UPORABA IKT V NARAVOSLOVJU IN TEHNIKI

Predavanje 10

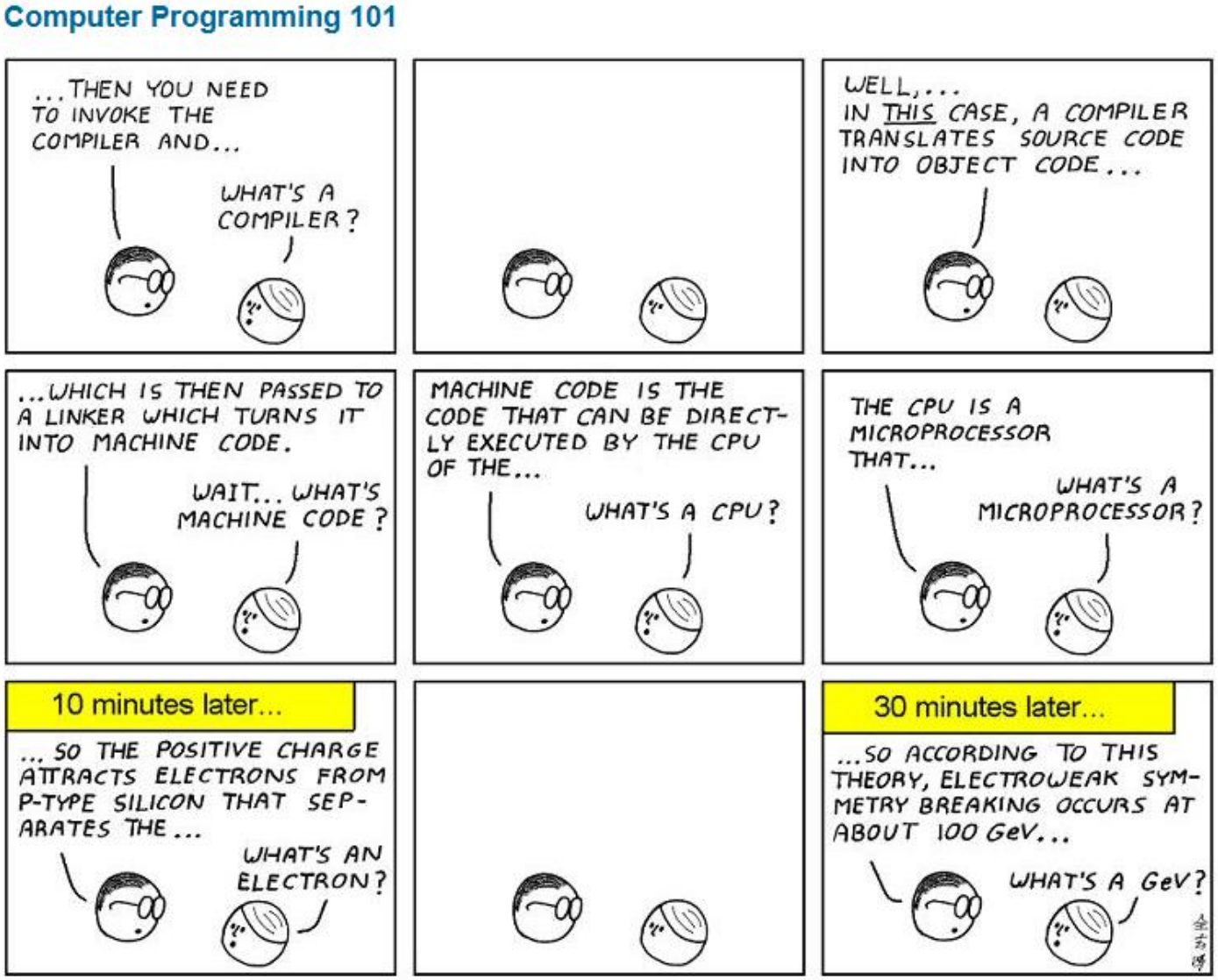
Uvod v programiranje - Python

doc.dr. Mira Trebar

Vsebina

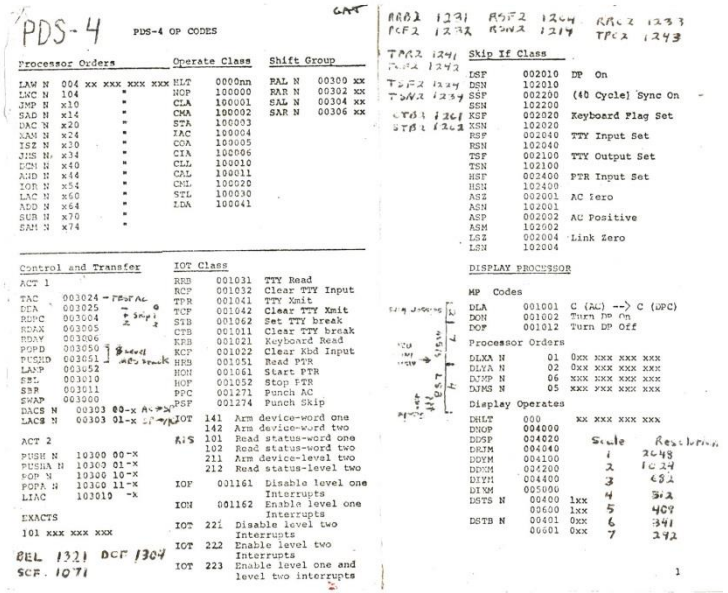
- Programski jeziki
  - Prevajalnik
  - Tolmač
- Python
  - Interaktivni tolmač
  - Podatkovni tipi
  - Spremenljivke
  - Funkcije
  - Pogojni stavki

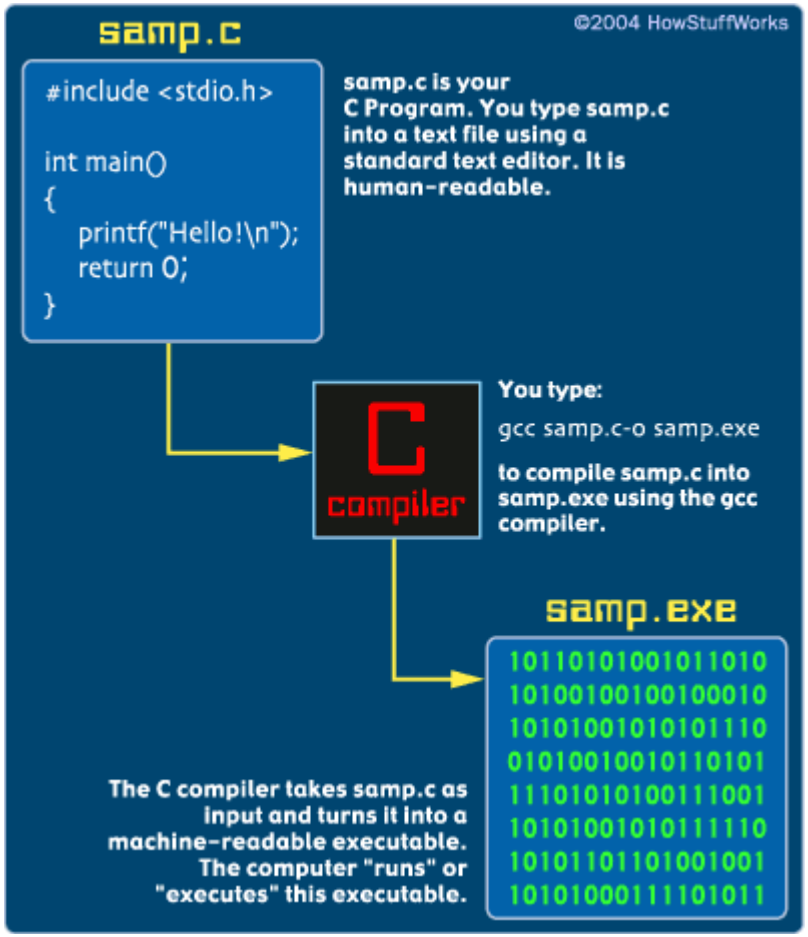
Računalniško programiranje



Programiranje - vprašanja

- KAJ?
- ZAKAJ?
- KAKO ?
- KDAJ?
- KJE?





Programski jeziki

- Nizkonivojski programski jeziki
  - **strojni jezik**
    - odvisen od strojne opreme

```
0001010101101100    156C
0001011001101101    166D
0101000001010110    5056
0011000001101110    306E
1100000000000000    C000
```

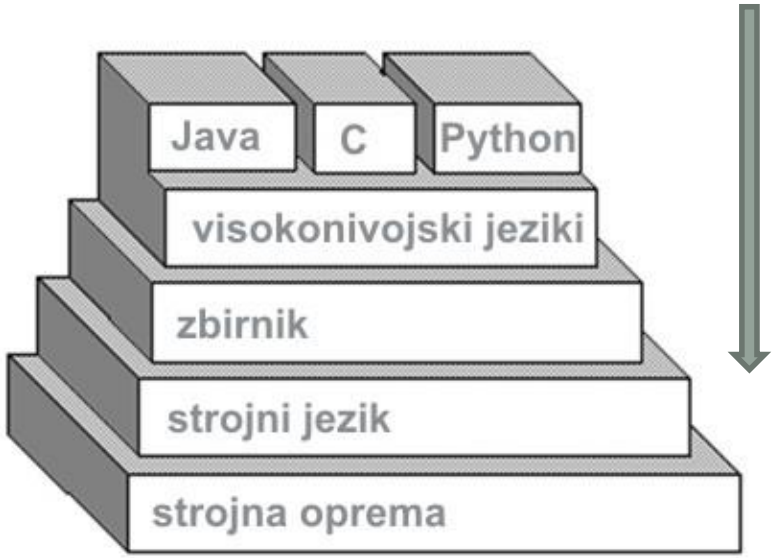
  - **zbirnik** (asembler)
    - mnemoniki zamenjajo številke
      - `ADDI = 50, ST = 30`
    - spremenljivke - pomnilniške lokacije
      - `operandA = 6C,`
      - `operandB = 6D,`
      - `vsota = 6E`

```
LD   R5, operandA
LD   R6, operandB
ADDI R0, R5, R6
ST   R0, vsota
HLT
```

    - mnogo bolj prijazen kot strojni jezik
    - še vedno popolnoma odvisen od strojne opreme

Programski jeziki

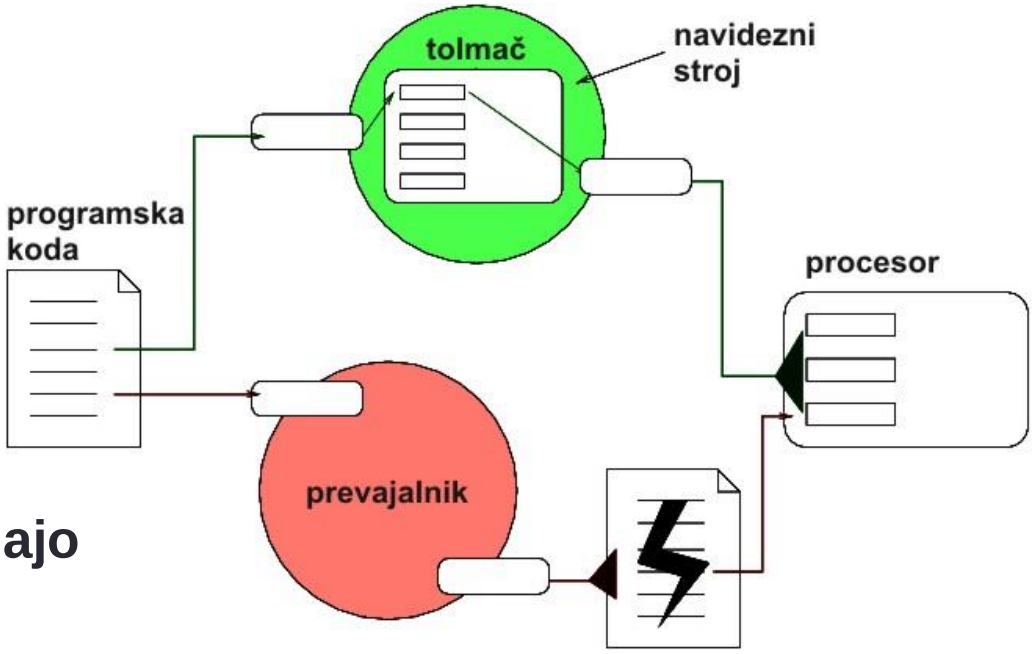
- visokonivojski programski jeziki
  - tretja generacija
  - neodvisni od lastnosti računalnikov
  - programska koda je prenosljiva
  - visokonivojski ukazi
    - angleške besede
    - podobnost z naravnim jezikom



- visokonivojske ukaze, ki so prilagojeni človeku, je potrebno preoblikovati v nizkonivojske ukaze, ki jih razume računalnik
  - slovnica programskega jezika je mnogo bolj striktna od slovnice naravnih jezikov
  - programska orodja, ki to znajo narediti so
    - **prevajalnik**i in
    - **Interpreterji (tolmači)**

Programski jeziki

- Prevajalnik (ang. compiler)
  - iz programov, napisanih v visokonivojskih ukazih, **sestavi programe z nizkonivojskimi ukazi**
  - programe z nizkonivojskimi **ukazi v strojem jeziku** shrani za kasnejšo izvedbo (izvršilne datoteke, .exe, .com)
- Tolmač (ang. interpreter)
  - visokonivojske ukaze **izvršuje enega za drugim** v visokonivojski obliki
    - za vsak visokonivojski ukaz ima pripravljene nizkonivojske zamenjave
  - **odziv je takojšen**
  - tolmačeni programi **se izvajajo počasneje** kot prevedeni

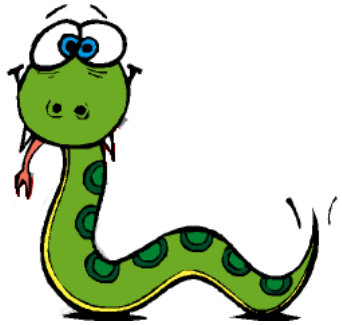


Programski jeziki

- Neodvisnost od strojne opreme
  - **Prenosljiva programska koda**
    - Potrebno prevajanje v izvršljivo datoteko za vsak računalnik in za vsak operacijski sistem posebej
    - Mnogokrat potrebni manjši popravki
      - Drugačen koncept dela z vhodno-izhodnimi napravami
      - Različni dialekti istega programskega jezika zaradi ohlapnih definicij
    - Jeziki: ADA, C, C++, FORTRAN
  - **Prenosljive izvršne datoteke**
    - Programi se prevajajo ali tolmačijo v nižjenivojske ukaze navideznega stroja (računalnika)
    - Navidezni računalnik (stroj) je pripravljen za vsak računalnik oziroma operacijski sistem posebej
    - Jeziki: Python, Java, C#
    - Navidezni stroji: Python, Java Virtual Machine, .NET framework

Programski jeziki

- Različni koncepti
  - imperativno ali **proceduralno programiranje**
    - algoritem (program) je opisan kot zaporedje enostavnejših korakov (podprogramov),
    - premikamo se iz enega stanja v drugega
    - primer: FORTRAN, C
  - **funkcijsko programiranje**
    - program je predstavljen kot matematična funkcija, ki sprejema vhode iz katerih izračuna izhode
    - primer: R, Mathematica, Excel
  - deklarativno ali **logično programiranje**
    - ne podamo algoritma ampak opišemo problem, ki ga hočemo rešiti
    - od programa zahtevamo kaj naj naredi, ne povemo mu kako
    - primer: Prolog
  - **objektno** usmerjeno **programiranje**
    - ukvarjamo se z objekti, ki imajo svoje lastnosti in metode
    - primer: C++, C#, Java, Python



Python - uvod

And now for something completely different …

Monty Python's Flying Circus

Python - definicija

- Python
  - je tolmačen objektni visokonivojski jezik z dinamično semantiko,
  - je izdelek odprtokodne skupnosti.
- V Pythonu
  - hitro in učinkovito brez pretiranega truda napišemo tako rekoč karkoli,
  - napisani programi so jasni in berljivi,
  - odlična kombinacija preprostosti in moči.
- Kdo, kdaj?
  - Guido van Rossum, 1990
  - na sistemih Linux za sistemsko administracijo
  - NASA za Integrated Planning System
  - Yahoo za podporo forumom
  - Google je v njem napisal precejšen del iskalnika
  - Zelo primeren je za začetnike

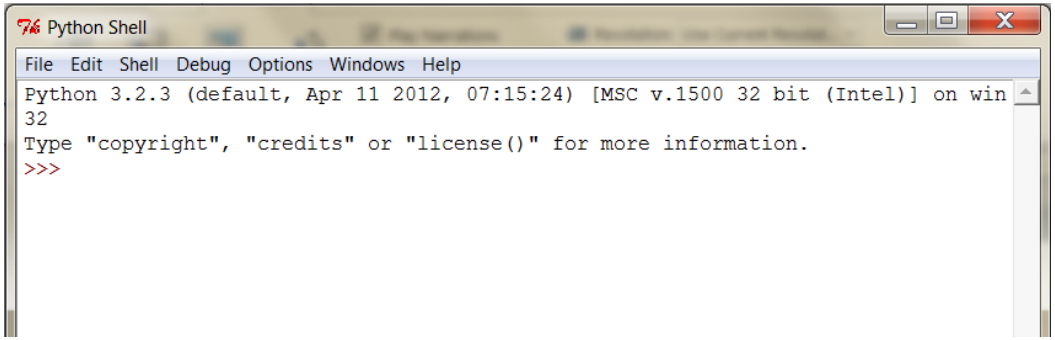
Python – okolje (1)

- Interaktivni tolmač in izvajalno okolje

  http://www.python.org

  različica 3.2.3

  zadnja verzija, 3.3.3 (2013)



- integrirano razvojno okolje **PyScripter**

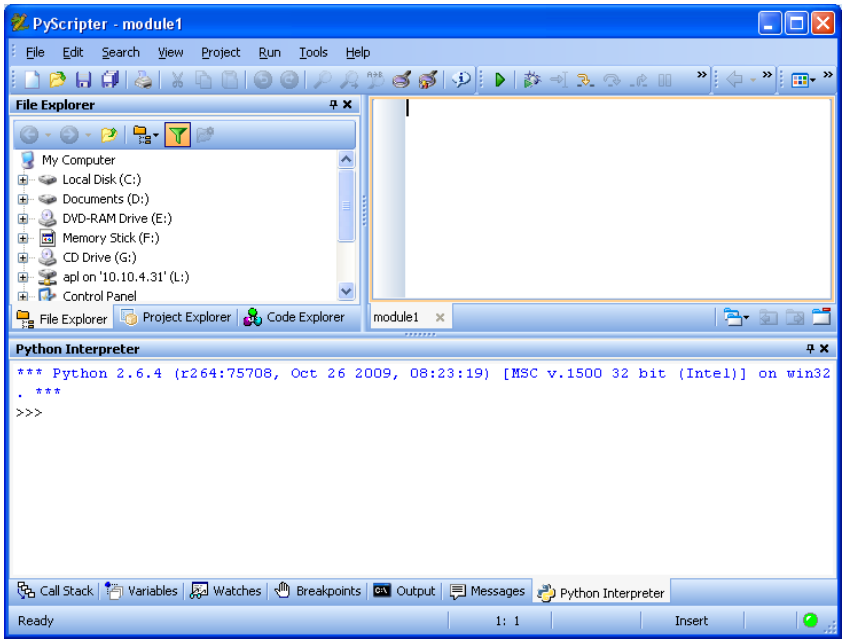
  http://code.google.com/p/pyscripter/

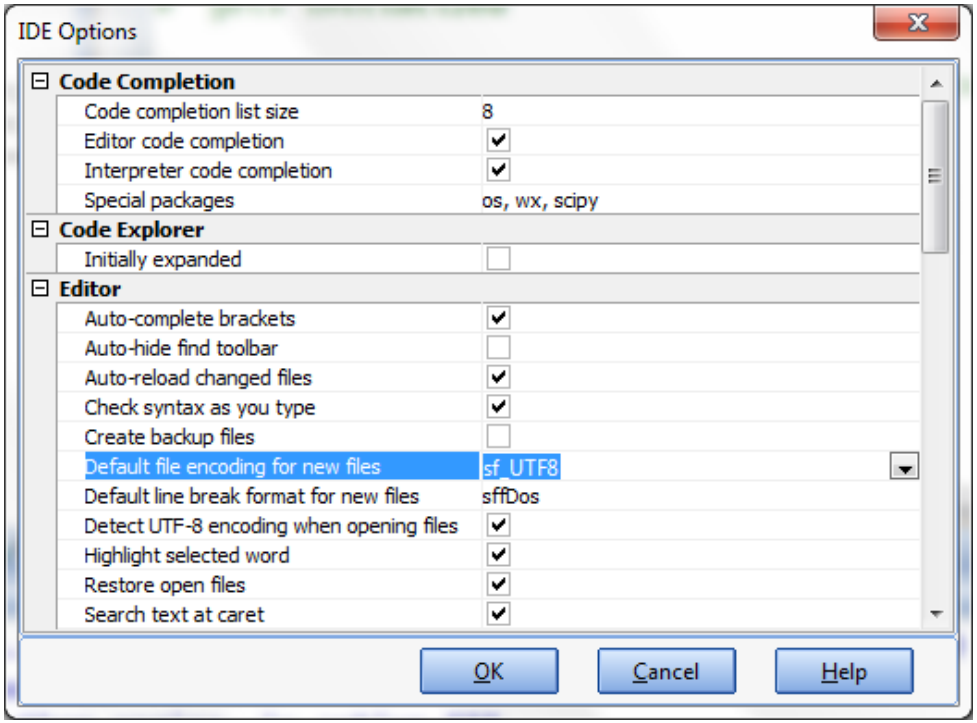
  Težave s slovensko tipkovnico in tipko Alt Gr (Ctrl + Alt)

  - V meniju izberemo Tools | Options | IDE Shortcuts in v novem oknu poiščemo
    - View, actViewFindResults odstranimo Ctrl+Alt+F
    - IDE Navigation, actNavBreakpoints odstranimo Ctrl+Alt+B
    - IDE Navigation, actNavVariables odstranimo Ctrl+Alt+V
    - IDE Navigation, actNavWatches odstranimo Ctrl+Alt+W

Python – okolje (2)

- Težave s shranjevanjem šumnikov
  - V meniju izberemo Tools|Options|IDE Options
  - Pod rubriko Editor poiščemo Default file encoding for new files
  - Nastavitev iz sf_Ansi spremenimo v sf_UTF8





Python – izrazi (1)

- Odprimo Python in poskusimo

```
>>> 2 + 2
4
>>>8 * 7
56
>>>12345678+87654321
999999999
```

- Imamo računalo. Vsekakor boljše kot Abak.

```
>>> 2+3 * 4+5
19
>>> 2 + 3*4 + 5
19
```

- Kot kaže presledki niso pomembni.

```
>>> (2+3) * (4+5)
45
>>> 2 + 3*4 + 5
19
```

Oklepaji vplivajo na rezultat.

Python – izrazi (2)

- Uporabni operatorji (potenciranje, deljenje, kvocient in ostanek):

```
>>> 5 ** 3
125
>>> 22 / 7
3.142857142857143
```

- Kaj pa to deljenje?

```
>>> 22 // 7
3
>>> 22 % 7
1
```

- Kaj pa če kaj pozabimo?

```
>>> (2+3) * (4+5
... 5
... )
45
```

- Python s tremi pikami pove, da od nas pričakuje še kaj.

Python – izrazi (3)

- Decimalna števila

```
>>> 3,4
(3, 4)
>>> 3.4
3.4
>>> 2.2 + 1.2
3.4000000000000004
```

- Očitno decimalne vejice ne razume pravilno.
- **Decimalno ločilo je pika!**
- Številke si predstavlja malo čudno. Razlog se skriva v predstavitvi števil v računalniku. No, če zaokrožimo je v redu.

Python - podatkovni tipi (1)

- *integer* ali *int* - cela števila,
- *floating point* ali *float* - necela števila
- *string ali str* - nizi znakov
- Pretvarjanje med podatkovnimi tipi:
  - Iz niza v številko

```
>>> int('1')
1
```

```
>>> float('1.23')
1.23
```

  - Iz cele številke v decimalno številko in niz

```
>>> float('1')
1.0
```

```
>>> str(1)
'1'
```

  - Iz decimalne številke v celo število in niz

```
>>> int(1.23)
1
```

```
>>> str(1.23)
'1.23'
```

Python - podatkovni tipi (2)

- Lepljenje (konkatenacija): niz ali *string* v narekovajih ''

```
>>> 'for' + 'mula'
Formula
```

- Preverjanje podatkovnega tipa:

```
>>> type(5)          >>> type(3.12)          >>> type('3.12')
<type 'int'>        <type 'float'>          <type 'str'>
```

- Kaj pa premi govor?

```
>>> 'Rekel je: 'Dober dan.''
  File "<interactive input>", line 1
    'Rekel je: 'Dober dan.''
                       ^
SyntaxError: invalid syntax
>>> 'Rekel je: "Dober dan."'
Rekel je: "Dober dan."
```

- Pri premem govoru uporabljamo dvojne narekovaje

Python – spremenljivke (1)

- Podatke in rezultate izračunov si zapomnimo v spremenljivkah.
- Poimenovanje spremenljivk (variable)
  - imena so lahko poljubno dolga (črka ali beseda)
  - začno se vedno s črko
  - nadaljujemo s črko ali številko, podčrtajem
  - **Uporabljamo samo črke angleške abecede!**
  - **Pyhon razlikuje med velikimi in malimi črkami**
- Nekaj primerov
  - x, y,
  - operand1
  - Koncentracija, molskaMasa
  - Avogadrovo_stevilo
- Dobra praksa:
  - iz imena spremenljivke naj bi razbrali kaj predstavlja
  - Vedno uporabljamo enak sistem poimenovanja

Python – spremenljivke (2)

- ključne besede - prepovedana imena spremenljivk

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

Primer:

```
>>> if = 3
 File "<interactive input>", line 1
    if = 3
         ^
SyntaxError: invalid syntax
```

Python – spremenljivke (3)

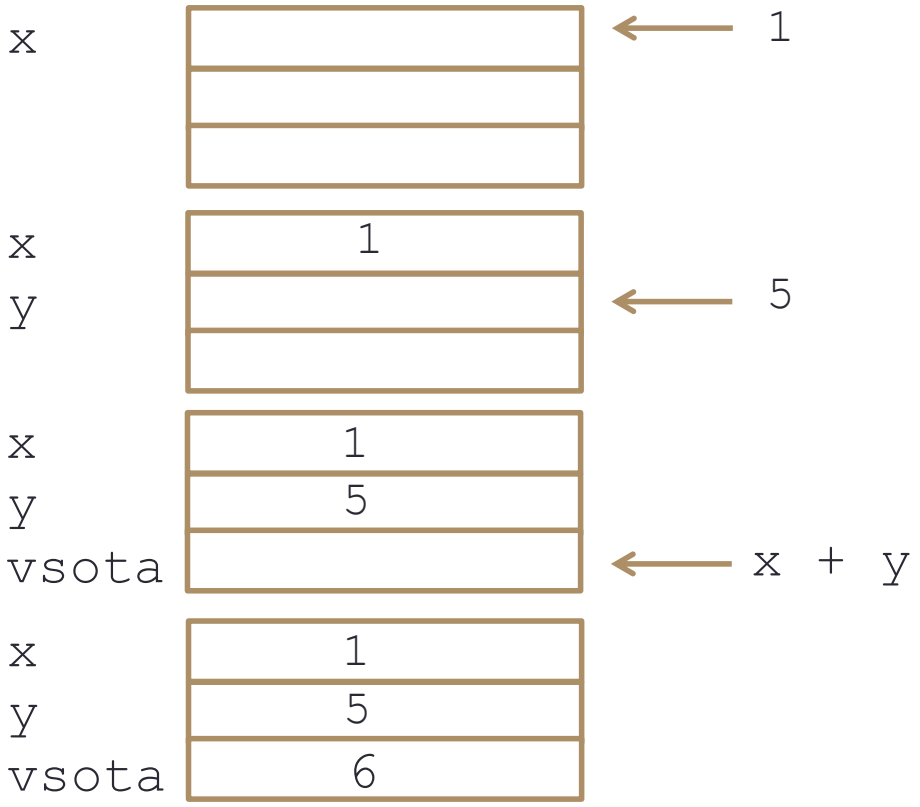
- Spremenljivka - spomin:

```
>>> x = 1

>>> y = 5

>>> x+y
6

>>> vsota = x+y
>>> vsota
6
```



Python – spremenljivke (4)

- Izračunajmo in si zapomnimo…

```
>>> x = 1 + 2
```

- Prireditveni stavek - levo od enačaja je ime, desno pa izraz.
- Tokrat nismo dobili nobenega odgovora. Python si je rezultat samo zapomnil pod imenom x. Kako lahko ta x uporabimo? Preprosto

```
>>> x * x
9
```

- x programerji označujemo kot spremenljivko
- Iz spremenljivke lahko izračunamo tudi nove spremenljivke

```
>>> y = 5*x
>>> y
15
```

Python – spremenljivke (5)

- Je tole matematično korektno?

```
>>> x = 1
1
>>> x = 2
2
```

- V matematiki se spremenljivke ne spreminjajo.
  Pri programiranju s tem ni nobenih težav.
- Kaj pa tole?

```
>>> x = x + 1
>>> x
3
```

- Saj ni res, pa je! Programiranje - enačaj označuje prireditev.
- Razmislek!
  - Python najprej izračuna izraz na desni `x + 1`,
  - nato rezultat, to je 3, priredi spremenljivki `x` na desni.
  - Prireditveni stavek `x = x + 1` preprosto pomeni povečaj `x` za 1.

Python - funkcije (1)

- Naše računalo zna še mnogo stvari: abs, pow

```
>>> abs(-13.5)
13.5
>>> pow(10, 2)
100
>>> pow(100, 0.5)
10
```

- Funkcija v programu nima enakega pomena kot v matematiki.
  - Matematika - funkcija ima pri izbrani vrednosti določeno vrednost
  - Računalnik - funkcijo pri določenih vhodnih podatkih (parametrih) izračuna, zato jo moramo poklicati. Funkcija je v zgornjem primeru izračunala vrednost in jo vrnila.
- Klic funkcije je tudi izraz. Lahko zapišemo tudi kaj takega

```
>>> xx = 15 + abs(-5)
>>> xx
20
```

Python – funkcija izpis (2)

- Izpis na zaslon

```
>>> print('Lep pozdrav!')
Lep pozdrav!
```

- Funkcije `print` ne potrebujemo, ko se pogovarjamo s Pythonom preko ukazne vrstice.
- Pri pisanju pravih programov jo moramo uporabiti vedno, kadar želimo, da nam kaj izpiše.

Python - funkcije (3)

- Vnašanje podatkov preko tipkovnice

```
>>> ime = input('Vpisi ime: ')
>>> ime
Miha
>>> izpis = 'Pozdravljen, ' + ime + '!'
>>> izpis
'Pozdravljen, Miha!'

>>> stevilka = input('Vpisi stevilko: ')
>>> stevilka*stevilka
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'str'
>>> stevilka
'42'
```

- Vpisali smo 42. Python si jo pri branju predstavlja kot niz znakov, torej '42'. Nizov znakov se ne da zmnožiti, zato napaka.

Python - funkcije (4)

- Vnašanje podatkov preko tipkovnice
  - Z input vedno preberemo niz, ki ga moramo spremeniti v število

```
>>> stevilkaNiz = input('Vpisi stevilko: ')
>>> stevilkaNiz
'42'
>>> stevilka = int(stevilkaNiz)
>>> stevilka
42
>>> stevilka * stevilka
1764
```

  - Krajša rešitev:

```
>>> stev = int( input('Vpisi stevilko: ') )
>>> stev
42
>>> stev * stev
1764
```

Python - funkcije (5)

- Zdaj pa pozdrav in računanje kvadrata povežimo

```
>>> ime = input('Vnesi ime: ')
>>> izpis = 'Pozdravljen, ' + ime + '!'
>>> print(izpis)
'Pozdravljen, Miha!'

>>> stev = int(input('Vpiši število))
>>> print('Kvadrat števila', stev, 'je'
 stev*stev)
Kvadrat števila 5 je 25
>>> input('Pritisni <Enter>')
```

Python - višja matematika

- Trigonometrija, logaritmi, eksponenti?

```
>>> exp(3)
    Traceback (most recent call last):
      File "<interactive input>", line 1, in <module>
    NameError: name 'exp' is not defined
>>> from math import sin, cos
>>> tan(10)
    Traceback (most recent call last):
      File "<interactive input>", line 1, in <module>
    NameError: name 'tan' is not defined
>>> from math import *      (Pred uporabo funkcij)
>>> tan(10)
    0.6483608274590866 3
>>> radians(180)
    3.141592653589793 1
>>> degrees(pi)
    180.0
```

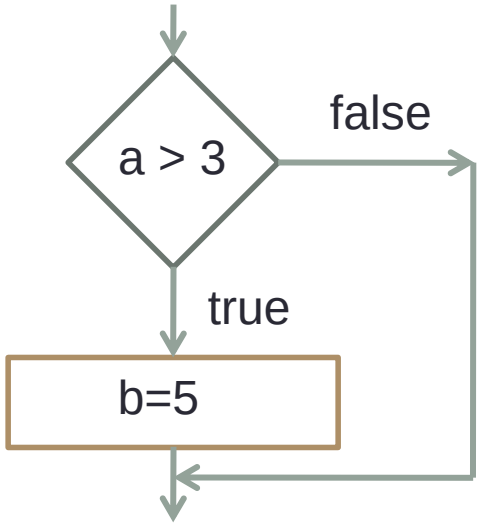
Pogojni stavki (1)

- Stavka if in if pogoj: else - najdemo na listi rezerviranih besed.
  - Besedi if mora slediti pogoj, temu pa dvopičje.
  - Stavki, ki sledijo pogoju se bodo izvedli le, če je pogoj resničen.
  - Zgornjim stavkom lahko sledi vrstica z else in dvopičjem.
  - Stavki, zapisani za to vrstico se bodo izvedli, kadar pogoj ni izpolnjen.
  - Pri prijaznejšem programu sta if in else poravnana. Zakaj? Preprosto zato, ker sodita skupaj.
- **Presledki** na začetku vrstice izjemno pomembni
  - Stavke, ki se izvedejo ob izpolnjenem pogoju je treba zamakniti.
  - In to vse enako, za koliko ni pomembno.
  - Dogovorimo se za 4 presledke. V PyScripterju 4 presledke dobimo tako, da stisnemo tipko TAB.
  - Enako velja za stavke, ki se izvedejo, kadar pogoj ni izpolnjen.

Pogojni stavki (2)

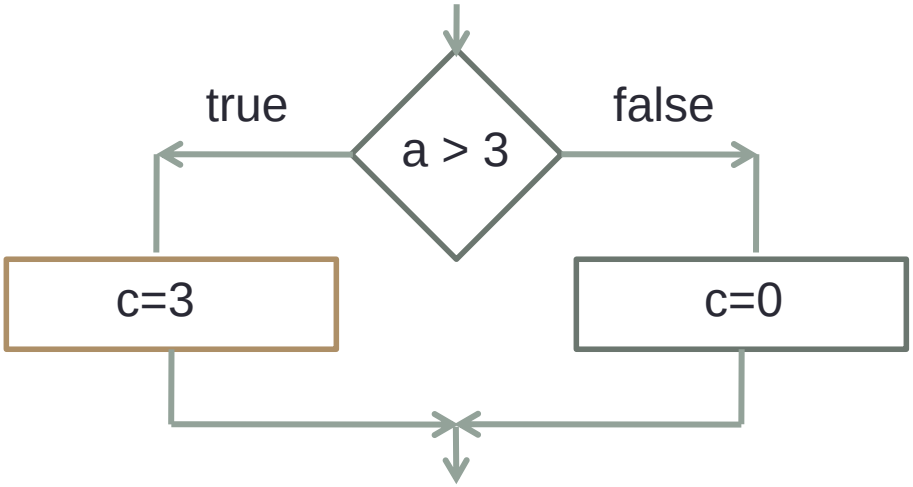
- Pogojni stavek: if pogoj: ...

```
>>> a=5
>>> if a>3: b=5
>>> b
5
```



- Pogojni stavek: if pogoj: ...
  else

```
>>> a=5
>>> if a>3: c=3
else: c=0
>>> c
3
```



Gnezdenje pogojnih stavkov (1)

```
x = int(input('Vpisi število:'))
if x < 0:
    print('Negativno število.')
else:
    if x == 0:
        print('Število 0.')
     else:
        if x > 0:
            print('Pozitivno število.')
        else:
            print('ni število.')

if x < 0:
        print('Negativno število.')
    elif x ==0:
        print(,Število 0.')
    elif x > 0:
        print('Pozitivno število.')
    else:
         print('ni število.')
```

Zapisovanje pogojev (1)

- Pogoj je izraz
  - pri računanju logičnega izraza delamo z logičnima vrednostma True in False.
  - prednostna lista: not, and,or
    - not se izračuna pred and, and pa pred or
    - če hočemo drugače, uporabimo oklepaje
  - primerjalni operatorji
    - <, <=, >, >=, ==, !=
    - dvojni enačaj!!!

```
>>> True and True
True
>>> True and False
False
>>> False and False
False
>>> True or True
True
>>> True or False
True
>>> False or False
False
>>> not False
True
>>> not True
False
```

Zapisovanje pogojev (2)

- Enostavni pogoji

```
>>> 5 < 10
True
>>> 5 >= 10
False
>>> 5 == 5
True
>>> 5 != 5
False
```

- sestavljeni pogoji
  - primerjalni operatorji imajo prednost pred logičnimi
  - vseeno je z oklepaji bolj pregledno

```
>>> 10 > 3 and -5 < 2
True
>>> (10 > 3) and (-5 < 2)
True
>>> (10 > 3) or (-5 > 2)
True
>>> not (10 == 5)
True
```

Zapisovanje pogojev (3)

- Pythonove posebnosti

```
>>> 5 < 10 < 15
True
>>> 5 < 0 < 15
False
>>> 5 < 20 < 15
```

- Pogoji dobijo smisel šele potem, ko vanje vstavljamo spremenljivke

```
>>> x = 10
>>> 15 > x > 5
True
>>> x = 5
>>> 15 > x > 5
False
```

Zapisovanje pogojev (4)

- Kako poteka računanje?

```
>>> (10 < 5 or 100 > 0) and 5 < 0
False
```

1. 10 < 5 → nepravilno
2. 100 > 0 → pravilno
3. nepravilno (1.) **or** pravilno (2.) → pravilno
4. 5 < 0 → nepravilno
5. pravilno (3.) and nepravilno (4.) → nepravilno

Zapisovanje pogojev (5)

- Pomembnost operatorjev pri računanju aritmetičnih in logičnih izrazov

pomembnost narašča

operator	opis
`or`	logični ali
`and`	logični in
`not x`	logični ne
`in, not in`	pripadnost množici
`is, is not`	preverjanje indentitete
`<, <=, >, >=, !=, ==`	primerjave
`+, -`	seštevanje in odštevanje
`*, /, //, %`	množenje in deljenje
`+x, -x`	pozitino in negativno
`**`	potenciranje
`x[indeks]`	dostop do elementa v listi
`x[index1:index2]`	več elementov v list
`f(parametri ...)`	klic funkcije

Literatura

- Knjiga
  - M. L. Hetland: Beginning Python from Novice to Professional, Apress, Berkeley, California, 2005.
- Knjige v formatu PDF
  - J. Cogliati: Non-Programmers Tutorial For Python, samozaložba, 2002
  - A. Downey, J. Elkner, C. Meyers: How to Think Like a Computer Scientist, Learning with Python, Green Tea Press, Wellesley, Massachusetts, 2008
  - J. Demšar: Python za programerje, Založba FE in FRI, Ljubljana, 2008.
  - P. Jerina: Spoznajmo P<ython, učbenik za učence (seminarska naloga).
- Video
  - Tečaj Pythona v slovenščini: http://www.praktik.si/ShowGroups.aspx?categoryId=3