Programiranje 2
Izvajanje programov

Vladimir Batagelj

Univerza v Ljubljani, FMF

Marec 2014/Marec 2012

Kazalo

Zvok

Zvok na računalniku lahko uporabljamo v obliki posnetkov (shranjenih na datotekah). Predvajanje posnetkov omogoča metoda `PlaySound` iz knjižnice `winsound`:

```
from winsound import *
PlaySound("c:/users/Batagelj/test/python/zvok/birthday03.wav",
  SND_ASYNC+SND_LOOP+SND_FILENAME)
PlaySound("c:/users/Batagelj/test/python/zvok/birthday03.wav",
  SND_PURGE)
```

V knjižnici `winsound` je na voljo metoda `Beep(`v`,`t`)`, ki ustvari zvok višine v, ki traja t milisekund.

```
from winsound import Beep
from time import time, ctime, sleep
n=0
while n<5:
   print(n,ctime(time()))
   sleep(5)
   n += 1
   Beep(210+50*n, 500)
```

Za zahtevnejšo glasbo se uporablja standard MIDI. Obstaja več knjižnic za delo s posnetki `sndhdr`, `audioop`, `sunau`, `wave`, ...

Note

V glasbi praviloma uporabljamo le tone izbranih višin, ki sestavljajo *tonski sistem*. Za označevanje osnovnih tonov uporabljamo v evropski glasbi običajno črke abecede c, d, e, f, g, a, h (angleži uporabljajo namesto črke h črko b) ali pa solmizacijske zloge *do*, *re*, *mi*, *fa*, *so* (*sol*), *la*, *ti* (*si*). V tonskem sistemu se zaporedje osnovnih tonov večkrat ponovi. Del tonskega sistema med zaporednima ponovitvama oznake tona imenujemo *oktava*. Glede na osnovno malo oktavo označujemo tone v nižjih oktavah z velikimi črkami in odmikom, npr. D_2; tone v višjih oktavah pa z malo črko in odmikom, npr a^1.
Razmiki med toni so večji ali manjši. Manjšemu pravimo polton, večjemu pa celi ton. Dva poltona sestavljata celi ton. Z višajem $\sharp$ označujemo zvišanje tona za polton, z nižajem $\flat$ pa znižanje tona za polton.

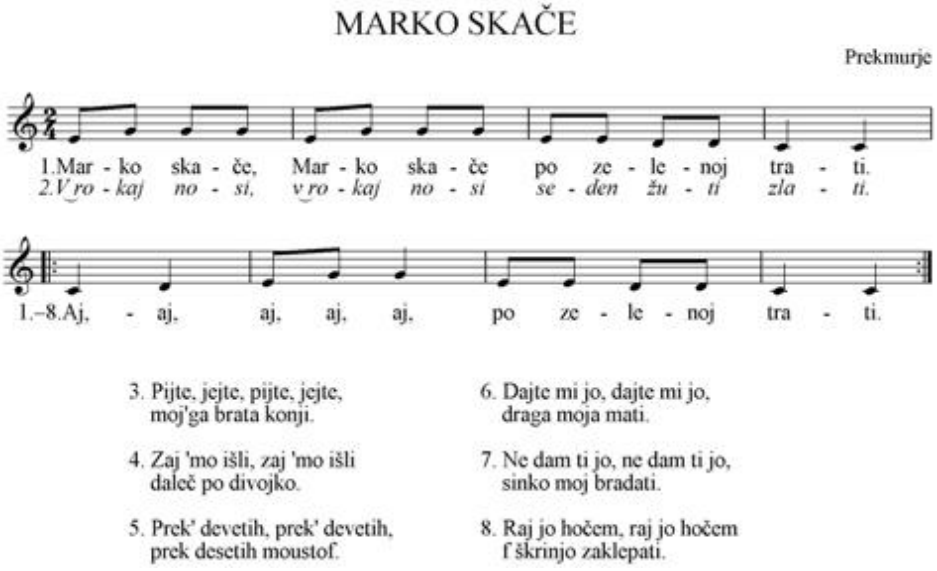
V računalništvu navadno uporabljamo c-durovo lestvico in tonski sistem, ki temelji na komornem tonu $a^1 = 440$ Hz. V tem sistemu sestavljajo toni geometrijsko zaporedje. Razmerje med višinama zaporednih tonov je enako $\sqrt[12]{2} = 1.05946$. To zagotavlja, da ima posamezni ton v nižji (višji) oktavi polovično (dvojno) višino.

Tonski sistem

ton	O	o	o^1	o^2	o^3
c	65	131	262	523	1046
$c\sharp$ ($d\flat$)	69	139	277	554	1109
d	73	147	294	587	1175
$d\sharp$ ($e\flat$)	78	156	311	622	1244
e	82	165	330	659	1318
f	87	175	349	698	1397
$f\sharp$ ($g\flat$)	92	185	370	740	1480
g	98	196	392	784	1568
$g\sharp$ ($a\flat$)	104	208	415	831	1661
a	110	220	440	880	1760
$a\sharp$ ($h\flat$)	117	233	466	932	1865
h	123	247	494	988	1976

Marko skače

```
from winsound import Beep
d = 240
def do(): Beep(262,d)
def re(): Beep(294,d)
def mi(): Beep(330,d)
def fa(): Beep(349,d)
def so(): Beep(392,d)
def la(): Beep(440,d)
def ti(): Beep(494,d)
def do1(): Beep(523,d)
def double():
   global d; d = 2*d
def halve():
   global d; d = round(d/2)
def MarkoSkace():
   global d
   d = 300
   mi(); so(); so(); so()
   mi(); so(); so(); so()
   mi(); mi(); re(); re()
   double()
   do(); do()
   do(); re()
   halve()
   mi(); so()
   double()
   so()
   halve()
   mi(); mi(); re(); re()
   double()
   do(); do()
```



Vislice – ugibanje besed

Zamisel programa je naslednja:

- preberi seznam besed z datoteke
- izberi slučajno besedo
- začetni vzorec je niz ????? dolg kot beseda
- ponovi do največ krat napačnih ugibanj:
  - izpiši trenutni vzorec in že izbrane črke
  - povprašaj po novi črki
  - preveri ali se ugibek nahaja v besedi: če se, ga postavi na ustrezna mesta v vzorcu, sicer sporoči, da je črka napačna
  - če je beseda odkrita končaj
- izpiši sporočilo o uspešnosti.

Slučajna števila

Včasih tudi na računalniku potrebujemo, da se kaj zgodi slučajno. Običajno nam to v izbranem programskem jeziku omogoča funkcija, ki nam ob vsakem klicu vrne kot vrednost slučajno realno število. Zaporedje teh vrednosti je enakomerno porazdeljeno na intervalu $[0, 1)$. V Pythonu so slučajnostne metode/funkcije zbrane v knjižnici random. Mi bomo omenili le metode:

`random()` – vrne naslednje slučajno realno število na $[0, 1)$.

`seed(`n`)` – nastavi seme funkcije `random()`; n je celo število; če je opuščeno ali `None`, uporabi za nastavitev sistemski čas.

`randint(`a, b`)` – a in b sta celi števili; vrne enakomerno porazdeljeno celo število z intervala $[a, b]$.

S postavitvijo semena z zahtevo `seed(`n`)` na isto izbrano celo število n lahko zagotovimo ponovljivost 'slučajnega' dogajanja – funkcija `random()` vrača isto zaporedje števil.

... Slučajna števila

```
>>> from random import random, seed
>>> (random(),random(),random())
(0.16163844164173513, 0.44569756158877627, 0.7794757305630318)
>>> seed(2010)
>>> (random(),random(),random())
(0.13915677287970796, 0.48665251176123714, 0.22860735382531394)
>>> seed(2010)
>>> (random(),random(),random())
(0.13915677287970796, 0.48665251176123714, 0.22860735382531394)
```

Ker v funkciji `random` zaporedja slučajnih števil računamo – naslednje število je določeno s prejšnjimi, ta zaporedja niso čisto slučajna – pravimo jim *psevdo-slučajna* zaporedja. Praviloma je zagotovljeno, da se obnašajo kot slučajna zaporedja.

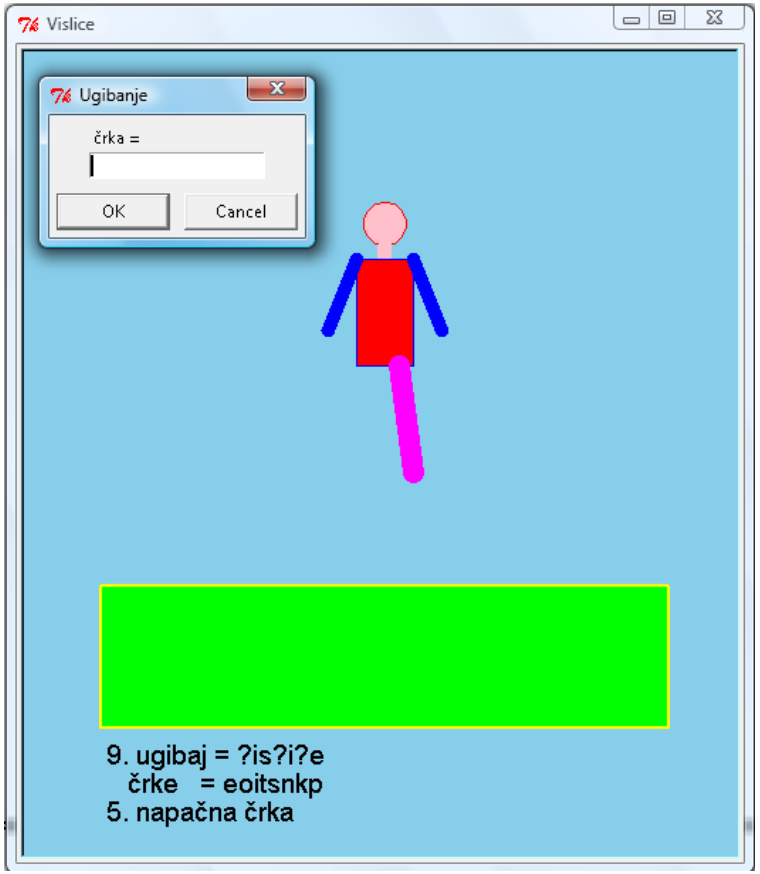
Drugače porazdeljena zaporedja števil je mogoče pridobiti iz osnovnega zaporedja `random`. Npr.

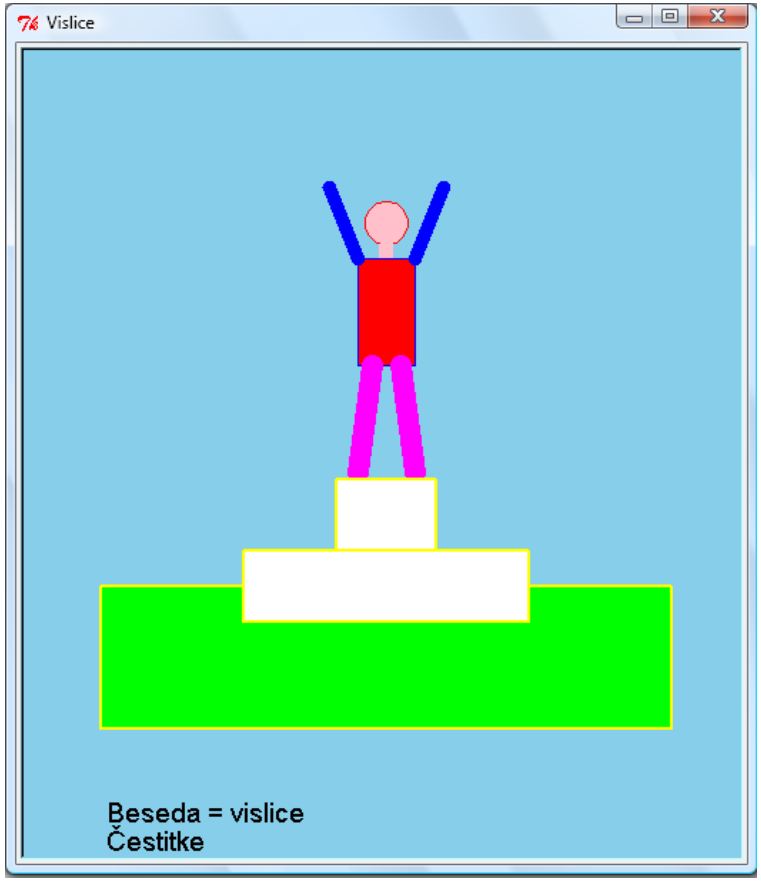
```
>>> from random import random
>>> from math import trunc
>>> def randint(a,b): return a + trunc((b+1-a)*random())
>>> for i in range(50): print(randint(1,6),end=',')
5,1,4,5,4,4,5,6,2,6,6,1,6,5,3,5,2,5,1,3,3,3,4,5,6,
6,2,2,2,5,6,4,5,3,6,3,3,6,4,3,3,3,3,3,6,2,1,2,5,3,
```

Vislice

```
from random import seed, randint
def run(sezBesed):
   try:
      besede = open(sezBesed,'rU').readlines()
   except IOError: print("Težave z datoteko",sezBesed)
   else:
      seed(None)
      beseda = besede[randint(0,len(besede)-1)].strip().lower()
      vzorec = "?"*len(beseda)
      izbrane = ""; odkrita = False; krat = 5; narobe = 0; k = 0
      while narobe < krat:
         k += 1
         print("\n",k,". ugibaj = ",vzorec,sep='')
         print("   črke   =",izbrane,"\n")
         znak = input("črka = ")[0].lower()
         izbrane += znak; vzorecNov = ""
         for i,z in enumerate(beseda):
            vzorecNov += znak if znak==z else vzorec[i]
         if vzorec==vzorecNov:
            narobe += 1
            print(narobe,". napačna črka",sep='')
         else: odkrita = beseda==vzorecNov
         if odkrita: break
         vzorec = vzorecNov
      print("\nBeseda =",beseda)
      print(["Obešen","Čestitke"][odkrita])
run(r'c:\test\python\vislice\besede.txt')
```

Vislice s sliko





Vislice s sliko 1

```
from random import seed, randint
from turtle import *

def pk(x,y,w,h,p,pc,fc):
   color(pc,fc); pensize(p); pu()
   begin_fill()
   setpos(x,y); pd(); setpos(x+w,y)
   setpos(x+w,y+h); setpos(x,y+h); setpos(x,y)
   end_fill()

def crta(x,y,s,t,p,pc):
   pencolor(pc); pensize(p); pu(); setpos(x,y); pd(); setpos(s,t)

def travnik(): pk(-200,-190,400,100,2,'yellow','green')

def vislice():
   crta(-130,-90,-130,205,20,'brown'); setpos(0,205)
   crta(0,195,0,180,2,'black')

def oder():
   pk(-100,-115,200,50,2,'yellow','white')
   pk(-35,-65,70,50,2,'yellow','white')

def glava():
   color('red','pink'); pensize(1); pu()
   begin_fill(); setpos(0,150); pd(); circle(15); end_fill()
   crta(0,140,0,150,10,'pink')

def telo(): pk(-20,65,40,75,1,'blue','red')
```

Vislice s sliko 2

```
def levaRoka(v): crta(20,140,40,140+v,10,'blue')

def desnaRoka(v): crta(-20,140,-40,140+v,10,'blue')

def levaNoga(): crta(10,65,20,-10,15,'magenta')

def desnaNoga(): crta(-10,65,-20,-10,15,'magenta')

def izpis(x,y,niz):
   pk(x,y,250,19,1,'skyBlue','skyBlue')
   pencolor('black')
   write(" "+niz,font=("Arial",14,"normal"))

def ugibaj(sezBesed):
   try:
      besede = open(sezBesed,'rU').readlines()
   except IOError:
      print("Težave z datoteko "+sezBesed)
   else:
      screensize(300,400,'skyBlue')
      title('Vislice'); reset(); ht(); travnik()
      seed(None); izberi = randint(0,len(besede)-1)
      beseda = besede[izberi].strip().lower()
      vzorec = "?"*len(beseda)
      krat = 6; izbrane = ""; odkrita = False; narobe = 0; k = 0
      while narobe < krat:
         k += 1
         izpis(-200,-220,str(k)+". ugibaj = "+vzorec)
         izpis(-200,-240,"   črke   = "+izbrane)
         while True:
            znak = textinput("Ugibanje","črka = ")
            if len(znak) > 0: break
         znak = znak[0].lower()
```

Vislice s sliko 3


```
            izbrane += znak; vzorecNov = ""
            for i,z in enumerate(beseda):
               vzorecNov += znak if znak==z else vzorec[i]
            if vzorec==vzorecNov:
               narobe += 1
               izpis(-200,-260,str(narobe)+". napačna črka")
               if   narobe==1: glava()
               elif narobe==2: telo()
               elif narobe==3: levaRoka(-50)
               elif narobe==4: desnaRoka(-50)
               elif narobe==5: levaNoga()
               elif narobe==6: desnaNoga()
            else: odkrita = beseda==vzorecNov
            if odkrita: break
            vzorec = vzorecNov
         if odkrita:
            reset(); ht()
            izpis(-200,-260,"Beseda = "+beseda)
            izpis(-200,-280,"Čestitke")
            travnik(); glava(); telo(); levaRoka(50); desnaRoka(50)
            levaNoga(); desnaNoga(); oder()
         else:
            izpis(-200,-260,"Beseda = "+beseda)
            izpis(-200,-280,"Obešen")
            vislice()
         exitonclick()

# ugibaj('besede.txt')
```

Poganjanje Pythonskih programov

- programski modul je v urejevalnikovem oknu.
- programski modul vnesemo z `import` v glavno okno vmesnika IDLE.
- programski modul poženemo z ukazne vrstice v ukaznem oknu.
- izvajanje sprožimo z dvoklikom na ikono programa.
- izvajanje sprožimo v Pythonskem programu.

Program na datoteki

Če nameravamo program poganjati na različne načine, ga 'opremimo' tako, kot je storjeno na datoteki vislice.py:

```
from random import seed, randint
from turtle import *
...

def ugibaj(sezBesed):
   """Program ugibaj(sezBesed) je izvedba Vislic - ugibanja
neznane besede slučajno izbrane iz seznama besed z
datoteke sezBesed.

V. Batagelj, januar 2010"""
   try:
      besede = open(sezBesed,'rU').readlines()
   ...

if __name__ == '__main__':
   import sys
   if len(sys.argv)>0: ugibaj(sys.argv[1])
   else: print(ugibaj.__doc__)
else:
   print(ugibaj.__doc__)
```

... Program na datoteki

```
>>> print(ugibaj.__doc__)
Program ugibaj(sezBesed) je izvedba Vislic - ugibanja
neznane besede slučajno izbrane iz seznama besed z
datoteke sezBesed.

V. Batagelj, januar 2010
>>> help(ugibaj)
```

Naslednja ukaza izvedemo v DOSovskem oknu

```
cd c:\users\Batagelj\test\python\vislice
c:\python31\python vislice.py besede.txt
```

ali pa ju shranimo na datoteko `vislice.bat` in zahtevamo

```
c:\users\Batagelj\test\python\vislice\vislice.bat
```

```
>>> import sys; import os
>>> wdir = r'c:\users\Batagelj\test\python\vislice'
>>> sys.path.append(wdir); os.chdir(wdir)
>>> from vislice import ugibaj
Program ugibaj(sezBesed) je izvedba Vislic - ugibanja
neznane besede slučajno izbrane iz seznama besed z
datoteke sezBesed.

V. Batagelj, januar 2010
>>> ugibaj('besede.txt')
```

IDLE

Programski modul je v urejevalnikovem oknu. Poženemo ga z izbiro Run. Dobro napisan program mora delovati v vseh teh primerih.

```
if __name__ == '__main__':
  a = sys.argv[1]; b = sys.argv[2]
  ...
```

Pogoj je izpolnjen, če je modul pognan kot program iz ukazne vrstice `python prog arg1 arg2`

Pri modulih-knjižnicah uporabimo ta del za preverjanje pravilnosti delovanja.

V glavno okno vmesnika IDLE vnesemo programski modul z `import`:

```
>>> wdir = 'C:/users/Batagelj/test/python/zvok'
>>> import sys
>>> sys.path = [wdir]+sys.path
>>> sys.path
['C:/users/Batagelj/test/python/zvok', 'C:\\Python31\\Lib\\idlelib',
'C:\\Windows\\system32\\python31.zip', 'C:\\Python31\\DLLs', 'C:\\Python
'C:\\Python31\\lib\\plat-win', 'C:\\Python31', 'C:\\Python31\\lib\\site-
>>> import note
Marko skače
>>>
```

DOS

Programski modul poženemo z ukazne vrstice v ukaznem oknu DOS:

```
C:\Users\Batagelj>cd test\python\zvok
C:\Users\Batagelj\test\python\zvok>C:\python31\python note.py
Marko skače
Pritisni na tipko

C:\Users\Batagelj\test\python\zvok>
```

Zagon z dvoklikom na ikono programa

It is useful to associate .py extensions with a Python interpreter. Start a DOS command line prompt and issue the commands

```
assoc .py=PyScript
ftype PyScript=python.exe "%1" %*
```

Depending on your Python installation, such file extension bindings may already be done. You can check this with

`assoc | find "py"`

To see the application associated with a file type, write

`ftype name`

where name is the name of the file type as specified by the `assoc` command. Writing `help ftype` and `help assoc` prints out more information about these commands along with examples.

```
C:\>assoc | find "py"
.py=Python.File
.pyc=Python.CompiledFile
.pyo=Python.CompiledFile
.pyw=Python.NoConFile

C:\>ftype Python.File
Python.File="C:\Python31\python.exe" "%1" %*
```

help assoc

```
C:\>help assoc
Displays or modifies file extension associations

ASSOC [.ext[=[fileType]]]

  .ext      Specifies the file extension to associate the file type with
  fileType  Specifies the file type to associate with the file extension

Type ASSOC without parameters to display the current file associations.
If ASSOC is invoked with just a file extension, it displays the current
file association for that file extension.  Specify nothing for the file
type and the command will delete the association for the file extension.
```

help ftype

```
C:\>help ftype
Displays or modifies file types used in file extension associations

FTYPE [fileType[=[openCommandString]]]

  fileType  Specifies the file type to examine or change
  openCommandString Specifies the open command to use when launching files
                    of this type.

Type FTYPE without parameters to display the current file types that
have open command strings defined.  FTYPE is invoked with just a file
type, it displays the current open command string for that file type.
Specify nothing for the open command string and the FTYPE command will
delete the open command string for the file type.  Within an open
command string %0 or %1 are substituted with the file name being
launched through the assocation.  %* gets all the parameters and %2
gets the 1st parameter, %3 the second, etc.  %~n gets all the remaining
parameters starting with the nth parameter, where n may be between 2 and 9,
inclusive.  For example:

    ASSOC .pl=PerlScript
    FTYPE PerlScript=perl.exe %1 %*

would allow you to invoke a Perl script as follows:

    script.pl 1 2 3

If you want to eliminate the need to type the extensions, then do the
following:

    set PATHEXT=.pl;%PATHEXT%

and the script could be invoked as follows:

    script 1 2 3

C:\>
```

Izvajanje sprožimo v Pythonskem programu

z ukazom exec

```
>>> exec('a=4; print("a=",a)')
a= 4
```

```
>>> wdir = 'C:/users/Batagelj/test/python/zvok'
>>> p = open(wdir+'/note.py','r')
>>> P = p.read()
>>> exec(compile(P,'','exec'))
```

ali z ukazom call iz knjižnice subprocess

```
>>> import os
>>> from subprocess import call
>>> print(os.path.abspath('.'))
C:\Users\Batagelj\test\python\zvok
>>> sts = call("c:/python31/python note.py", shell=False)
```

Še nekaj ukazov iz knjižnice os


```
>>> import os
>>> print(os.path.abspath('.'))
D:\Python31
>>> wdir
'd:/test/python/2011/prog'
>>> os.chdir(wdir)
>>> print(os.path.abspath('.'))
d:\test\python\2011\prog
>>> os.system('d:\python31\python note.py')
0
>>> os.system('note.py')
0
>>> os.path.exists(wdir)
True
>>> os.path.exists('d:\blabla')
False
>>> os.path.exists(wdir+'/note.py')
True
>>> os.path.isdir(wdir+'/note.py')
False
>>> os.path.isfile(wdir+'/note.py')
True
>>> os.environ['PATH']
'C:\\TeX\\texmf\\miktex\\bin;C:\\WINDOWS\\system32;C:\\WINDOWS;C:\\WINDOWS\\System32\\Wbem;
...
C:\\WINDOWS\\system32\\WindowsPowerShell\\v1.0'
>>> os.environ
environ({'TMP': 'C:\\DOCUME~1\\Vladimir\\LOCALS~1\\Temp',
'COMPUTERNAME': 'BATAGELJPREN1',
'USERDOMAIN': 'BATAGELJPREN1',
...,
'OS': 'Windows_NT',
'PROGRAMFILES': 'C:\\Program Files'})
>>> os.environ['windir']
'C:\\WINDOWS'
>>> os.system('dir > datoteke.txt')
0
>>>
```

Nameščanje Pythonskih paketov

Za to lahko uporabimo standardne arhivarske programe `zip`, `Winzip`, `gzip`, `Rar`, `bzip2`, `7z`, `tar`, ...
Za pripravo zahtevnejših namestitvenih programov, ki vključujejo tudi prevajanje delov napisanih npr. v C-ju, Python ponuja knjižnici `distutils` in novejšo (nadomestno) `packaging`.
Program v Pythonu lahko predelamo tudi v izvršljiv program za izbrani operacijski sistem. Za to obstaja več orodij. Pod Windowsi se najpogosteje uporabljata py2exe in cx-freeze ter na MAC OSih py2app. Zanimiv je tudi program PyInstaller.
Ta orodja prevod programa v kodo za PVM (Pythonski stroj) dopolnijo s PVM in potrebnimi knjižnicami ter združijo v samostojno izvršljivo celoto.
Trenutno (marec 2012) različica `py2exe` za Python 3 še ni dostopna.
Poglejmo, kako uporabimo program `cx-freeze`. Kratka navodila za cx-freeze so na http://cx-freeze.sourceforge.net/cx_Freeze.html

Priprava izvršljivih različic s cx-freeze

Na datoteki zSetup.py pripravimo navodila

```
from cx_Freeze import setup, Executable

base = "Win32GUI"

setup(
   name = "Vislice",
   version = "0.1",
   description = "Vislice - igra: ugibanje besed",
   executables = [Executable("vislice.py", base = base)]
)
```

ki jih poženemo s python zSetup.py build
build je področje, na katerem bomo ustvarili izvršljivo različico programskega modula. V področje na build moramo prepisati še morebitne podatke (za vislice – seznam besed besede.txt).

Priprava izvršljivih različic s `cx-freeze`

Zelo zmogljiv prost program za pripravo namestitvenih programov je Inno Setup. Različica QuickStart Pack vsebuje še ustrezni urejevalnik s "čarovnikom" za prijazno pripravo namestitvenih programov. Namestimo Inno Setup QuickStart Pack in poženemo InnoIDE. S čarovnikom (Wizard) pripravimo ustrezna namestitvena navodila za naš program in jih shranimo (Save). Izdelavo namestitvenega programa zahtevamo z izbiro Compile.