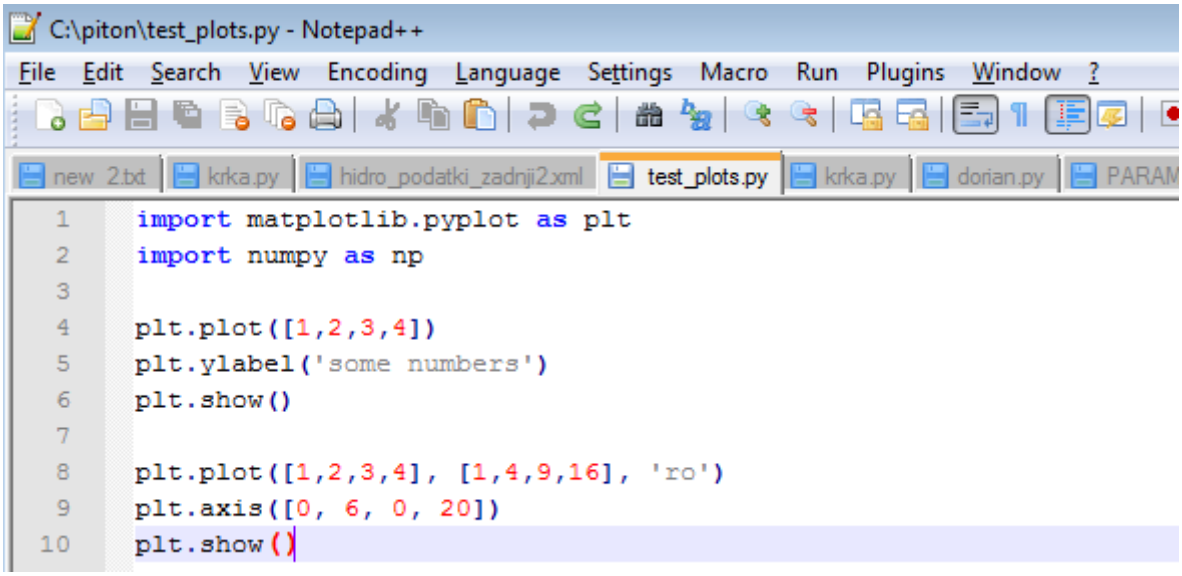
Ukaze redko vpisujemo direktno v osnovno okno pač pa tvorimo dadoteke z nizom ukazov, ki jih nato zaženemo hkrati. Podobno, kot to pri Matlabu naredimo v .m dadoteki , imajo dadoteke v Pythonu končnico .py in jih lahko urejamo s poljubnim programom za urejanje besedil. Vse ukaze torej zapišemo v dadoteko in ji damo končnico py (npr. mojskript.py) ter jo izvedemo z ukazom *python mojskript.py.* Če niste predhodno nastavili, da se dadoteke izvajajo iz ustrezne mape, je potrebno vpisati celotno pot do dadoteke, npr. *python C:\WinPython-64bit-2.7.5.0\mojskript.py*.

NAREDI SAM 4: Zapiši nekaj ukazov v dadoteko in jih izvedi.

Sam uporabljam urejevalnik besedil Notepad++, ki tudi vsebuje določena orodja, ki omogočajo pravilno urejanje ukazov in zagon dadotek direktno iz urejevalnika. Če želite ubrati enako pot, si naložite program Notepad++ (brezplačen) iz interneta in ga ustrezno konfigurirajte. Nastavite »jezik« v zavihku Language> P>Python. Nato izberite možnost, da izvajate skripte direktno iz programa. Tu lahko nastopi nekaj težav s pravilno nastavitvijo, saj se običajno program zažene in izgine. V tem primeru uporabite enega od nasvetov iz interneta; npr. da v Run ukazu dodate *C:\potdoukaznevrstice\python.exe "$(FULL_CURRENT_PATH)"*.
Če ste Pitona inštalirali v drugo mapo, ustrezno spremenite ukazno vrstico. Mimogrede, program Notepad++ vam bo prišel prav še v drugih primerih, morda pri delu s html dadotekami, pri drugih programskih jezikih, hitrem pregledovanju tekstovnih dadotek, hitrem iskanju in spreminjanju besed v eni ali več dadotekah itd.

NAREDI SAM 4: Inštaliraj Notepad++, nastavi jezik (Language) ter ukaz Run, vpiši nekaj vrstic kode in jih zaženi z ukazom Run.

POMNI: Predhodno moraš dadoteko shraniti (hiter ukaz je Ctrl S).



3. Korak: izračun funkcij in izris

Naslednji korak je izračun in izris funkcij. To lahko naredimo brez posebnega definiranja funkcij, preprosto tako, da tvorimo niz neodvisne spremenljivke in izračunamo vrednosti funkcije za te vrednosti. Npr., želimo izračunati tok kot funkcijo časa, ki jo v matematični obliki zapišemo v obliki $i(t) = Ae^{-t/tau}\cos(\omega t)$, kjer so A, tau in ω konstante, t pa je čas. Gre za časovno upadanje sicer harmoničnega toka, kar bi dobili npr. pri izklopu RLC vezja od napajanja.
Najprej tvorimo niz vrednosti časovnih trenutkov, v katerih bomo izračunali vrednosti toka, npr. od 0 do 5 s korakom po 0,01. Na ta način bomo dobili veliko število diskretnih časovnih trenutkov, za katere bomo v nadaljevanju izračunali vrednosti toka. V tem trenutku je pametno naložiti modul numpy (***import numpy as np***), ki omogoča vrsto manipulacij z nizi, na podoben način kot to dela Matlab. (Več o tem modulu npr. na strani http://physics.nmt.edu/~raymond/software/python_notes/paper003.html), lahko pa malo pobrskate tudi za drugimi razlagami in primeri (npr. http://nbviewer.ipython.org/urls/raw.github.com/jrjohansson/scientific-python-lectures/master/Lecture-4-Matplotlib.ipynb). Sedaj lahko tvorimo potreben niz z ukazom arange(začetek, konec, korak). Če npr. napišemo b = np.arange(0,11), bomo kot rezultat tvorili spremenljivko b, v kateri bo niz vrednosti od nič do 10:

Primer:

```
>>> b = np.arange(0,11)                <- tvorimo niz in ga shranimo v spremenljivko b
>>> b                                  <- izpišemo vrednosti spremenljivke
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])   <- rezultat izpisa
```

V našem primeru uporabimo ukaz
t=np.arange(0,5,0.01).
Alternativno lahko tvorimo niz z ukazom linspace(začetek, konec, število korakov).

Nato določimo parametre A, tau in omega, npr: A=10, tau=2, omega=20.
Nato tvorimo vrednosti toka za določene vrednosti časa, preprosto kot
i=A*np.exp(-t/tau)*np.cos(omega*t)

Sedaj bomo preskočili mnogo zadev v zvezi z nizi (kar trenutno ne potrebujemo) in izrisali tok kot funkcijo časa. V ta namen uporabimo modul **pyplot**, kije vključen v modul **matplotlib**, ki ga vključimo z ukazom *import matplotlib.pyplot as plt*. Sedaj lahko uporabimo vrsto zelo učinkovitih ukazov za grafične izrise, ki jih omogoča knjižnica. Več o tem npr. na strani http://matplotlib.org/ ali pa na strani https://klassenresearch.orbs.com/Plotting+with+Python .

Tok kot funkcijo časa preprosto izrišemo z ukazom
plt.plot(t,i), pri čemer moramo za izris na ekran dodati še ukaz
plt.show()

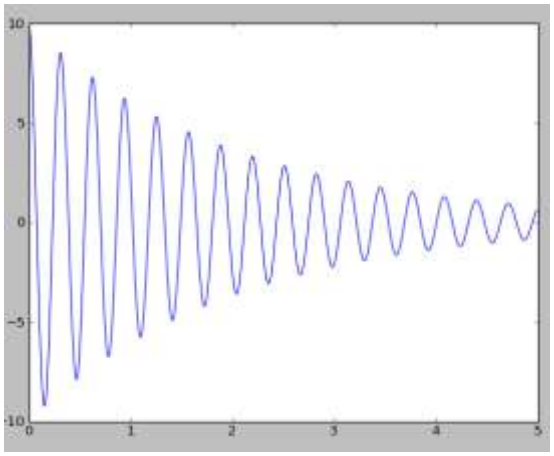
Primer (če povzamemo vse potrebne ukaze oz. zaženemo dadoteko s spodnjimi ukazi dobimo sliko na desni):

```
import numpy as np
import matplotlib.pyplot as plt
```

A=10
tau=2
omega=20

t=np.arange(0,5,0.01)
*i=A*np.exp(-t/tau)*np.cos(omega*t)*

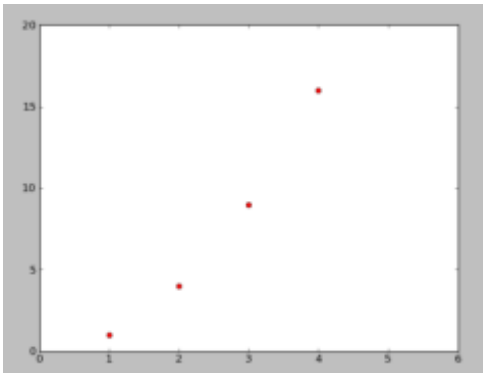
plt.plot(t,i)
plt.show()



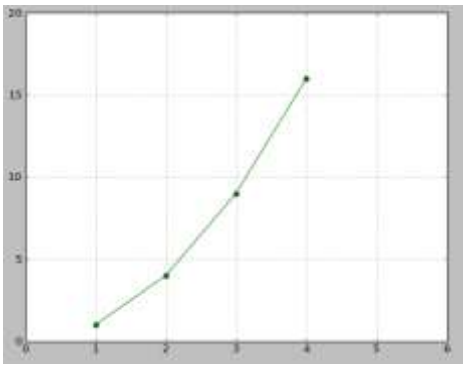
Sedaj se je potrebno navaditi na nekatere ukaze, ki jih programski jezik Python omogoča. V nadaljevanju prikažimo nekaj primerov, ki jih lahko uporabite (in seveda ustrezno preoblikujete) v praksi.

Primer 2:

```
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```

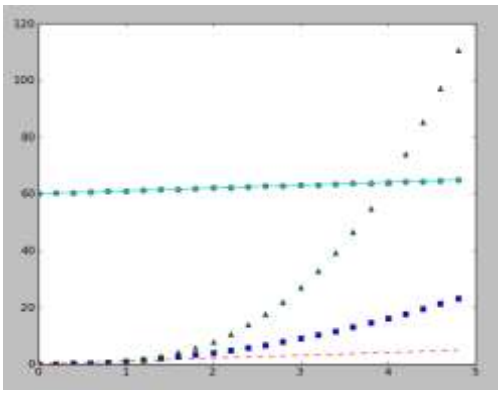


izrišemo krivuljo, ki ima x vrednosti 1,2,3,4 in y vrednosti 1,4,9,16. Krivulja se izriše z rdečo barvo in krogci ob vrednostih. Meje abscise so od 0 do 6, ordinate od 0 do 20. Enak rezultat bi dobili, če bi definirali dva niza (x in y) kot
x=[1,2,3,4]
y=[1,4,9,16]
plt.plot(x,y, 'g-o')
in ju uporabili v plot ukazu. V tem primeru izrišemo zeleno polno črto s krogci in z ukazom plt.grid(True) dodamo še črte (grid).



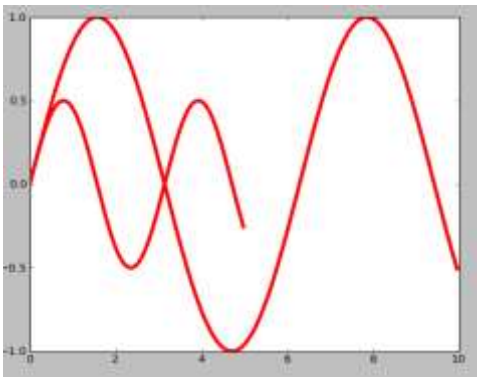
Primer 3: Izrišemo štiri krivulje z različnim označevanjem.

```
t = np.arange(0., 5., 0.2)
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g
plt.plot(t, t+60, 'o', linestyle='-', color='c')
plt.show()
```



Primer 4: Dve krivulji z različnima abscisama in spremembo debeline črte.

```
x = np.arange(0, 10, 0.055)
y = np.sin(x)
plt.clf()
lines=plt.plot(x, y, 'r', x/2, y/2, 'b')
plt.setp(lines, color='r', linewidth=4.0)
plt.show()
```

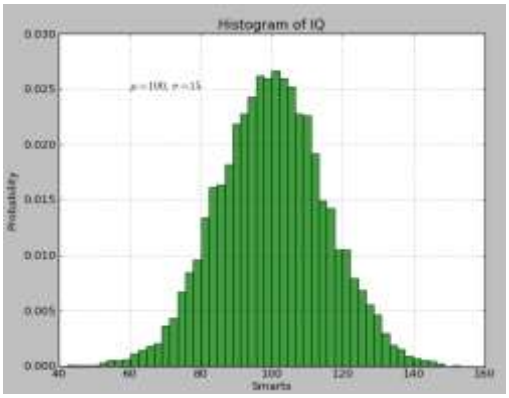


Primer 5: Izris histograma z označitvijo osi in tekstom znotraj grafa:

```
mu, sigma = 100, 15
x = np.random.normal(mu, sigma, size=10000)

histvals, binvals, patches = plt.hist(x, bins=50
normed=True, facecolor='g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```



4. Korak: branje podatkov, izračun in izris

nedokončano

```
data =np. loadtxt("C:\Anaconda\datafile.txt")

#'data' is a matrix containing the columns and rows from the file
mass   = data[:,0]  # Python indices are (row,col) as in linalg
radius = data[:,1]  # Creates arrays for first two columns

# Create a  plot of data
np.plot(mass,radius)
#xlabel(r'Mass ($M_{\odot}$)')
#ylabel(r'Radius ($R_{\odot}$)')

# Turn on a grid
grid(True)
```