podatkovna baza

Delo s podatki

podatkovna baza, podatkovni model, obdelava podatkov
relacijski podatkovni model
tabele, podatkovni tipi, ključi, povezave
preglednice in grafikoni

Model realnosti

- predmeti in dogodki v realnosti so **entitete**, npr. oseba, avtomobil, nogometna tekma…

- *Primer: entiteta = oseba*

  zanima nas atribut "rojstni datum", ne zanima pa nas atribut "barva oči".
- entitete, ki imajo eno ali več skupnih lastnosti, lahko združimo v **entitetno množico**, npr. članice iste ekipe

Podatkovna baza

Podatkovna baza je model, ki vsebuje entitete in lastnosti, ki jih preučujemo in tiste povezave, ki nas zanimajo pri čemer delovanje modela ustreza razmeram v realnosti.

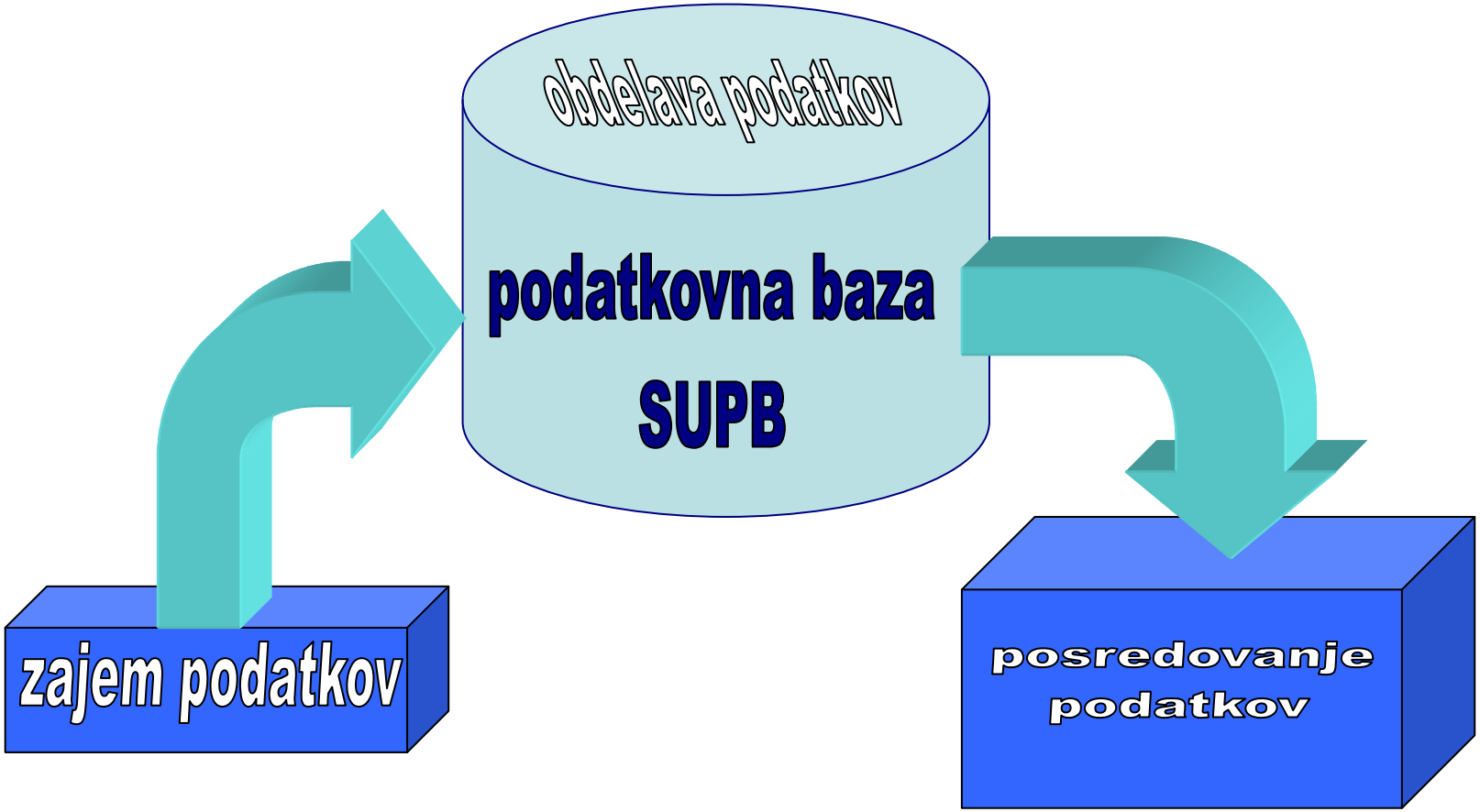
Podatkovna baza mora izpolnjevati naslednje pogoje (ANSI):

1. urejen vrstni red podatkov
2. sočasno jo lahko uporablja en ali več uporabnikov
3. podatki se ne ponavljajo
4. podatkovna baza je shranjena v računalniku

- **AOP** – sistem za **a**vtomatsko **o**bdelavo **p**odatkov
- **SUPB** (**s**istem za **u**pravljanje **p**odatkovnih **b**az) – zbirka programov, ki omogočajo shranjevanje, spreminjanje in iskanje podatkov v podatkovni bazi

en računalnik, en SUPB
centralizirana podatkovna baza

več računalnikov, različne lokacije, več SUPB
porazdeljena podatkovna baza



Podatkovni modeli

Podatkovni model je formalna struktura s katero realnost predstavimo z relevantnimi podatki.

Podatkovni model vsebuje:

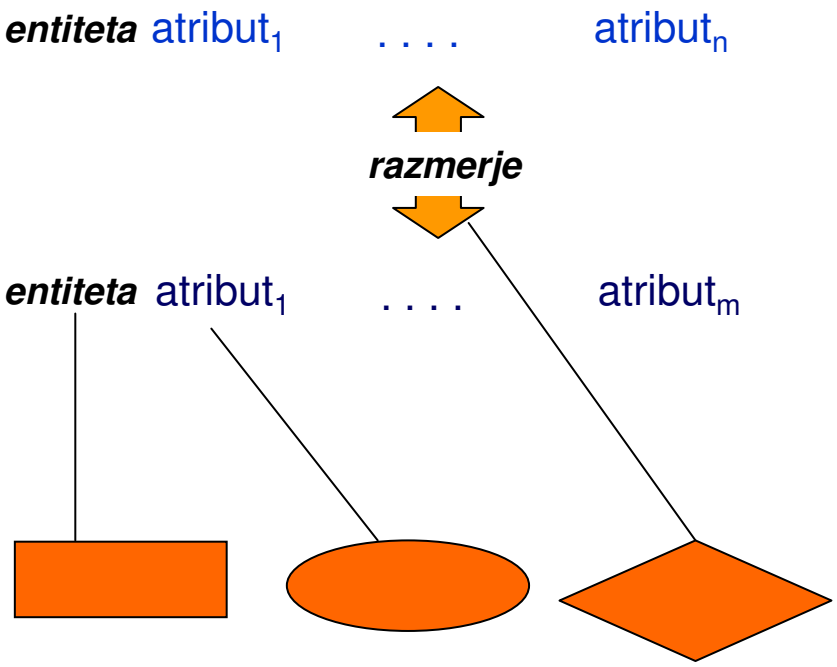
- pravila, ki določajo organizacijo podatkov
- strukturo podatkov
- dovoljene operacije nad podatki

Pri načrtovanju podatkovnega modela razvijamo

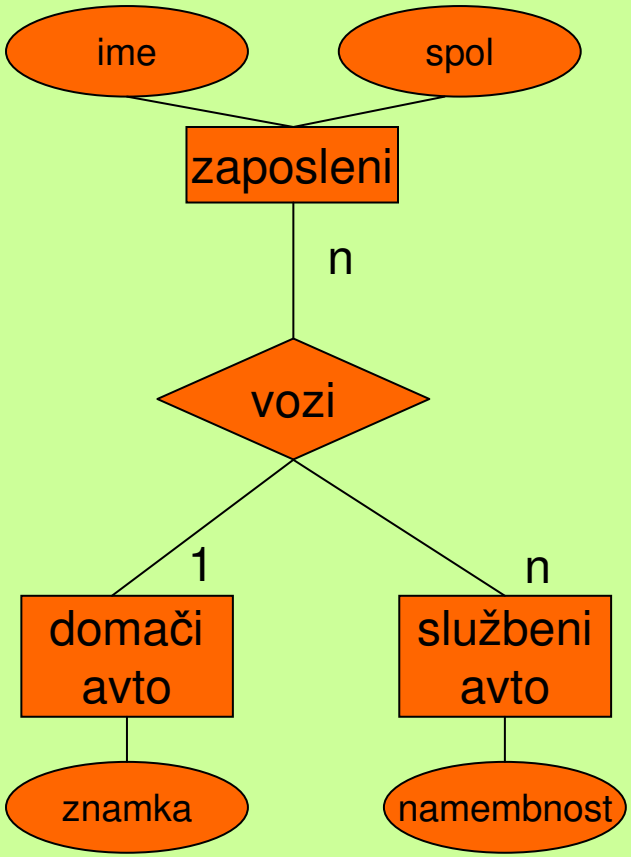
- globalni model
- konceptualni model
- logični model (relacijski, objektni, mrežni…)
- fizični model (dejanska podatkovna baza)

Konceptualni model

Konceptualni model temelji na ***E-R*** (*entiteta-razmerje*) modelu.



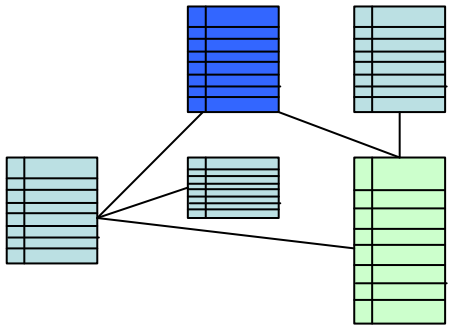
Primer:

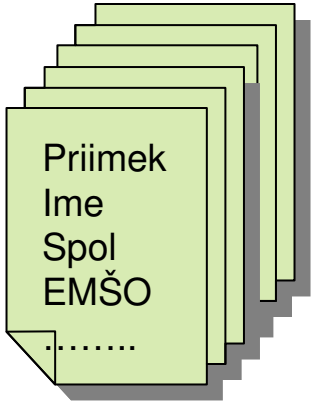


Zaposleni se vozi z enim zasebnim avtom.
Zasebni avto uporablja več zaposlenih.
Več službenih avtov vozi več zaposlenih.

Relacijski podatkovni model

- **osnova** relacijskega podatkovnega modela so **matematične relacije** izvedene **s tabelami**





podatek entitete

polje

lastnosti entitet

zapis

Startna_stevilka	Ime	Priimek	Rojstni_datum	Moski	Panoga
1	Igor	Pavlinski	11.10.1979	☑	Drsanje
2	Larisa	Ostanjuk	6.6.1972	☐	Smučarski te
3	Josef	Wallinger	4.10.1970	☑	Atletika
4	Mojca	Mihelič	18.2.1974	☐	Smučarski te
5	Greta	Hogljund	13.8.1982	☐	Atletika
6	Uroš	Matevžič	1.9.1979	☑	Rokomet
7	Jack	Brown	22.8.1976	☑	Atletika
8	Ho	Li Jin	9.8.1985	☐	Gimnastika
9	Ji	Ma Jing	14.6.1988	☑	Gimnastika
10	George	Kipkemboi	12.2.1982	☑	Atletika
11	Jonas	Svensson	8.11.1982	☐	Hokej na ledu

Tabele in tipi podatkov

- Tabela je osnovni element relacijske baze podatkov.
- Vrstico tabele imenujemo zapis.
- Čelna vrstica v tabeli pojasnjuje oz. opisuje posamezne podatke v zapisu.
- Zapis je sestavljen iz polj. Vsako polje hrani samo en podatek.
- Vrstni red polj je v vseh zapisih enak. Vsako polje v tabeli ima **svoj tip podatkov**.

Osnovni tipi podatkov

- besedilo (text, string)
- število (number)
- datum (date)
- logični tip (logical, boolean)

Startna_stevilka	Ime	Priimek	Rojstni_datum	Moski	Panoga
1	Igor	Pavlinski	11.10.1979	☑	Drsanje
2	Larisa	Ostanjuk	6.6.1972	☐	Smučarski te
3	Josef	Wallinger	4.10.1970	☑	Atletika
4	Mojca	Mihelič	18.2.1974	☐	Smučarski te
5	Greta	Hogljund	13.8.1982	☐	Atletika
6	Uroš	Matevžič	1.9.1979	☑	Rokomet
7	Jack	Brown	22.8.1976	☑	Atletika
8	Ho	Li Jin	9.8.1985	☐	Gimnastika
9	Ji	Ma Jing	14.6.1988	☑	Gimnastika
10	George	Kipkemboi	12.2.1982	☑	Atletika
11	Jonas	Svensson	8.11.1982	☐	Hokej na ledu

Odvisno od podatkovne baze, so podatki v njej lahko tudi slika, zvok, daljše besedilo (*memo*)…

Ključ

Ključ je polje (ali skupina polj), ki enoznačno in neponovljivo določa zapis v tabeli. Vsaka entiteta ima enega ali več ključev.

Primarni ključ je polje, ki povsem enolično določa zapis.

Primer:

Model	…….	Znamka
Yaris		Toyota
Punto		Fiat
Avensis		Toyota

Znamka	……..	Država
Toyota		Japonska
Fiat		Italija
Mazda		Japonska

Urejenost tabele

- Tabela je **neurejena**, če ni pravila, ki bi urejal zapise v njej.

Ime	Priimek	Točke
Tilen	Puh	35
Manca	Cerar	28
Andrej	Mežan	29
Anja	Cerar	40

- Tabela je **urejena**, če so zapisi v njej urejeni po enem ali več poljih zapisa.

Ime	Priimek	Točke
Anja	Cerar	40
Manca	Cerar	28
Andrej	Mežan	29
Tilen	Puh	35

Indeksiranje

Indeksiranje je postopek s katerim uredimo zapise v tabeli v želenem vrstnem redu. Rezultat indeksiranja je nova, t.i. indeksna tabela, v kateri je manj podatkov kot v prvotni tabeli.

Neka tabela ima lahko več indeksnih tabel.

ID	Znamka	Model	Drzava	Ser_stevilka	Leto_izd
1	Toyota	Avensis	Japonska	223AA7721	2002
2	Audi	A3	Nemčija	A11BTR1C	2003
3	BMW	750	Nemčija	AA3633OBA2	2001
4	Toyota	Avensis	Japonska	222AA4002	2004
5	Fiat	Punto	Italija	FPAAA56X1	1999

ID	Leto_izd
5	1999
3	2001
1	2002
2	2003
4	2004

indeksna datoteka, ki zapise v tabeli razvršča glede na leto izdelave

ID	Drzava
5	Italija
1	Japonska
4	Japonska
2	Nemčija
3	Nemčija

indeksna datoteka, ki zapise v tabeli razvršča glede na državo, ki proizvaja avtomobil

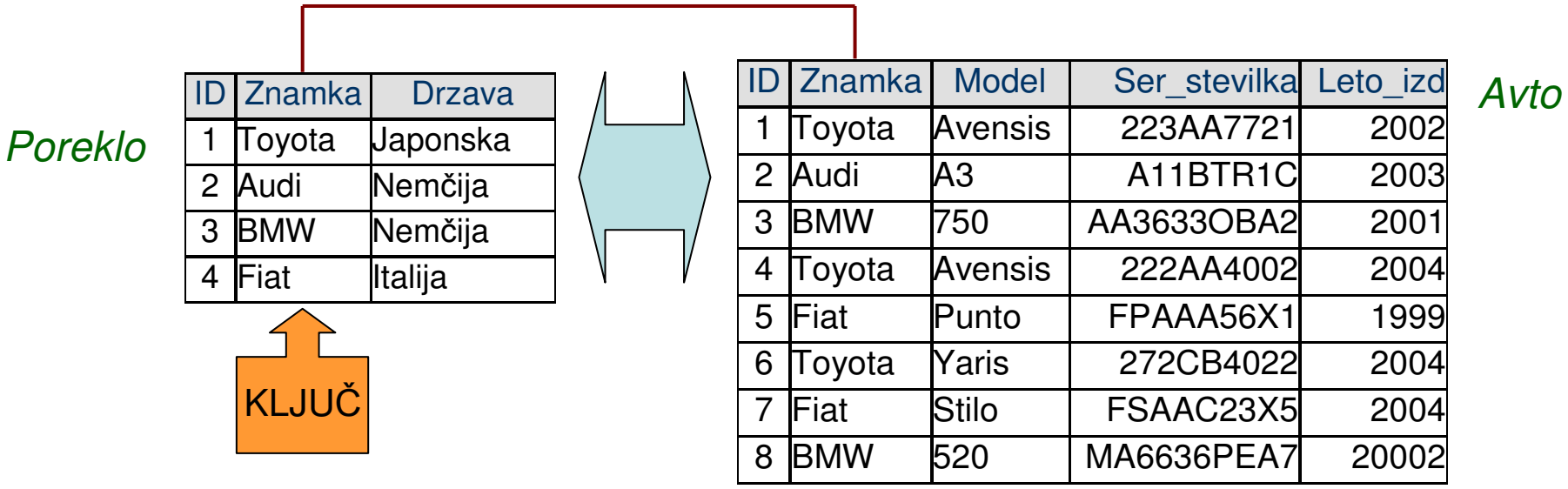
Cilj indeksiranja je hitrejši dostop do zapisov v tabeli.

Primer: *če tabela ne bi bila indeksirana, potem bi poizvedovanje "najdi avtomobile proizvedene v Nemčiji" terjalo linearno preiskovanje od prvega do zadnjega zapisa…*

Povezave med tabelami

ID	Znamka	Model	Drzava	Ser_stevilka	Leto_izd
1	Toyota	Avensis	Japonska	223AA7721	2002
2	Audi	A3	Nemčija	A11BTR1C	2003
3	BMW	750	Nemčija	AA3633OBA2	2001
4	Toyota	Avensis	Japonska	222AA4002	2004
5	Fiat	Punto	Italija	FPAAA56X1	1999
6	Toyota	Yaris	Japonska	272CB4022	2004
7	Fiat	Stilo	Italija	FSAAC23X5	2004
8	BMW	520	Nemčija	MA6636PEA7	2002

Mnogo učinkoviteje je podatke nad isto entitetno množico združevati v ločenih tabelah. Nato je potrebno zapise med seboj le še ustrezno povezati.



SQL in opravila v podatkovnih bazah

Podatke v tabelah ažuriramo s pomočjo povpraševalnih jezikov.

SQL (*Structured Query Language*) – standardiziran in strukturiran relacijski povpraševalni (poizvedovalni) jezik, ki omogoča:

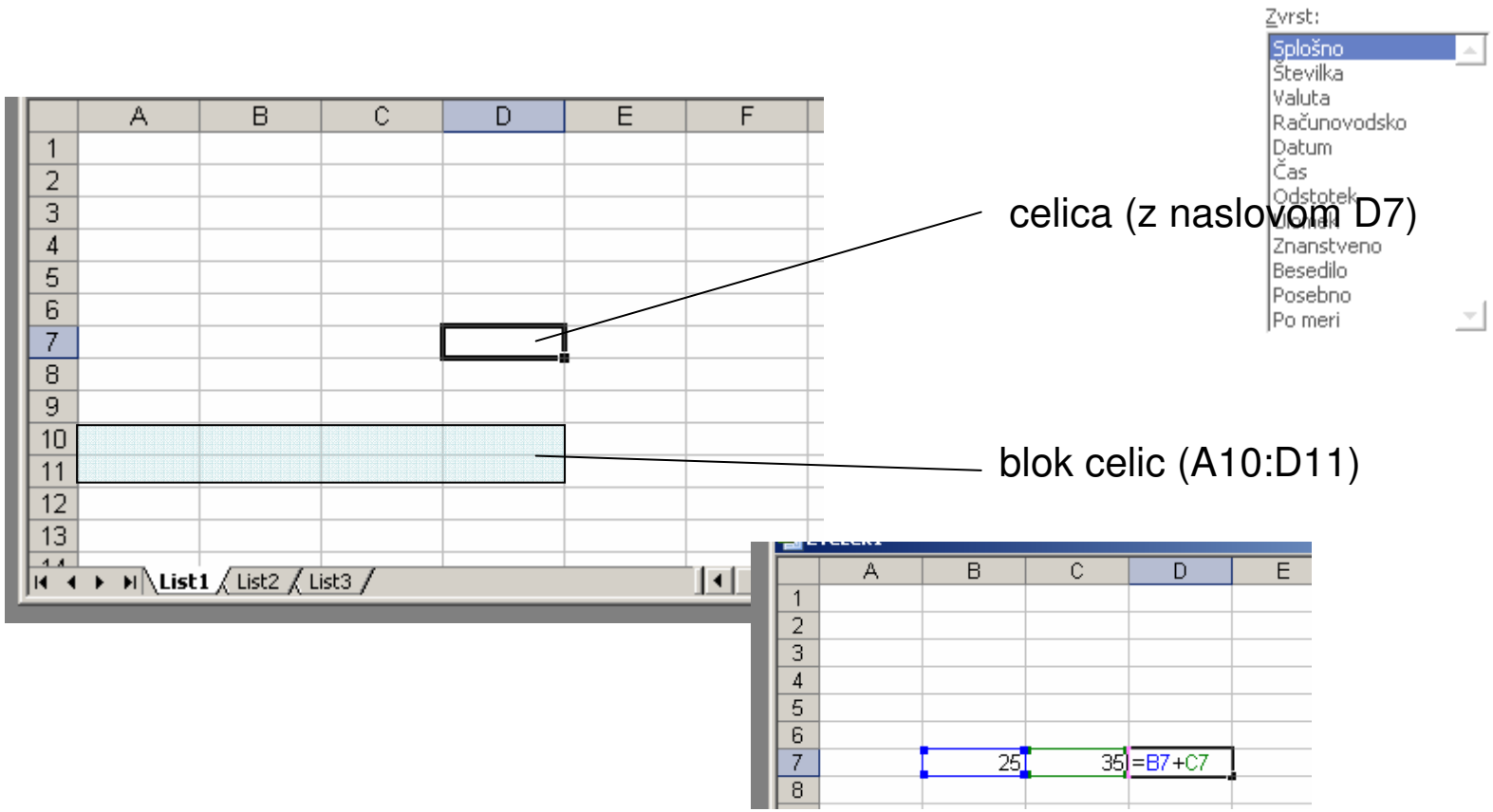
- povpraševanje
- osnovne operacije posodabljanja (dodajanje, brisanje, spreminjanje podatkov in zapisov)
- nadzor (*če … potem … sicer*)

Primer:

```
SELECT Avto.Znamka, Avto.Leto_izd, Poreklo.Drzava
FROM Poreklo, Avto
WHERE (Avto.Znamka = Poreklo.Znamka) AND (Avto.Leto_izd > 2002)
ORDER BY Avto.Leto_izd
```

Preglednice

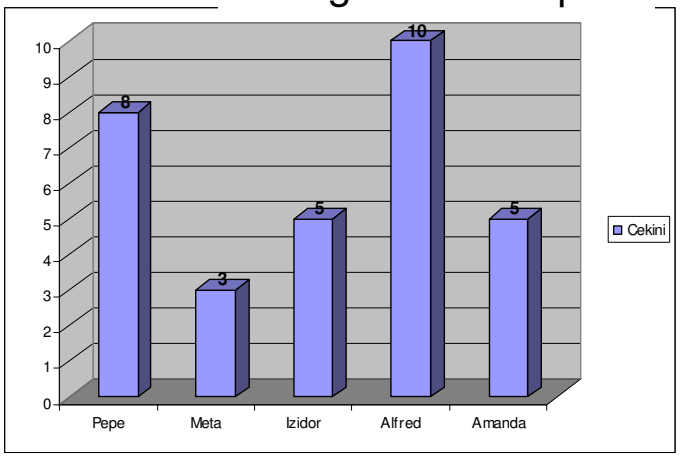
- programi za elektronske preglednice: *MS Excel, Quattro Pro, OO.o Calc…*
- namenjene preračunavanje tabelaričnih podatkov, risanju grafikonov, oblikovanju preprostih podatkovnih baz, *"kaj-če"* analizi…



Grafikon

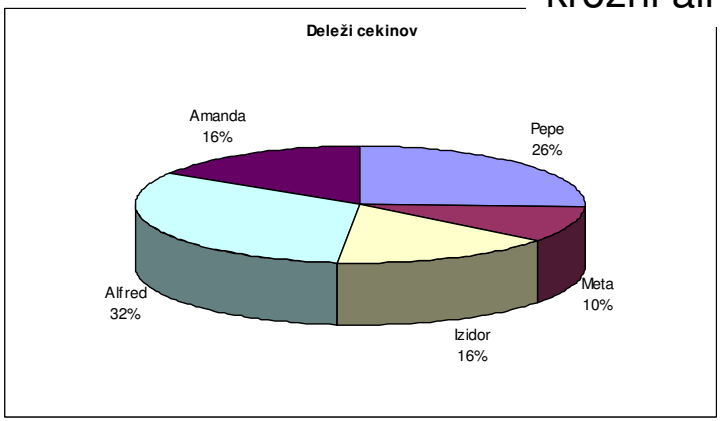
- namenjen grafični predstavitvi podatkov ⇨ razumljivost, preglednost
- vsak grafikon ni primeren za prikaz vseh podatkov

histogram ali stolpčni

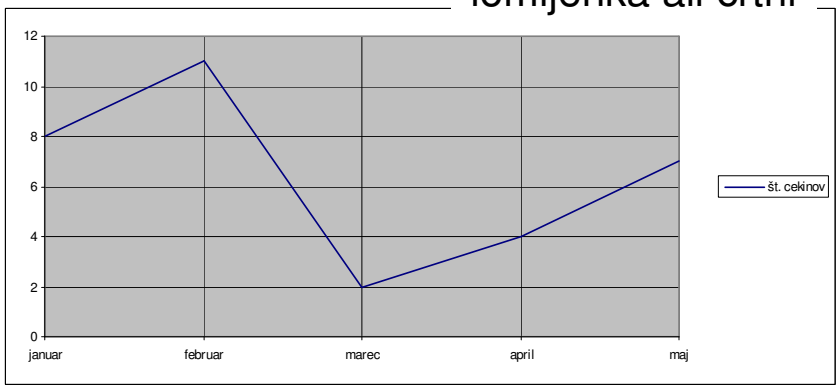


Vrste grafikonov:

krožni ali tortni



lomljenka ali črtni

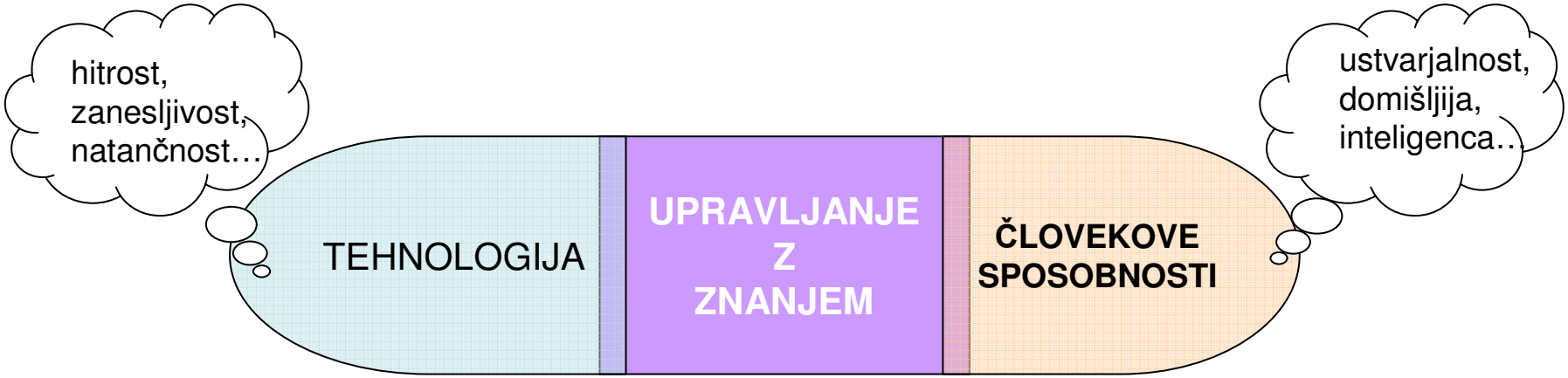


tehnologija znanj

Delo s podatki

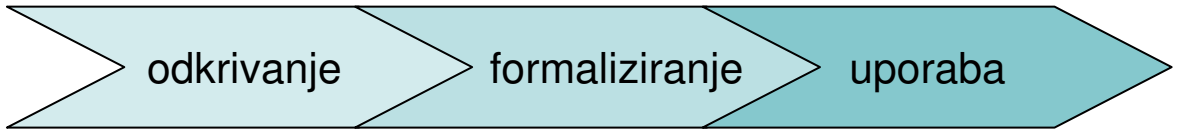
proces upravljanja znanja, tehnologija znanj
umetna inteligenca
večparametrsko odločanje in odločitveni modeli

Upravljanje z znanjem



- Upravljanje znanja z metodami in tehnikami obdelave podatkov odpravlja pomanjkljivosti naših miselnih procesov.
- Uporaba na področjih odkrivanja, zbiranja, razpolaganja, dostopanja in uporabe znanja.
- **Znanje**
  - *deklarativno znanje* (opazovanje realnosti, dogovori, dejstva, pravila)
  - *proceduralno znanje* (zmožnost opravljanja postopkov)
  - *strateško znanje* (sprejemanje odločitev katero znanje uporabiti)

Proces upravljanja znanja



Tehnologija znanja je vsaka tehnologija, ki jo uporabljamo pri upravljanju z znanjem.

Najpogostejše tehnologije znanj:

- tehnologije pisarniškega poslovanja (besedila, slike, preglednice…)
- tehnologije skupinskega odločanja (el. pošta, skupinski koledar…)
- tehnologije za iskanje znanja in upravljanja z njim (arhiviranje, iskanje podatkov…)
- tehnologije umetne inteligence (procesiranje naravnega jezika, ekspertni sistemi, robotika…)

Umetna inteligenca

- razvoj metod in tehnik za inteligentno reševanje zapletenih problemov, ki so težko rešljivi s klasičnimi metodami

- glavna področja delovanja umetne inteligence so:
    - računalniško zaznavanje
    - strojno učenje
    - predstavitev znanja
    - ekspertni sistemi
    - igranje iger
    - robotika
    - hevristično* reševanje problemov

* hevristika – način reševanja problemov na osnovi neeksaktnih postopkov za katere obstaja upanje, da bodo uspešno delovali (npr. težki kombinatorični problemi)

Računalniško zaznavanje

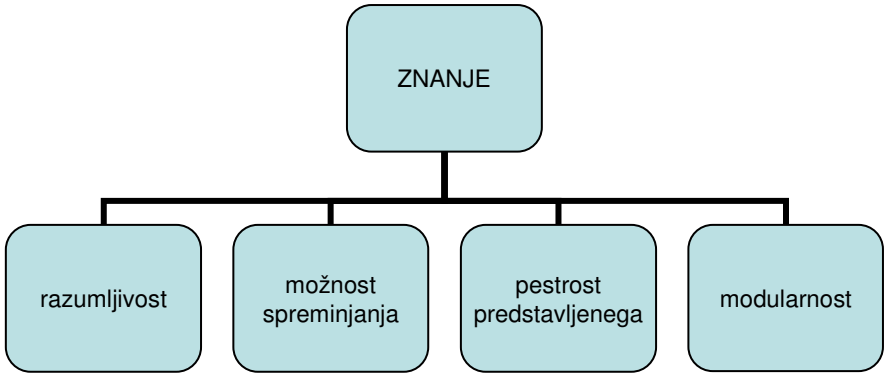
- zaznavanje realnosti je za tehnologijo zapleteno in zahtevno
- *računalniški vid* (npr. prepoznavanje predmetov, orientacija v prostoru)
- *obdelava naravnega jezika* (npr. zaznavanje in razumevanje govora)

Strojno učenje

- rezultat strojnega učenja je baza znanja, ki se avtomatsko dopolnjuje

Predstavitev znanja

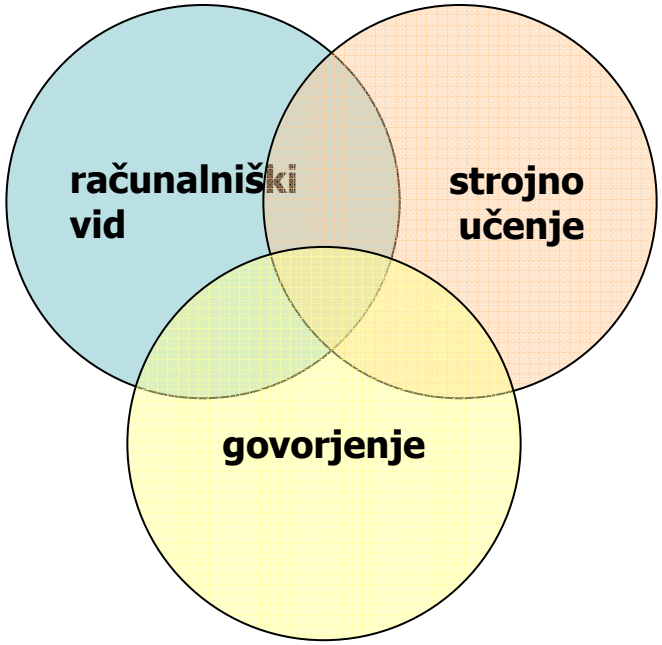
Znanje, ki ga želimo uporabljati v različnih sistemih umetne inteligence, mora biti ustrezno predstavljeno.



- *predstavitev s pravili* (podatki in pravila, ki opisujejo njihovo odvisnost)
- *predstavitev s semantičnimi mrežami* (vozlišča v grafikonu predstavljajo entitete, povezave pa relacije med entitetami)

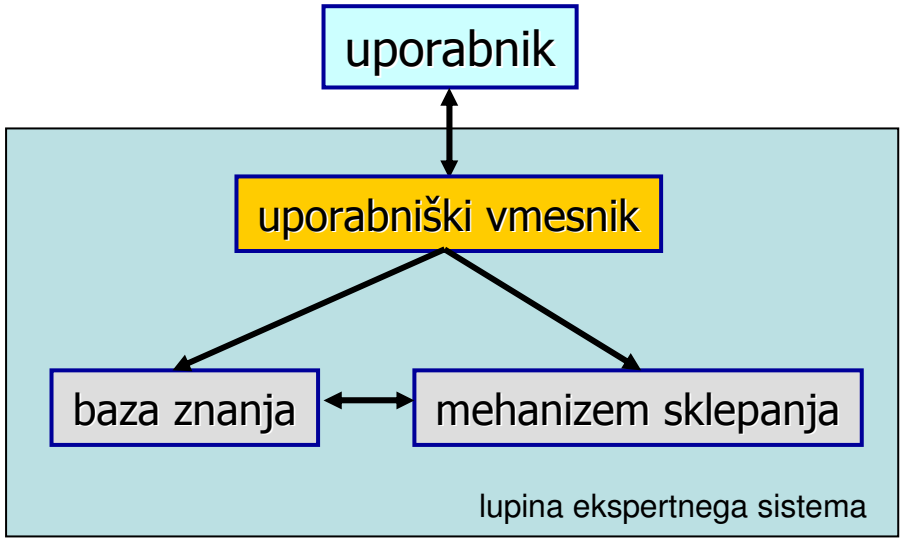
Robotika

področje umetne inteligence, ki se ukvarja z upravljanjem robotov



Ekspertni sistemi

- **Ekspertni sistem** je sistem, ki temelji na ozko specializiranem znanju, namenjen reševanju specifičnih problemov iz realnega sveta.
- Ekspertni sistem vsebuje
  - bazo znanja
  - mehanizem sklepanja
  - uporabniški vmesnik



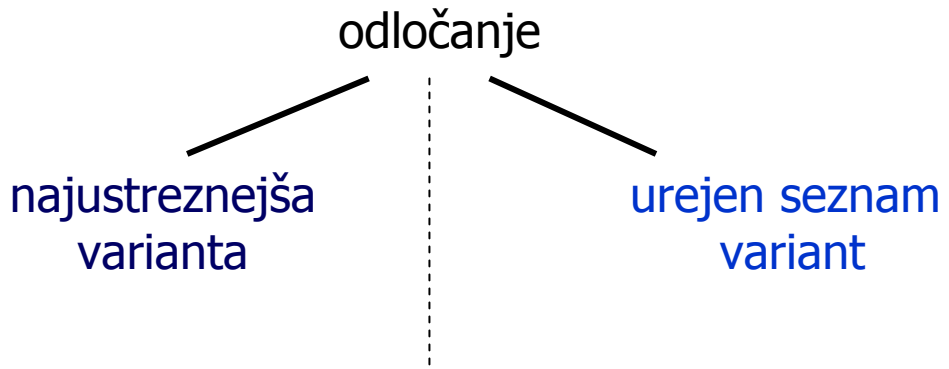
- **Lupina ekspertnega sistema** je rač. program katerega baza znanja je prazna, ima pa vgrajen mehanizem sklepanja in uporabniški vmesnik

Proces odločanja

- “prava odločitev” – najugodnejše posledice
- iz množice variant izbrati najboljšo
- **odločanje**
  - intuitivno, racionalno
  - individualno, skupinsko
- **težave pri odločanju**: nedosegljivost podatkov in nepoznavanje dejavnikov, nedoločenost variant…

Odločanje in odločitveni modeli

Odločanje je proces, v katerem med več variantami (t.j. entitetami) izberemo tisto, ki nam najbolj ustreza.



težave v procesu odločanja

- veliko število slabo opredeljenih variant in dejavnikov,
- nepopolno poznavanje odločitvenega problema,
- omejen čas za izpeljavo odločitvenega problema…

- spreminjanje odločitvenih dejavnikov
- majhne razlike, ki se seštevajo
- dve varianti nista nikoli popolnoma enaki

Sistemi za podporo odločanju

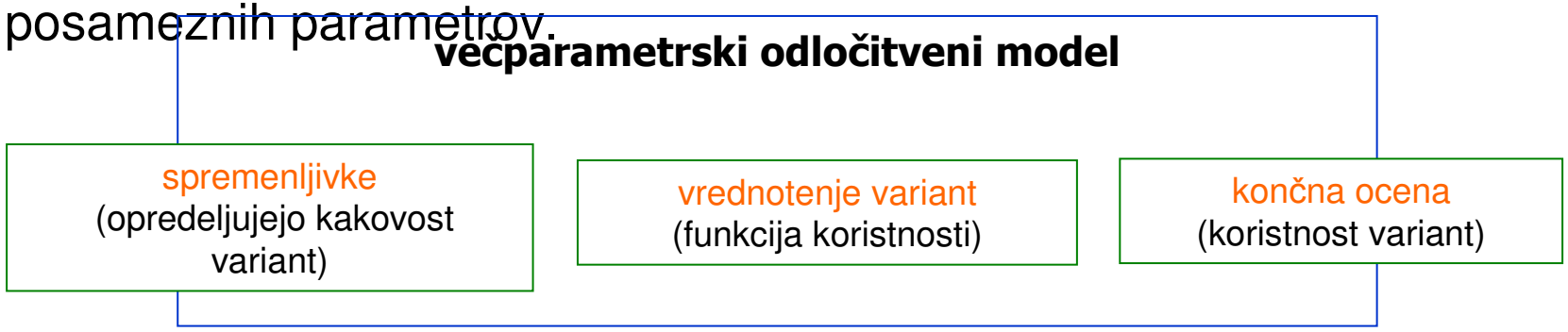
- so metode in rač. programi, ki omogočajo
  - preseči človekove omejitve pri procesiranju informacij
  - sistematizirano odločanje

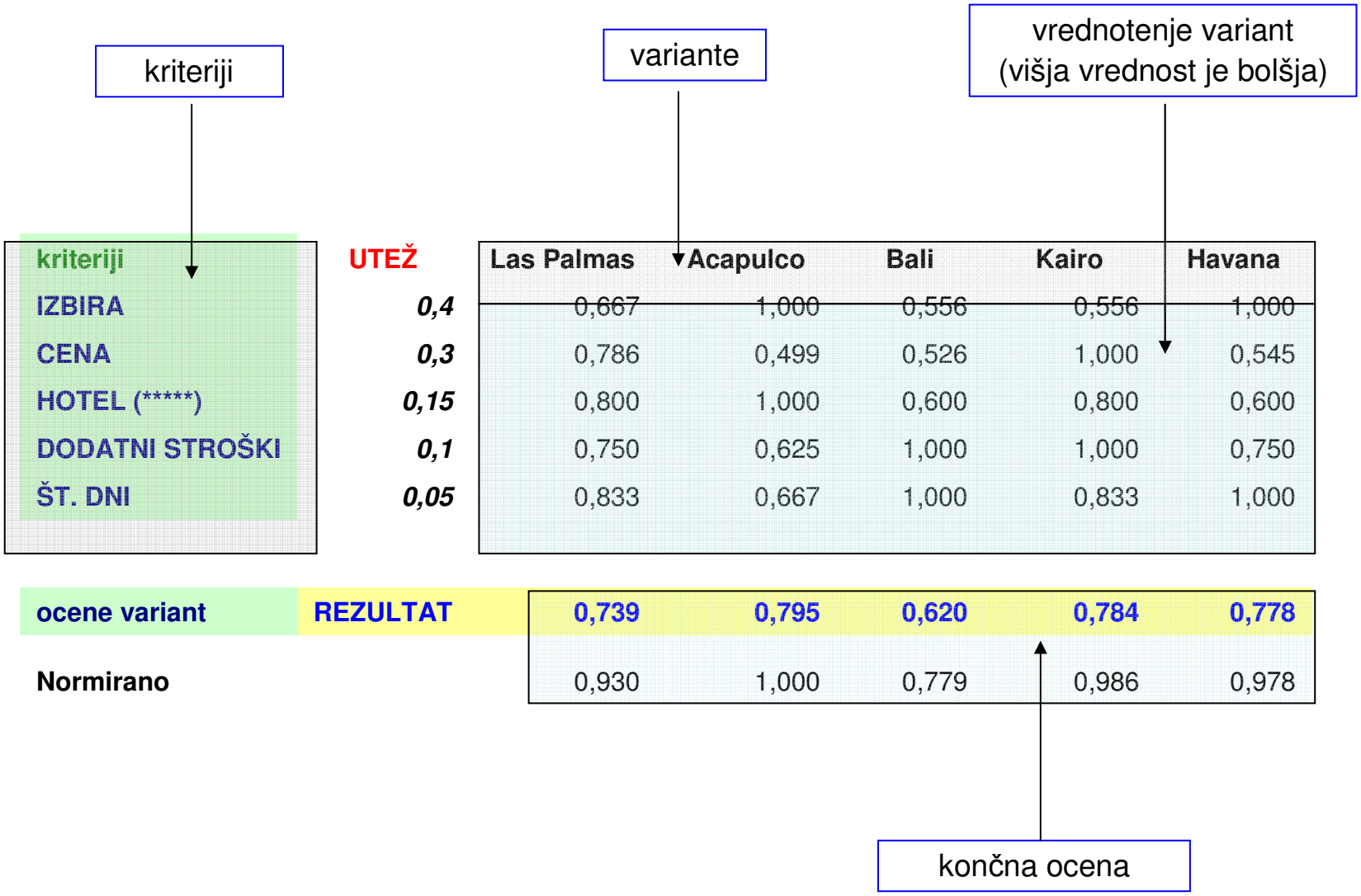
- so lahko:
  - **abacon - ročne metode** (npr. papir in svinčnik, tabela)
  - **elektronske preglednice** (npr. *Excel*)
  - **lupina ekspertnega sistema** za gradnjo odločitvenega modela (npr. *DEX*)

Večparametrsko odločanje

- Kadar je parametrov, ki vplivajo na odločitev več, govorimo o **večparametrskem odločanju.**

- **Funkcija koristnosti** je predpis, ki združuje vrednosti (“koristi”) posameznih parametrov.




kriteriji
variante
vrednotenje variant
(višja vrednost je bolšja)
kriteriji
UTEŽ
Las Palmas
Acapulco
Bali
Kairo
Havana
IZBIRA
0,4
0,667
1,000
0,556
0,556
1,000
CENA
0,3
0,786
0,499
0,526
1,000
0,545
HOTEL (*****)
0,15
0,800
1,000
0,600
0,800
0,600
DODATNI STROŠKI
0,1
0,750
0,625
1,000
1,000
0,750
ŠT. DNI
0,05
0,833
0,667
1,000
0,833
1,000
ocene variant
REZULTAT
0,739
0,795
0,620
0,784
0,778
Normirano
0,930
1,000
0,779
0,986
0,978
končna ocena

Faze odločitvenega procesa…

- **opredelitev problema** – opis problema in ciljev, oblikovanje odločitvene skupine ter odgovornosti za odločitev
- **opredelitev kriterijev** – kriteriji vrednotijo variante, kateri so bolj (manj) pomembni kriteriji…
  - *spisek kriterijev* oblikujemo v *drevo*, ki kriterije združuje na podlagi vsebinskih povezav in določimo *zalogo vrednosti* posameznim kriterijem
- **opredelitev funkcije koristnosti** – meritev koristnosti posameznih in/ali združenih kriterijev s katero določamo koristnost variant na osnovi posameznih parametrov $\Rightarrow$ ocena variant

…faze odločitvenega procesa

- **opis variant** – variante opišemo z vrednostmi posameznih kriterijev

Varianta	Gimnastika	Ples	Atletika	Tenis	Smučanje
otrok	velik	zmeren	velik	slab	zmeren
starši	zmeren	zmeren	velik	velik	slab
prijatelji	je	je	ni	ni	je
predispozicije	srednja	majhna	velika	velika	srednja
obremenjenost otroka	ne poveča	ne poveča	zmerno pove	poveča	poveča
popularnost	majhna	majhna	srednja	srednja	velika
perspektivnost	srednja	majhna	srednja	majhna	velika
obremenjenost staršev	mala	mala	srednja	velika	velika
pripomočki	manj potrebni	nepotrebni	manj potrebn	potrebni	potrebni
Samostojna vadba	redko	vedno	pogosto	redko	nikoli

- **vrednotenje variant** – določanje končne ocene, iskanje variante z najvišjo oceno in analiza variant
  - zakaj je končna ocena takšna, kot je?
  - kateri kriteriji so najbolj odločali pri oceni?
  - ali je odločitev v skladu s pričakovanji? ali odstopa?
  - ali je funkcija koristnosti ustrezna?
- **odločitev:** razlogi in utemeljitev, *kaj-če* analiza

programiranje

Delo s podatki

osnove programiranja
programski jeziki
algoritem
strukturirano, objektno in dogodkovno programiranje
programski jezik *Python*

Uvod

- **program** – zaporedje ukazov in podatkov, ki jih računalnik izvede samostojno in pri tem reši določen problem;
- programi
  - že izdelani (izdelale npr. programske hiše…),
  - programi za reševanje specifičnih problemov (izdelamo npr. sami)

- **programiranje** – vsa dela, ki se nanašajo na izdelavo programa: od analize problema do implementacije* delujočega programa;
- opredelitev problema predstavlja opis tega, kar je potrebno narediti, da bi rešili problem

implementacija – izvršitev, izvedba

Algoritem

- **algoritem** je zaporedje navodil, ki razgrajuje problem na preprostejše probleme, te pa lahko rešimo z zaporedjem preprostih ukazov ⇨ rešitev problema

  Algoritem je postopek, zaporedje definiranih opravil in ukazov, ki zagotavljajo rešitev problema v končnem številu korakov.
- primeri:
  - *hornerjev algoritem* za deljenje dveh polinomov,
  - *evklidov algoritem* za iskanje največjega skupnega delitelja dveh števil,
  - algoritem za uporabo mikrovalovne pečice;
- Kako podrobno razčlenimo algoritem je odvisno od tistega, ki bo algoritem uporabljal.
- Pogoji, ki jim mora algoritem zadoščati: ***zaustavljiv***, ***jasen*** *(zapisan z elementarnimi navodili),* ***nedvoumen****;*

Primer algoritma

Evklidov algoritem za iskanje največjega skupnega delitelja dveh števil (prednost Evklidovega postopka je, da ni potrebo razcepiti števil).
Izračunajmo največji skupni delitelj števil a in b.

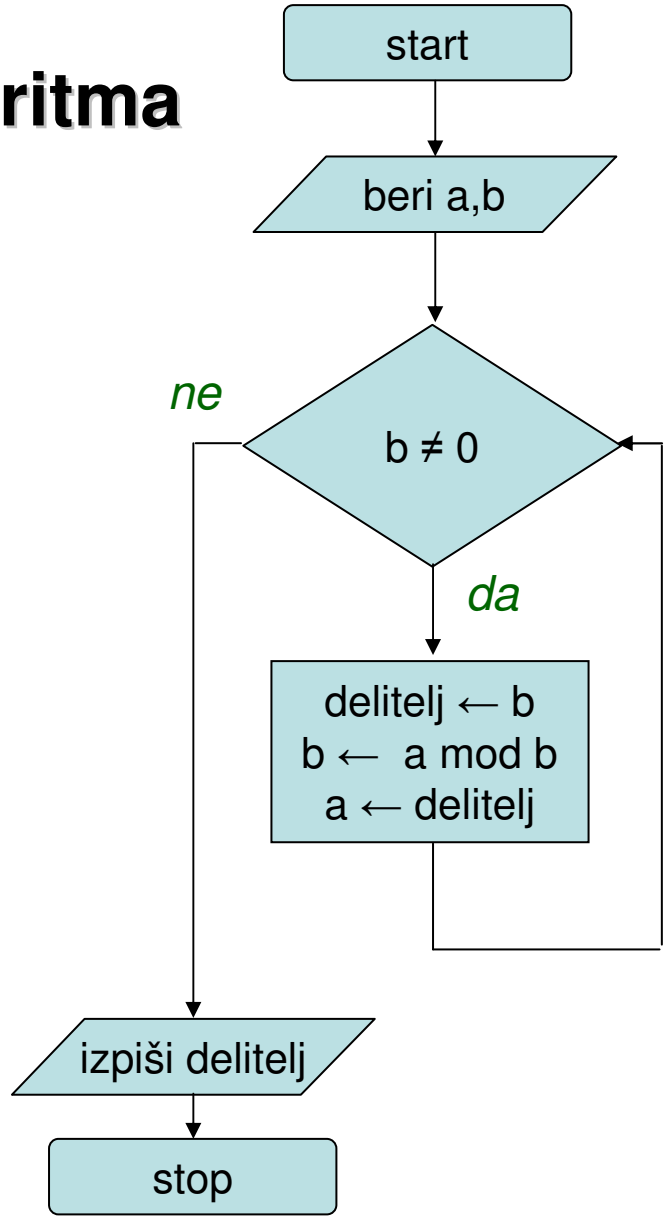
Ponavljaj dokler b ≠ 0
    delitelj ← b
    b ← celoštevilčni ostanek pri deljenju a z b
    a ← delitelj
Izpiši delitelj
Končaj

Po evklidovem algoritmu izračunaj največji skupni delitelj števil 75 in 30.

Zapis algoritma

- slikovni zapis (npr. s piktogrami)
- besedni opis
- z dogovorjenimi simboli in navodili – **diagram poteka**

Diagram poteka (*flow-chart*) je grafični prikaz struktur s poudarkom na krmiljenju in osnovnih akcijah programa.

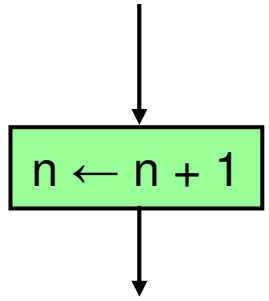


Prirejanje

Ker želimo algoritmu (programu) zagotovi univerzalnost in ker ne vemo, kako se v postopku spreminjajo vrednosti podatkov, podatke opišemo s spremenljivkami.

Vsaka spremenljivka ima

- svoje *ime*,
- v določenem trenutku svojo *vrednost*;



N ← 1 spremenljivka N dobi vrednost 1

spr ← a + b

n ← n + 1 spremenljivka n dobi za 1 večjo vrednost

Najprej se izračuna desni del prireditve, ki se nato priredi levi strani prireditvenega stavka.

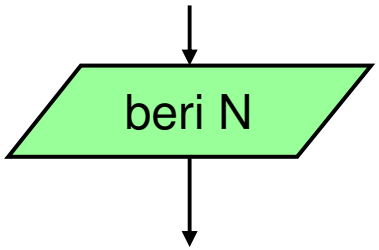
branje / izpis

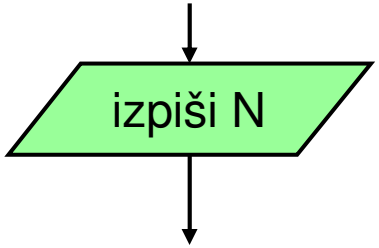
Nekaterim spremenljivkam v programu moramo posredovati vrednosti.

Branje / izpis je t.i. *vhodno / izhodni* stavek.

Vrednost spremenljivki v programu posreduje bodisi uporabnik, drug program, naprava…

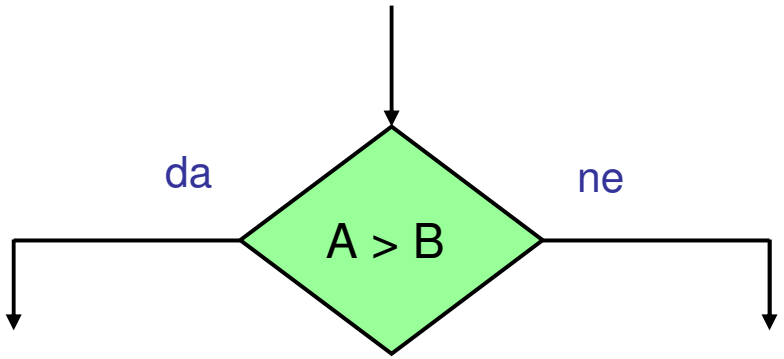
Program rezultate lahko izpiše na zaslon, tiskalnik, jih posreduje drugemu programu.





Vejitev

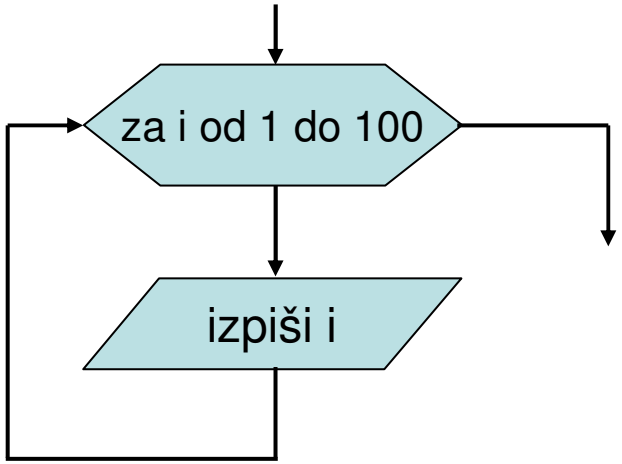
Vejitev je del algoritma, ki njegovo izvajanje cepi na dve veji.
Postopek bo zajel izvajanje tiste veje, ki ustreza vrednosti pogoja.

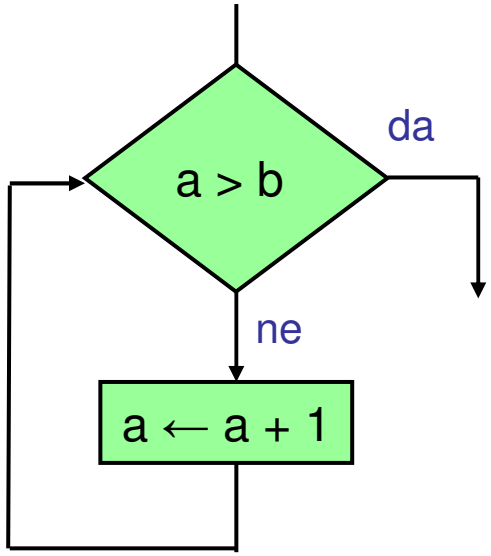


Zanka

Zanka ponazarja ponavljanje enega ali zaporedja ukazov.

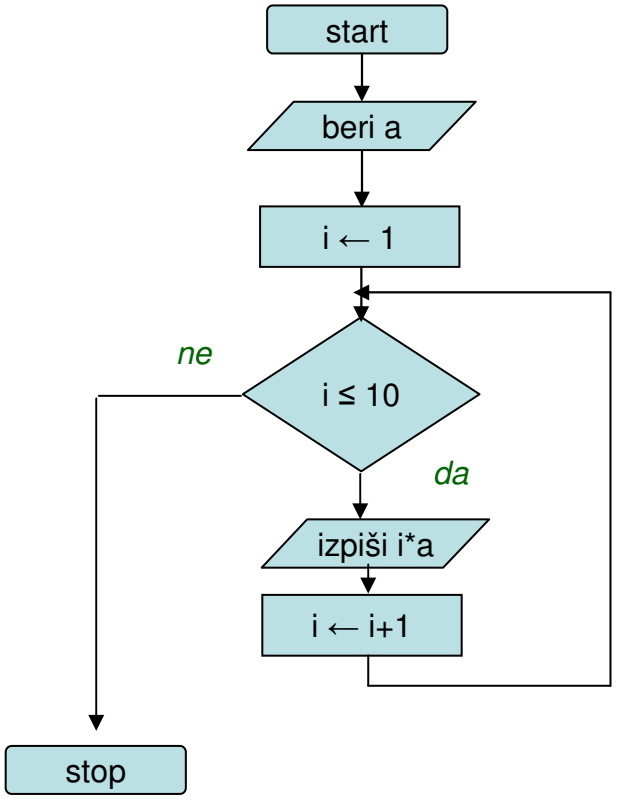
V postopku izdelave algoritma je število ponovitev lahko znano ali pa tudi ne.





Naloga

Sestavi diagram poteka za poštevanko (prvih 10 večkratnikov) prebranega števila.

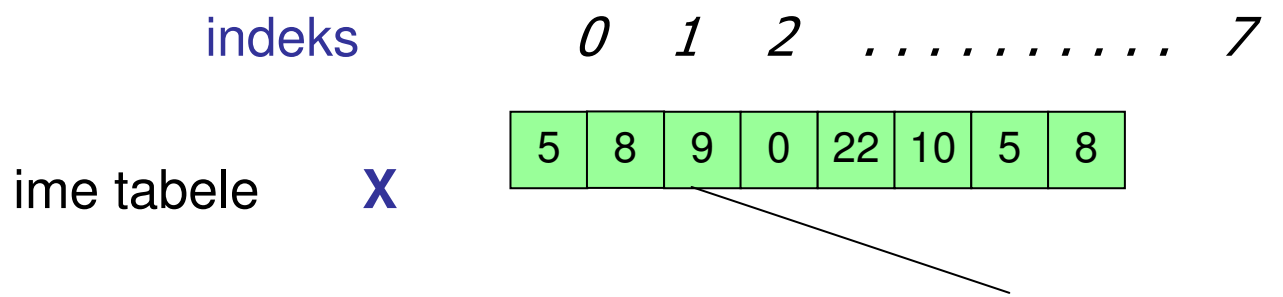


Tabelarične spremenljivke

Tabelarične spremenljivke uporabimo ko

- za rešitev problema potrebujemo zelo veliko spremenljivk,
- ne moremo predvideti števila potrebnih spremenljivk;

Tabelarične spremenljivke imajo isto ime in podatkovni tip toda drugačen indeks.



sklicevanje na to komponento oz. sprem. X(2) (včasih zapišemo tudi X_2)

V algoritmih običajno izvedemo postopek nad enim elementom tabele, nato pa le spreminjamo indeks.

Programski jeziki

Programski jeziki so umetni jeziki, ki služijo natančnemu zapisovanju računalniških programov.

Primeri prog. jezikov: *Pascal, Basic, Cobol, Prolog, Python, Java,*

Programski jeziki morajo omogočati:

- opis problema (podatki, rezultati),
- opis postopka (opis korakov, ki pripeljejo do rešitve)

zato jih delimo na

- **nepostopkovne programske jezike** – vsebujejo način za opis podatkov in relacij med njimi (npr. *Prolog*) in običajno zahtevajo le opis problema,
- **postopkovne programske jezike** – vsebujejo izrazne možnosti za opredelitev podatkov in algoritmičnih gradnikov za rešitev problema;

Programski jeziki

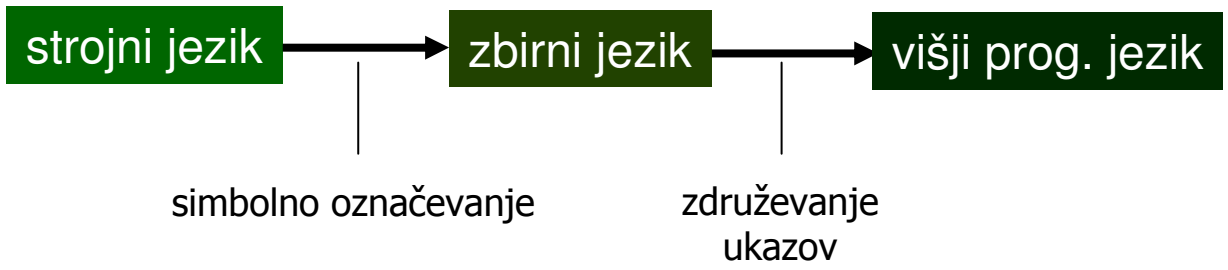
Računalnik razume edino program napisan v t.i. **strojnem jeziku**.

Primer: 11001111 11000000 00010001 strojni jezik

ADD 192 17 zbirni jezik

Slabosti strojnega jezika:

- nepregleden in težko razumljiv,
- od procesorja do procesorja različen;



Poleg višjih programskih jezikov poznamo tudi jezike t.i. *četrte generacije* in *jezike umetne inteligence*.

Prevajanje programskih jezikov

Prevajalni program (prevajalnik) prevede program zapisan v programskem jeziku v program v strojnem jeziku.

Programski jezik je določen s svojo

- **sintakso** ali jezikovno pravilnostjo - opis zgradbe jezika in s tem katero zaporedje znakov in simbolov pomeni smiseln program,
- **semantiko** t.j. opis pomena, ki nam pove, kako sintaktično pravilen program deluje oz. kaj počne;

Semantiko jezika navadno opišemo z besedami, medtem ko si pri sintaksi pomagamo s t.i. sintaksnimi diagrami.

Primer

Ukaz v prog. jeziku *Python*

```
>>> beseda = ‘Grosuplje’
>>> len(beseda)
9
```

Sintaktična napaka:

```
>>> length(beseda)
```

Semantika ukaza:

ukaz vrne število znakov spremenljivke *beseda* (9 znakov)

Prevajalniki, tolmači

Prevajalni program je lahko

- **prevajalnik**
- **tolmač**

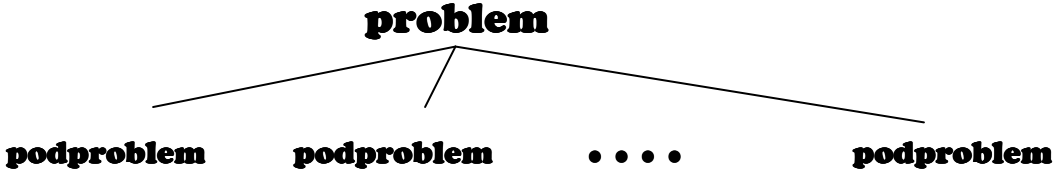
Prevajalnik (*compiler*) celoten program v višjem programskem jeziku prevede v njemu ustrezen program v strojnem jeziku.

Tolmač (*interpreter*) ukaze sproti pregleduje in tolmači v ukaze strojnega jezika.

S stališča hitrosti izvajanja programa so v prednosti programi, ki jih prevajamo.

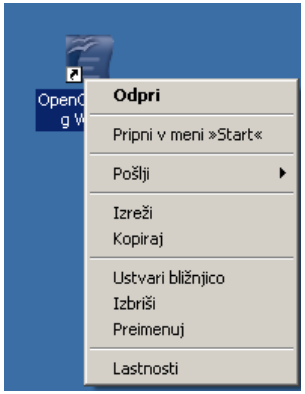
Postopki pri programiranju

- zapleteni algoritmi, preglednost, nadzor, odpravljanje napak
- **strukturirano programiranje:** postopnost razvoja rešitve



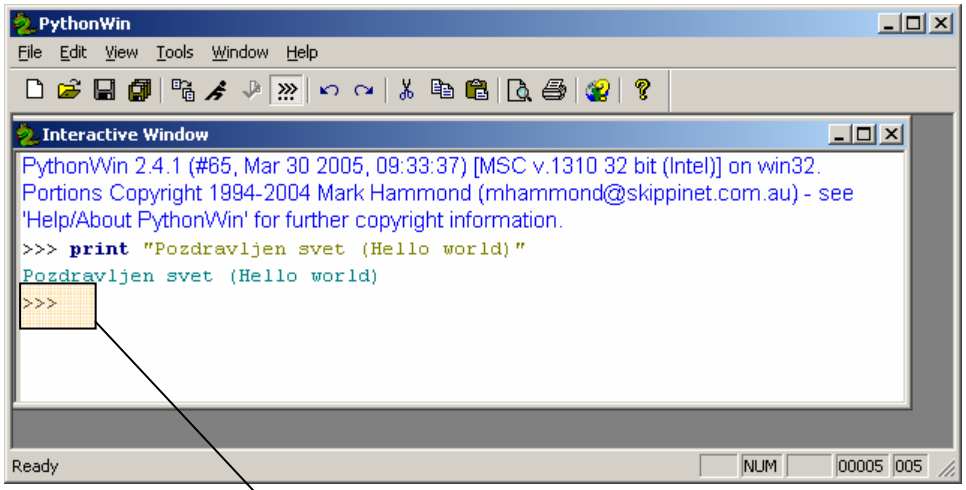
- **objektno programiranje**:

  razredi (skupek pravil) + objekti (predstavniki razredov) + metode (ustvarjanje in uporaba objektov)
- **dogodkovno programiranje:** proženje procesov, ko se zgodi ustrezen dogodek;



Programski jezik PYTHON

- višji programski jezik
- odprtokodna licenca
- interaktiven, objektno usmerjen skriptni jezik, ki se tolmači



pozivnik (*prompt) – kar vnesemo za pozivnikom je vhodni podatek*

Števila in izrazi

```
>>> 2+5
7
>>>
```

izraz - vhodni podatek

rezultat - izhodni podatek

Operatorji

- odštevanje

* množenje, ** potenca

/ deljenje

% ostanek pri celoštevilčnem deljenju, // celoštevilčni rezultat deljenja

```
>>> 5-2
3
>>> 5*2
10
>>> 15/4
3
>>> 15%4
3
>>>
```

berljivost: lahko delamo presledke…

… števila in izrazi

```
>>> 15.0/4
3.75
>>>
```

rezultat deljenja dveh celih števil
je celo število;
če je vsaj en operand realno število,
je rezultat realen;

```
>>> 5*4+2
22
>>> 5*(4+2)
30
>>> ((2+1)*(1+3))+ 2*(5+5)
32
>>>
```

Python upošteva prioriteto operatorjev

pri zapletenejših izračunih uporabimo oklepaje

Spremenljivke…

Spremenljivka je količina, ki med izvajanjem programa spreminja svojo vrednost.

Imena spremenljivk:

- se začnejo s črko (šumniki niso dovoljeni),
- lahko vsebujejo črke, števke in podčrtaje;

Primeri: a, A, x1, x2, vsota, stevec_stevk

Python razlikuje velike in male črke: **ime** in **Ime** sta različni spremenljivki !!

```
>>> prvo_stevilo = 10
>>> drugo_stevilo = 5
>>> prvo_stevilo + drugo_stevilo
15
>>>
```

```
>>> prvo_stevilo = 10
>>> drugo_stevilo = 5
>>> vsota = prvo_stevilo + drugo_stevilo
>>> vsota
15
>>> a=b=c=10
>>> a+b+c
>>>30
```

…spremenljivke

```
>>> 1a=22
Traceback (  File "<interactive input>", line 1    1a=22
                                                    ^
SyntaxError: invalid syntax
>>>
```

sintaktična napaka
(napačno ime spremenljivke)

Komentar je besedilo, ki ga tolmač razume kot pojasnilo, s katerim povečamo razumljivost ukazov. Zapišemo ga za znakom #.

```
>>> prvo_stevilo==drugo_stevilo # preverjanje enakosti dveh spremenljivk
False
>>> prvo_stevilo=5.25 # do sedaj je bila spremenljivka celoštevilčna
>>>
```

V *Python-u* lahko spremenljivka med izvajanjem programa zamenja podatkovni tip, npr. iz celega v realno število.

Niz znakov

Niz je podatkovni tip, ki vsebuje enega ali več znakov.
Da niz znakov razlikujemo od imena spremenljivke, ga pišemo med narekovaji.

```
>>> "Pepe“
'Pepe‘
>>> Ime, Priimek="Pepe“,"Pepnik“
>>> Ime+" "+Priimek
'Pepe Pepnik‘
>>>
```

prirejanje spremenljivkam lahko zapišemo tudi tako

Niz znakov je tabelarična spremenljivka: prvi znak v nizu ima indeks 0.

```
>>> niz="Poljane“
>>> niz[0]
'P‘
>>> niz[5]+niz[6]
'ne‘
>>>
```

Krmilni stavki - if

If stavek predstavlja v programu vejitev.

```
If pogoj1:
        stavki1     # stavki, ki se izvedejo, če je pogoj1 izpolnjen
elif pogoj2:
        stavki2     # stavki, ki se izvedejo, če je pogoj2 izpolnjen
else:
        stavki      # stavki, ki se izvedejo sicer
```

opcijsko (lahko dodamo tudi dodatne pogoje)

opcijsko

Zamikanje pri pisanju programa je nujno – *Python* na ta način razbere, kateri stavki se izvedejo.

```
A=input("Vnesi 1. stevilo: ")
B=input("Vnesi 2. stevilo: ")
if A>B:
        max=A
else:
        max=B
print max
```

pojavi se okence za vnos podatka

program zapisan v datoteki

Krmilni stavki – while zanka

While stavek predstavlja v programu zanko**.**

while *pogoj*:
    stavki$_1$ # stavki, ki se izvajajo dokler je pogoj izpolnjen
else:
    stavki$_2$ # stavki, ki se izvedejo na koncu zanke, ko pogoj ni več izpolnjen
        razen, če stavki$_1$ ne vsebujejo ukaza **break**

```
A=input("Vnesi 1. stevilo: ")
B=input("Vnesi 2. stevilo: ")
while not A==B:
    A=B
    B=input("Vnesi 2. stevilo: ")
print "2x zapored si vnesel ",B
```

rezultat se izpiše v interaktivnem oknu

Krmilni stavki – for zanka

Tudi **for stavek** v programu uporabljamo za ponavljanje stavkov.

for *spremenljivka* in S:
    stavki$_1$ # stavki, ki se izvajajo dokler je vrednost *spremenljivke* v obsegu tabelaričnega indeksa
else:
    stavki$_2$ # stavki, ki se izvedejo na koncu zanke razen, če stavki$_1$ ne vsebujejo ukaza **break**

Za določanje tabelarične spremenljivke S lahko uporabimo vgrajeno funkcijo *range(a,b) – vrednost spremenljivke doseže vrednosti od vključno a do vključno b-1 .*

```
>>> for spr in range(1,4):
...             print spr
...
1
2
3
>>>
```

```
>>> for spr in range(1,10,2):
...             print spr
...
1
3
5
7
9
>>>
```

korak

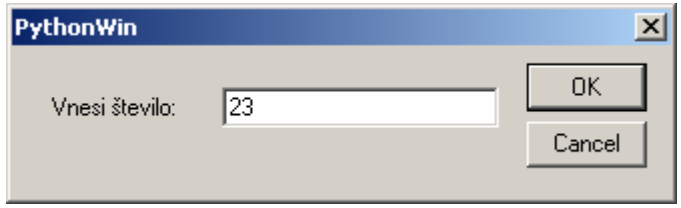
Modul

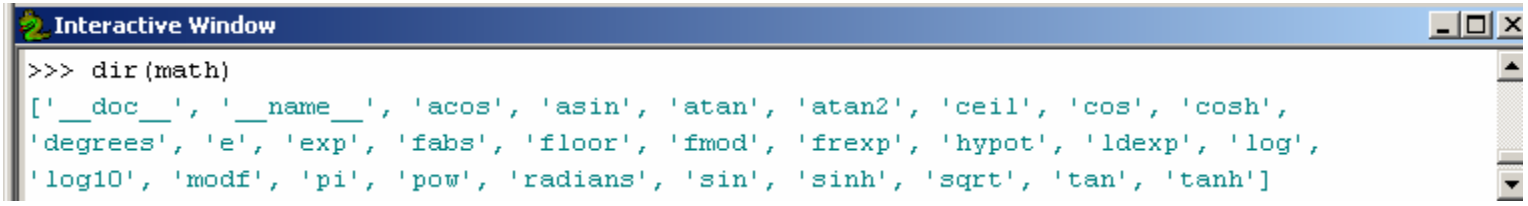
Python vsebuje knjižnico modulov, ki omogočajo uporabo različnih funkcij in podatkov, ki so združene v posameznem modulu.

Npr.: modul za matematične izračune, vhodno-izhodni modul, modul za sistemske opeacije…

vnos podatka

```
>>> s=input('Vnesi stevilo: ')
>>> import math
>>> math.sqrt(s)
4.795831523312719 1
>>>
```





Program v datoteki

- datoteko ustvarimo s *File / New – Python Script*
- datoteka ima končnico *.py*
- program zaženemo s File / Run…
- privzeto se rezultati izpisujejo v interaktivnem oknu

Primer:

```
a=2
b=10
while b >= a:
    b=b-a
print 'ostanek pri deljenju je ',b
```

Na internetu dostopna literatura

- A Byte of Python – http://www.ibiblio.org/g2swap/byteofpython/read/
  - v obliki spletnega priročnika
  - v obliki PDF dokumenta

- "uradni" priročnik
  http://www.python.org/doc/current/tut/tut.html