



# Uvod u računalno razmišljanje

# Teme

- Temelji računalnog razmišljanja
- Algoritmi – dijagram toka i pseudo kôd
- Primjena elemenata računalnog razmišljanja



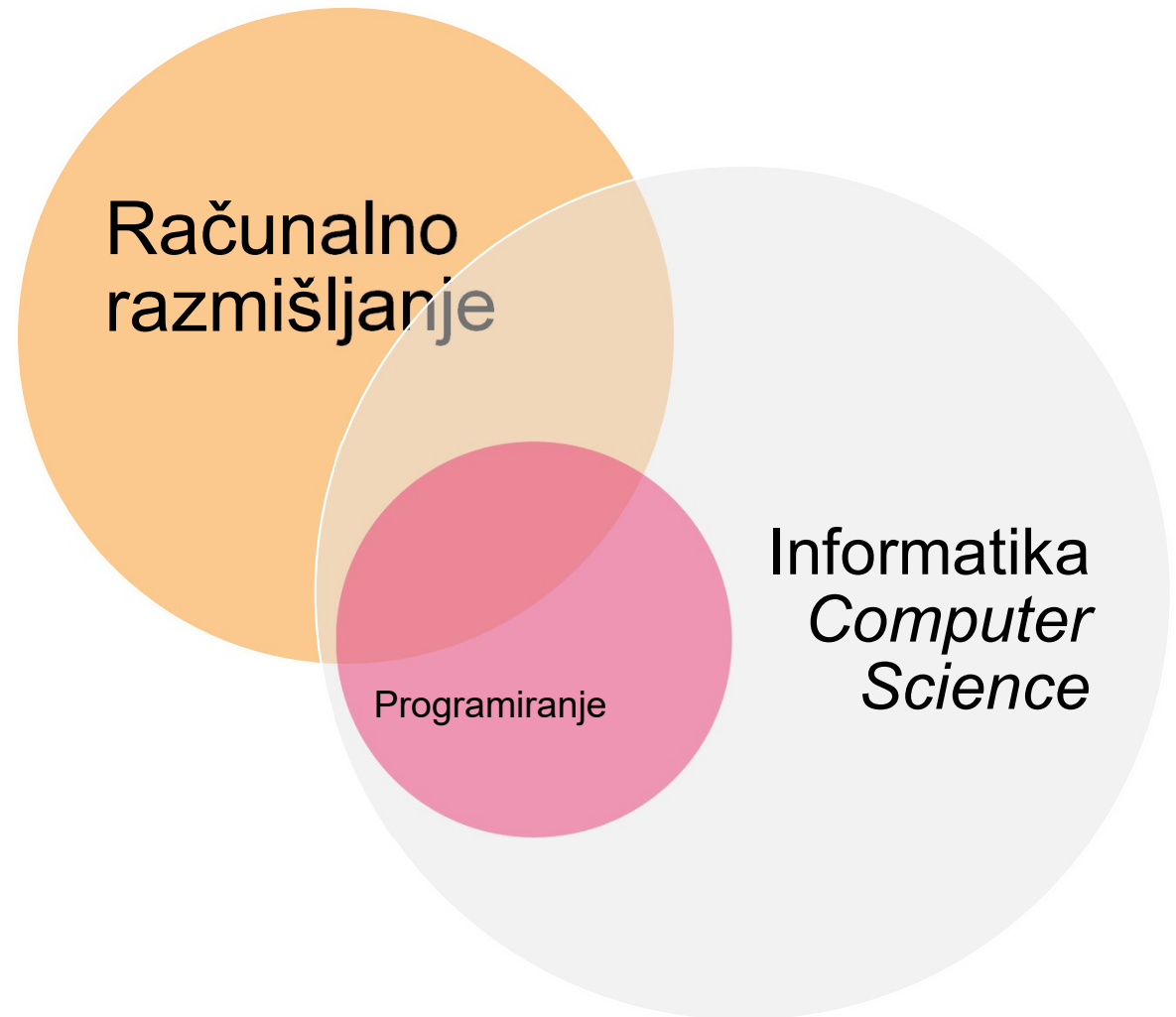
# Temelji računalnog razmišljanja

# Sadržaj

- Značajke računalnog razmišljanja
- Osnovni elementi računalnog razmišljanja
- Elementi računalnog razmišljanja kao alat u rješavanju svakodnevnih problema

# Značajke računalnog razmišljanja

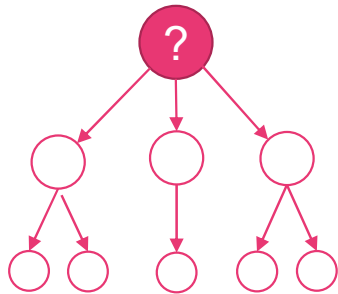
- Programiranje i kodiranje
- Informatika ili računalna znanost (Computer Science)
- Računalno razmišljanje
  - Način kako doći do mogućeg rješenja za kompleksni problem ... i kasnije predstaviti rješenje koje će razumjeti čovjek, računalno ili oboje.
- Računalno razmišljanje NIJE
  - Razmišljanje kao računalno ili robot
  - Nije programiranje



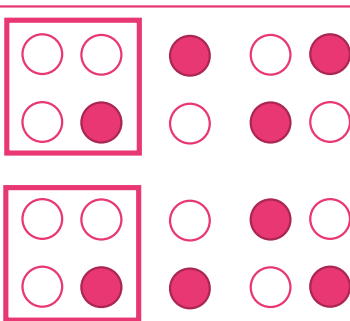
# Osnovni elementi računalnog razmišljanja

- Dekompozicija – Rastavi na manje dijelove
  - Rastavljanje kompleksnih problema na manje, lakše rješive probleme
  - „*Rastavi*” pa *vladaj*”
- Uočavanje uzoraka – Uočiti ponavljanja i sličnosti
  - Tražiti sličnosti između i unutar manjih problema
- Apstrakcija – Usredotoči se samo na bitno
  - Usredotočiti se samo na one ključne karakteristike i informacije, a sve druge zanemariti
- Algoritam – Korak-po-korak upute
  - Kreirati korak-po-korak slijed aktivnosti ili pravila koja će dovesti do rješenja problema
- Provjera kvalitete rješenja
  - Ne ulazi u elemente računalnog razmišljanja, ali je nužan obavezan korak za kvalitetno rješavanje problema
  - Provjerite je li osmišljeno rješenje zaista rješenje problema
  - Postoji li jednostavnije rješenje?

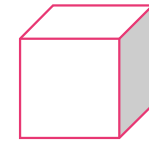
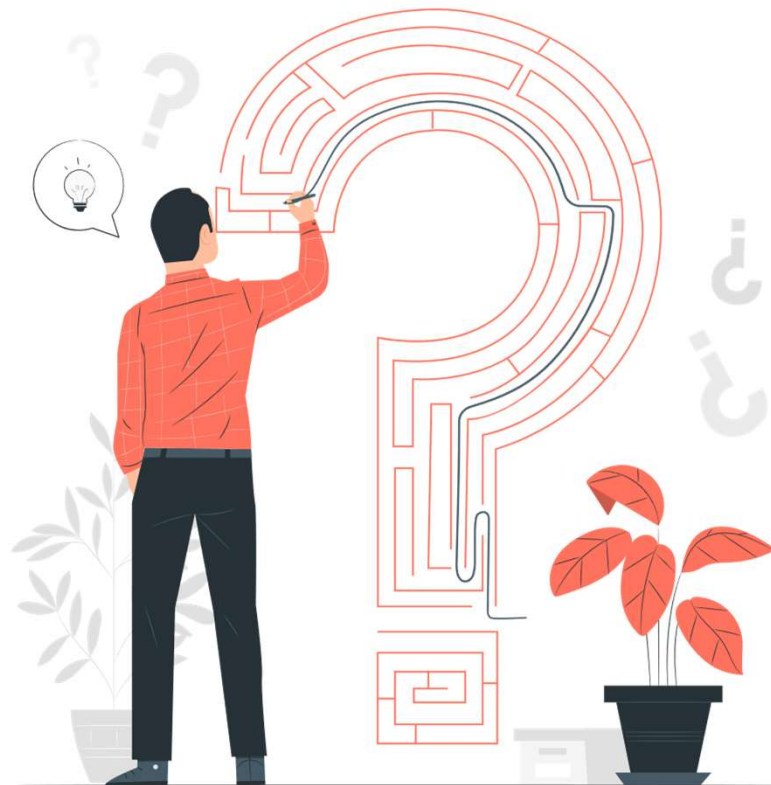
# Računalno razmišljanje



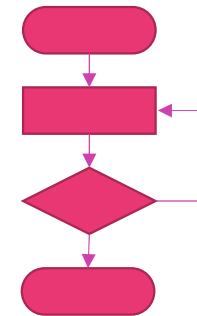
**Rastavljanje**



**Uzorak**



**Apstrakcija**



**Algoritam**

# Dekompozicija

- Jedan od četiri osnovna elementa računalnog razmišljanja.
- Predstavlja rastavljanje kompleksnih problema na manje, jednostavnije, lakše razumljive probleme. Jednostavniji problemi se onda mogu lako i jednostavno razumjeti i riješiti.
- **Koje komponente/dijelovi čine/opisuju ...?**
- Jednostavnije je fokusirati se na rješavanje manjeg, jednostavnijeg problema nego istovremeno rješavati cijeli problem odjednom.



# Dekompozicija – Važnost

- Multitasking NIJE moguć zbog načina kako radi naš mozak
  - Vrlo brzo mijenjamo fokus s jedne aktivnosti na drugu.
  - Rješavanje svih zadataka/problema odjednom je jako teško (ponekad i nemoguće)
- Razbijanjem problema na manje dijelove dobit ćemo jasniju sliku o redoslijedu ili prioritetu rješavanja manjih dijelova/zadataka kompleksnijeg problema.
- Pokušati razumjeti kako rade kompleksni sustavi je jako teško, ali ako razumijemo kako rade pojedini, manji dijelovi tog sustava, onda razumijevanje rada kompleksnog sustava u cjelini nije nikakav problem.

# Dekompozicija – PRIPREMA KOLAČA

- Što nam sve treba kako bi riješili ovaj problem?
- Sastojci
  - Ovisno o kolaču: brašno, šećer, čokolada, jaja, mlijeko ...
- Posuđe
  - Ovisno o kolaču: velika zdjela, tava, kalupi ...
- Alati / uređaji
  - Ovisno o kolaču: pećnica, hladnjak, mikser ...
- Obratite pažnju na dio „ovisno o kolaču”. Specifična pitanja kao: Koje točno sastojke, posuđe, alate, pribor ... u ovom koraku zanemarujemo. Njih ćemo koristiti kada budemo pripremali točno specifičan kolač.



# Dekompozicija – POGODI BROJ

- Zadatak: Pogodite broj između 1 i 100 u maksimalno 9 pokušaja.



# Dekompozicija – pogodi broj

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

# Dekompozicija – pogodi broj

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
<b>51</b>	<b>52</b>	<b>53</b>	<b>54</b>	<b>55</b>	<b>56</b>	<b>57</b>	<b>58</b>	<b>59</b>	<b>60</b>
<b>61</b>	<b>62</b>	<b>63</b>	<b>64</b>	<b>65</b>	<b>66</b>	<b>67</b>	<b>68</b>	<b>69</b>	<b>70</b>
<b>71</b>	<b>72</b>	<b>73</b>	<b>74</b>	<b>75</b>	<b>76</b>	<b>77</b>	<b>78</b>	<b>79</b>	<b>80</b>
<b>81</b>	<b>82</b>	<b>83</b>	<b>84</b>	<b>85</b>	<b>86</b>	<b>87</b>	<b>88</b>	<b>89</b>	<b>90</b>
<b>91</b>	<b>92</b>	<b>93</b>	<b>94</b>	<b>95</b>	<b>96</b>	<b>97</b>	<b>98</b>	<b>99</b>	<b>100</b>

# Dekompozicija – pogodi broj

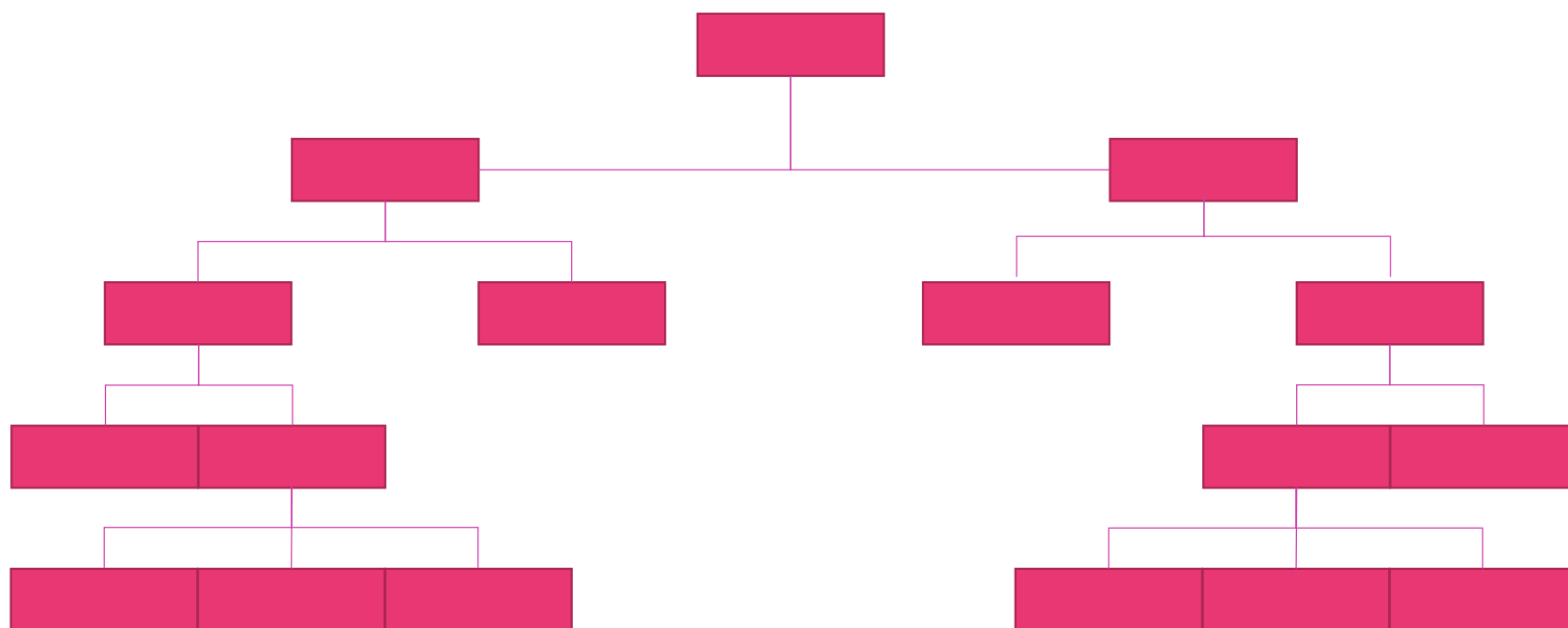
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	<b>76</b>	<b>77</b>	<b>78</b>	<b>79</b>	<b>80</b>
<b>81</b>	<b>82</b>	<b>83</b>	<b>84</b>	<b>85</b>	<b>86</b>	<b>87</b>	<b>88</b>	<b>89</b>	<b>90</b>
<b>91</b>	<b>92</b>	<b>93</b>	<b>94</b>	<b>95</b>	<b>96</b>	<b>97</b>	<b>98</b>	<b>99</b>	<b>100</b>

# Dekompozicija – PUSTI OTOK

- Kako preživjeti?
  - Hrana
    - Ribolov
    - Lov
    - Skupljanje plodova
    - Kuhanje / pečenje
  - Voda
    - Izvor
    - Pročišćavanje
  - Sklonište
    - Lokacija
    - Ugrijati se i osušiti
    - Zaštita
  - Bijeg
    - Signalna vatra za brod i avion
    - Signal za avion



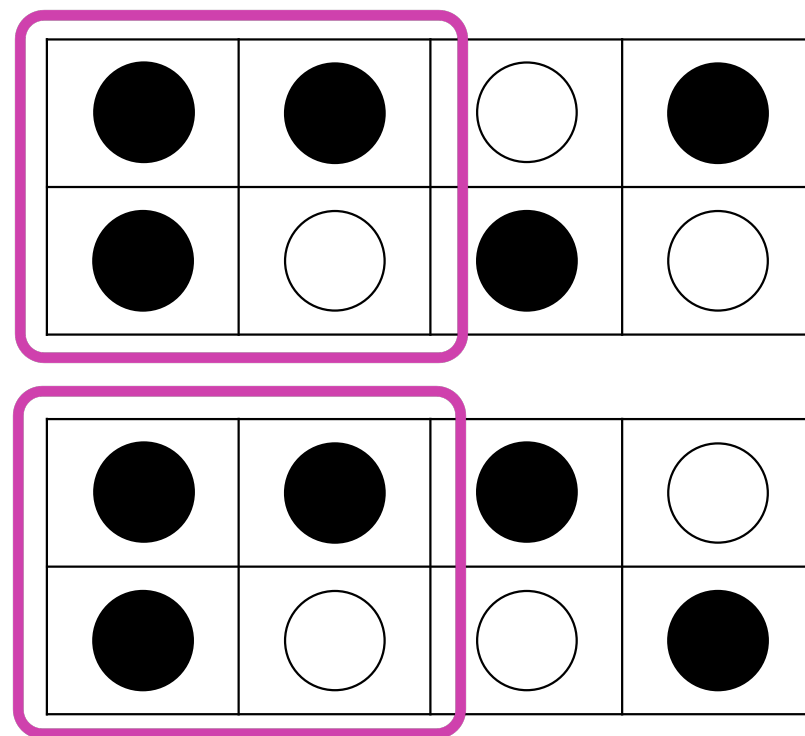
# Dekompozicija – Strukturni diagram





# Uočavanje uzoraka - generalizacija

- Jedan od elemenata računalnog razmišljanja
- Traženje sličnosti koje manje problemi imaju međusobno ili unutar samih sebe
- **Što vrijedi za sve ...?**
- Ove sličnosti se u računalnom razmišljanju zovu uzorci



# Uočavanje uzoraka – Važnost

- Jedan općeniti primjer iz svakodnevnog života:
  - Ako ste ikada sastavljali lego kockice (ili komad namještaja kupljenog u dućanu ili nešto slično), u početku je bilo teško, dok pri kraju postaje sve lakše i lakše. To je zbog čega je lakše pri kraju je zbog uočavanja uzoraka.
- Važnost
  - „Otkrivanje tople vode”
  - Gubljenje vremena i energije na pronalaženje (otkrivanje) rješenja za probleme za koje smo rješenje već prije pronašli rješavajući drugi dio problema ili rješavajući neki sasvim drugačiji problem.

# Uočavanje uzoraka – PRIPREMA KOLAČA

- Jedan od elemenata računalnog razmišljanja
- Traženje sličnosti koje manje problemi imaju međusobno ili unutar samih sebe
- **Što vrijedi za sve ...?**
- Ove sličnosti se u računalnom razmišljanju zovu uzorci



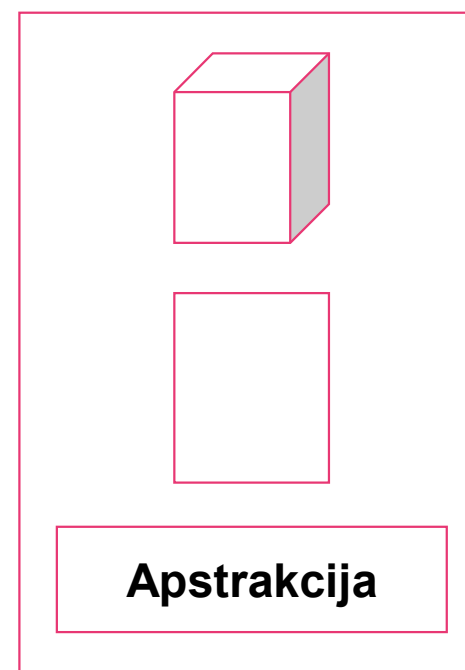
# Uočavanje uzoraka – Pusti otok

- Vatra
  - Suho i toplo sklonište
  - Sušenje odjeće
  - Signalna vatra
  - Kuhanje hrane
  - Kuhanje kao pročišćavanje vode
  - I jedna vrsta zaštite
  - Kako zapaliti vatru???
  - IDEJA – Rješenje koje smo naučili (kampiranje) ili čuli kako se radi prije (škola, tv ...)
- Koplje
  - Zaštita
  - Ribolov i lov
- Lišće
  - Za pokriti sklonište
  - Za korpu za nositi/čuvati stvari
  - Za dio zamke za ribolov i lov



# Apstrakcija – Kreiranje modela

- Jedan od elemenata računalnog razmišljanja
- Apstrakcija je proces filtriranja (ignoriranja) nebitnih karakteristika.
- Apstrakcija služi kako bismo kreirali generičku sliku o problemu, odnosno služi za kreiranje **modela**.



# Apstrakcija - važnost

- Pomoću apstrakcije bolje razumijemo problem, odnosno saznamo što zaista predstavlja problem
- Odbacivanjem svega što nije ključno, dobivamo pojednostavljenu sliku problema ili ideju o tome što je problem.
- Ta ideja je u stvari model problema.
  - Raditi s modelom je puno lakše nego bez modela.

# Apstrakcija – PRIPREMA KOLAČA

Općenito Važno za SVE kolače	Specifično Važno za JEDAN kolač
Svi kolači imaju sastojke. Koje točno sastojke, nije ključno.	Jaja, brašno, šećer ... Neki kolači nemaju šećer nego koriste med
Svaki sastojak ima podatak o količini	5 jaja, 300 g brašna.
Vrijeme pečenja	20 minuta na 180° C

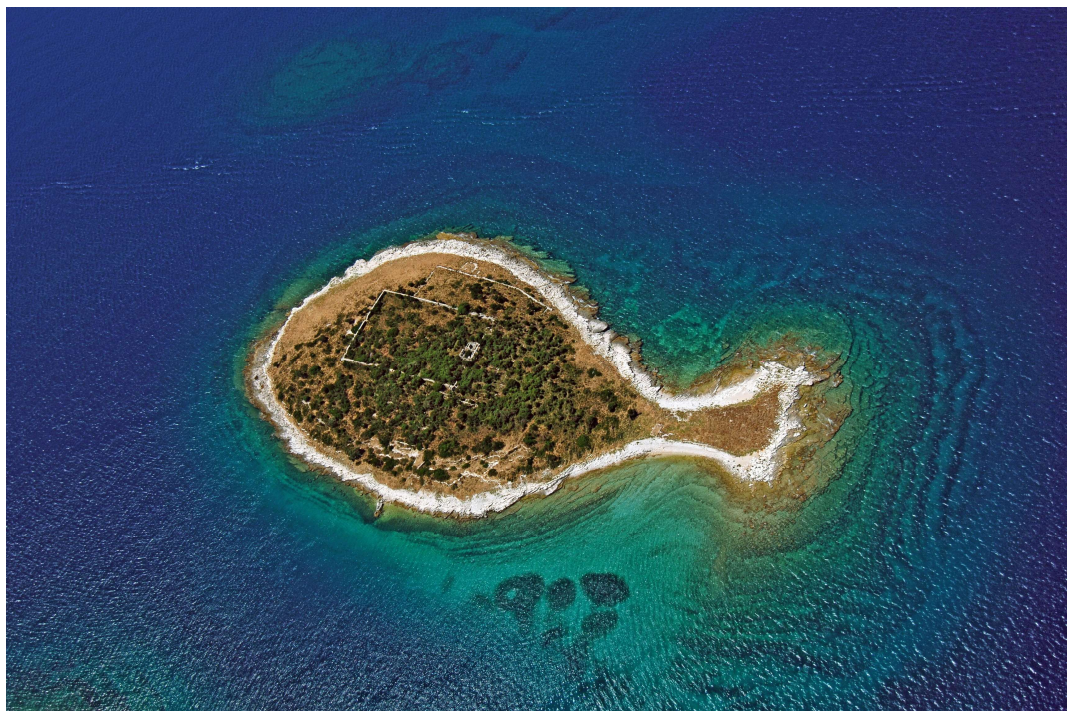


# Apstrakcija – primjeri





# Apstrakcija – primjeri



<https://www.htz.hr/hr-HR/promo-materijali/galerije-slika/more>  
Otok Gaz, Brijuni - Renco Kosinožić



Water vector created by brgfx - [www.freepik.com](http://www.freepik.com)

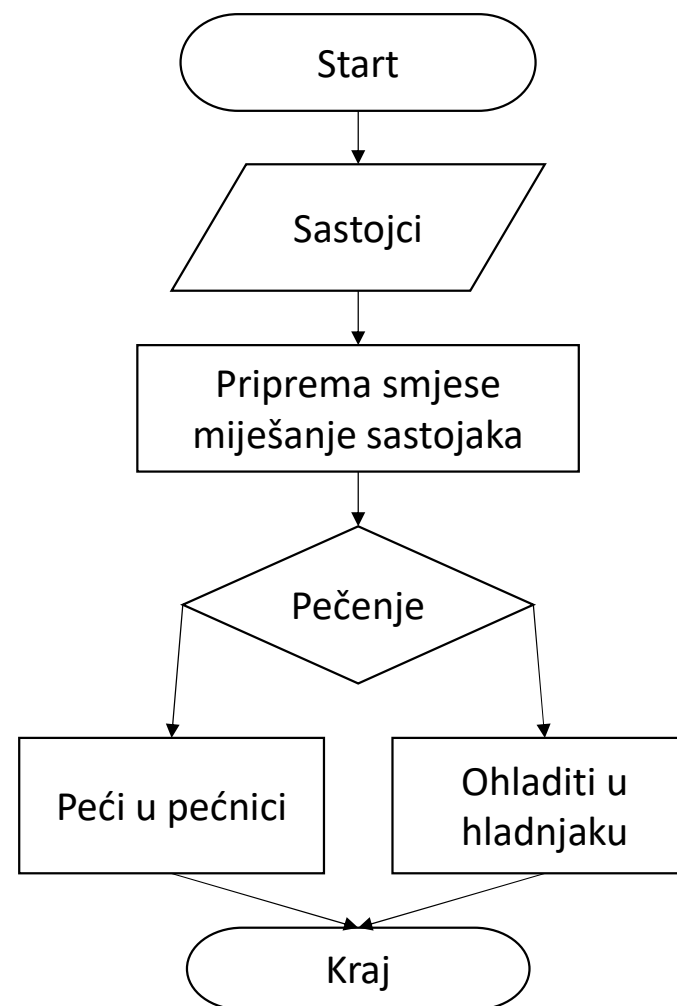
# Apstrakcija – PUSTI OTOK

- Mapa otoka
- Štap za ribolov
  - Točno definirana dužina štapa za ribolov – nije bitno
  - Čvrstoća da izdrži težinu i otpor ribe – bitno
- Štap za koplje
  - Točno definirana dužina štapa za ribolov – nije bitno
  - Točno definirana težina – nije bitno
  - Čvrstoća da može izdržati zakačenu oštricu, udaranje štapom ...



# Algoritam

- Jedan od elemenata računalnog razmišljanja
- Plan izvršavanja
- Korak-po-korak slijed instrukcija
- Prikaz grafički kao **dijagram toka** ili tekstualno pomoću **pseudo kôda**



# Algoritam – Važnost

- Primjeri algoritama:
  - Upute za sastavljanje lego kockica. Općenito upute za korištenje.
  - Navigacija putovanja od jedne točke do druge
- Važnost
  - Realizacija plana je dobra koliko su dobro osmišljeni koraci i koliko dobro je definiran njihov redoslijed izvršavanja.
  - Računala, odnosno računalni programi su onoliko dobri koliko su dobri algoritmi po kojima se ti programi izvršavaju.

# Algoritam – PUSTI OTOK

- Svaki manji dio problema treba imati svoj algoritam koji se onda uklapa u jedan veliki algoritam za rješenje problema
- Algoritam za hranu
  - Algoritam za ribolov
  - Algoritam za skupljanje voća, povrća, biljaka
  - Algoritam za lov
  - Algoritam za pripremu hrane
    - Algoritam za zapaliti vatru
    - Recept kao algoritam za pripremu hrane prije konzumiranja



# Provjera kvalitete rješenja – Evaluacija

- Obavezno proći kroz korake algoritma i provjeriti:
  - Jesu li svi koraci uključeni
  - Je li redoslijed koraka ispravan
- Uvjeti uspješnog algoritma:
  - Jesu li koraci razumljivi/jasni?
  - Je li kompletan – je li svaki dio problema uključen?
  - Je efikasan? – rješava li algoritam problem u najmanje moguće koraka, uporabom dostupnih resursa ...?

# Provjera kvalitete rješenja – Evaluacija

- Proces testiranja je jednostavan treba proći kroz sve korake algoritma. Jednostavno, ali za kompleksnije algoritme može biti teško.
  - Algoritam za svaki manji problem
  - Testiranje algoritma svakog manjeg problema
- Provjera kvalitete algoritma - Preživljavanje na pustom otoku



# Algoritmi Dijagram toka i pseudo kôd








# Sadržaj

- Dijagram toka
- Pseudo kôd

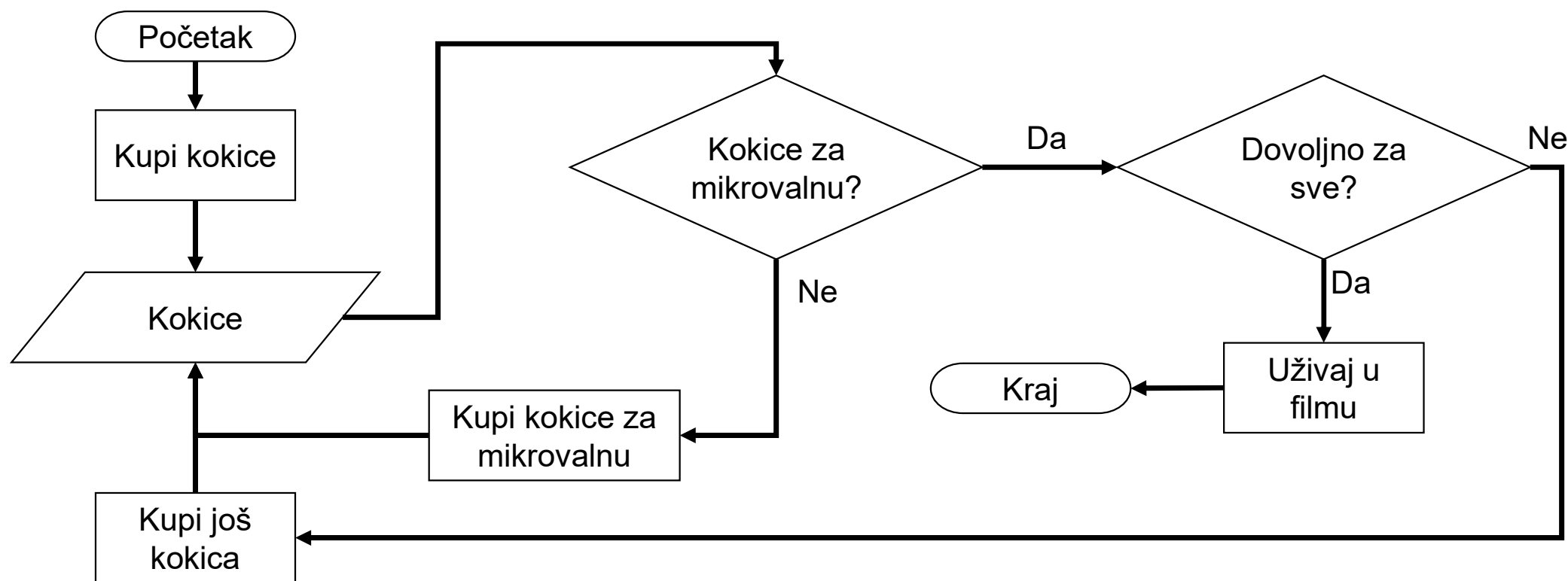
# Dijagram toka

- Jedan od dva glavna načina prikaza algoritma
- Jedan od najčešće korištenih načina za grafički prikaz algoritma
- Pomoću simbola definira tip koraka i redoslijed izvršavanja koraka

# Dijagram toka - simboli

Naziv	Simbol	Primjena
Početak / Kraj		Predstavlja prvi, odnosno zadnji korak algoritma.
Proces		Nekakva instrukcija, zadaća, aktivnost koju treba izvršiti
Odluka		Uvjetno grananje slijeda aktivnosti na osnovu odluke.
Ulaz podataka / Izlaz podataka		Ulazni podaci u algoritam predstavljaju informacije čijom obradom se onda kreiraju izlazne informacije koje se koriste kao informacije za izvršavanje algoritma.
Smjer		Povezuje simbole u cjelinu. Strelica definira smjer, odnosno slijed izvršavanja koraka algoritma.

# Dijagram toka – Gledanje filma



# Pseudo kôd

- Jedan od dva glavna načina prikaza algoritma
- Svaki programski jezik (Python na primjer) ima jasno definiranu sintaksu, odnosno pravila pisanja programskog kôda (instrukcija za izvršavanje naredbi).
- Pseudo kôd može sličiti programskom kôdu nekog programskog jezika, ali pseudo kôd NEMA definiranu sintaksu.
- Pseudo kôd možete pisti na bilo kojem jeziku kojeg vi i vaš tim možete razumjeti
- Mogu (ne moraju) se koristiti neke preporuke kao:
  - UNOS za unos podataka (kako nije bitno)
  - ISPIS za prikaz podataka (kako i na ekran ili pisač ili na mrežu ... nije bitno)

# Pseudo kôd - primjer

```
ISPIS      „Vaše ime i prezime”
UNOS       korisnik utipka ime i prezime
POHRANI    informacije koje je korisnik unio pohraniti u varijablu punolme
ISPIS      „Pozdrav” + name
ISPIS      „Koliko godina imate?”
UNOS       korisnik utipka broj godina
POHRANI    pohrani broj koji je korisnik unio u varijablu dob
AKO JE dob >= 18 ONDA
    ISPIS  „Vi ste punoljetni.”
INAČE
    ISPIS  „Još niste punoljetni.”
```



# Primjena elemenata računalnog razmišljanja

# Sadržaj

- Elementi računalnog razmišljanja
- Primjena računalnog razmišljanja na probleme koji nisu povezani s računarstvom
- Ponavljanje s vježbama



# Sažetak

## Elementi računalnog razmišljanja

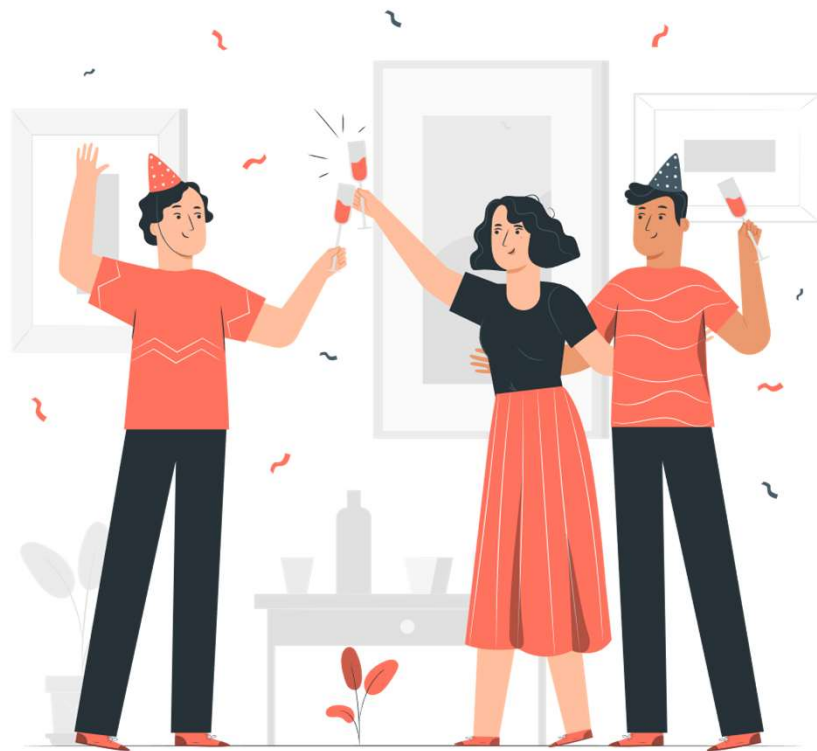
	Dekompozicija ili Rastavljanje	Uočavanje uzoraka ili Generalizacija	Apstrakcija	Algoritam ili Automatizacija
<b>Kratka definicija</b>	Rastavljanje složenih problema na manje, jednostavnije probleme	Uočavanje sličnosti između dijelova	Filtriranje bitnih od nebitnih informacija.	Povezivanje dijelova u korak-po-korak slijed aktivnosti
<b>Potkategorije</b>	Analiza podataka	Vizualizacija podataka	Generalizacija, izrada modela na osnovu uzoraka	Izrada algoritma, dijagrama toka, pseudo kôd, simulacija
<b>Primjeri iz svakodnevnog života</b>	Memoriranje broja telefona: <ul style="list-style-type: none"> <li>- Pozivni broj države</li> <li>- Pozivni broj mreže</li> <li>- Prvi dio broja telefona</li> <li>- Drugi dio broja telefona</li> </ul>	Svi psi imaju: <ul style="list-style-type: none"> <li>- Rep</li> <li>- Krzno</li> <li>- Šape</li> </ul>	Karta ulica grada, Simboli na vratima nekih prostorija	Perilica rublja automatizira proces pranja prljavog veša, bez obzira jesu li to hlače, košulja, čarape ili posteljina.

# Primjena računalnog razmišljanja

- U svakodnevnom životu
  - Pečenje kolača
  - Organizacija proslave
  - Snimanje filma
- U poslovanju
  - Novi proizvodi: Mobilni telefon, Električni bicikl ...
- Razvoj softvera
  - Igrice. Primer jednostavnijih igrica: Križić–Kružić, Poveži 4, Vješala, Potapanje brodova
  - Aplikacije. Primjer: Pametna kuća (Smart Home)

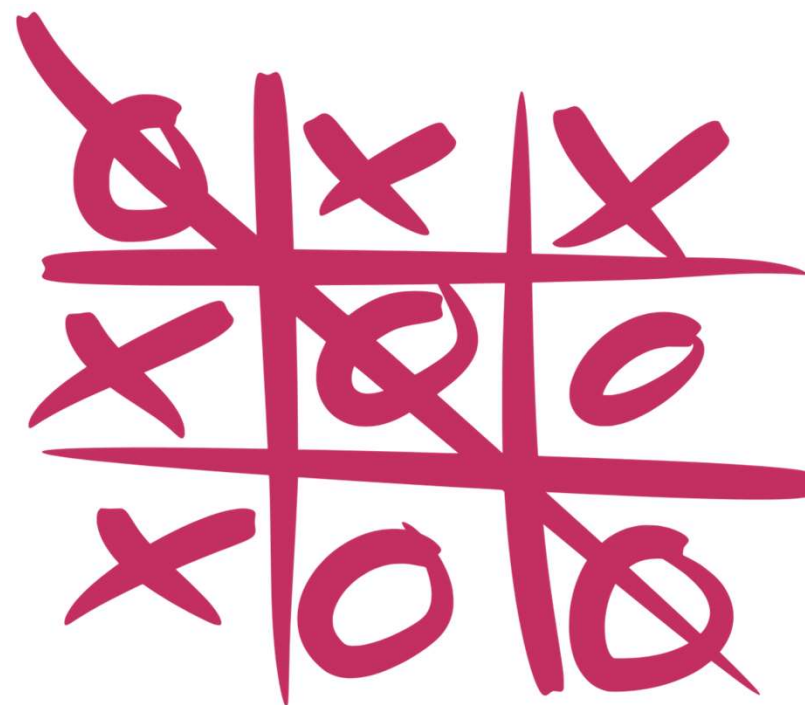
# Vježba

- Organizacija proslave ...
- Snimanje filma



# Vježba

- Križić kružić



# Križić – kružić

- START – pokreni igricu
- SET – kreiraj varijable: *igrac*, *status\_igre*, *polja\_na\_ploci*, *oznaka*, *izabrano\_polje* te po potrebi varijabli dodijeli početnu vrijednost
- REPEAT UNTIL – ponavljaj niže navedene korake sve dok je vrijednost varijable *status\_igre* jednaka -1.
  - PRINT – na ekranu iscrtaj ploču koristeći vrijednosti iz varijable *polja\_na\_ploci*.
  - SET – ovisno o aktivnom igraču dodijeli vrijednost „X” ili „O” varijabli *oznaka*
  - INPUT – zatraži od igrača da napravi potez tako što će utipkati broj polja na koje želi postaviti svoju *oznaku*.
  - SET – izbor igrača dodijeliti varijabli *izabrano\_polje*.
  - IF – provjeriti je li to polje slobodno
    - SET – ako je polje slobodno ažurirati varijablu *polja\_na\_ploci*
    - PRINT – a polje nije slobodno, javiti igraču da je napravio pogrešan potez te neka pokuša ponovo
  - SET *status\_igre* – pomoću funkcije *Status\_Igre()* provjeriti status igre. Funkcija *Status\_Igre()* treba vratiti broj koji ćemo dodijeliti varijabli *status\_igre*.
  - IF – provjeriti koja je vrijednost varijable *status\_igre*
    - Vrijednost 1 – PRINT – ispiši koji igrač je pobijedio
    - Vrijednost 0 – PRINT – ispiši neriješeno
    - Vrijednost -1 – SET – promijeni igrača tako da varijabli *igrac* dodijeliš vrijednost suprotnu od one koju je imao te se vrati u sljedeći krug
- END – završi igru

Definicije funkcija

Blok koraka koji se ponavljaju dok je uvjet ispunjen *while* petlja



# Križić – kružić – Primjer u Pythonu

```
ALGEBRA d.o.o.  
  
Krizic Kruzic  
  
IGRAC 1 (X) - IGRAC 2 (O)  
  
  |  |  |  
 1 | 2 | 3  
---|---|---  
 4 | 5 | 6  
---|---|---  
 7 | 8 | 9  
  |  |  |  
  
Igrac 1  
Unesite broj polja na ploci: 
```

# Križić – kružić – Inicijalizacija

## Scratch blok



## Python kod

```
import os

polje_na_ploci = [0,1,2,3,4,5,6,7,8,9]

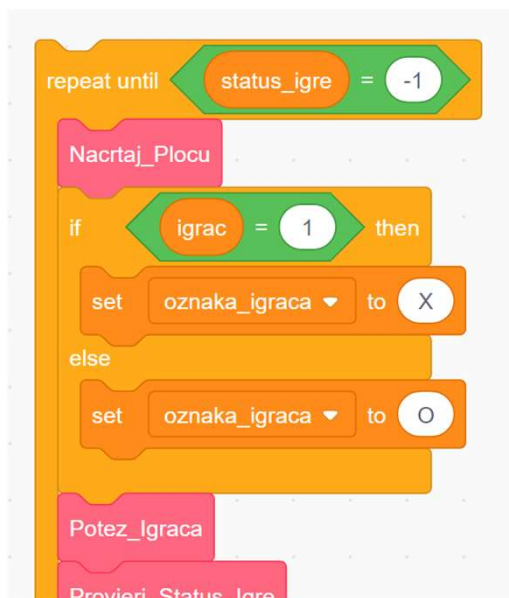
def Status_Igre(): #funkcija

def Iscrtaj_Plocu(): #funkcija

igrac = 1
status_igre = -1
... Dio kôda je izostavljen zbog čitljivosti slidea
```

# Križić – kružić – Igra

## Scratch blok



## Python kod

... Dio kôda je izostavljen zbog čitljivosti slidea

```
while status_igre == -1:
    Iscrtaj_Plocu()
    if igrac % 2 == 1:
        igrac = 1
    else:
        igrac = 2

    print('\n\nIgrac', igrac)
    izabrano_polje = int(input('Unesite broj polja na ploci: '))
    if igrac == 1:
        oznaka_igraca = 'X'
    else:
        oznaka_igraca = 'O'
```

... Dio kôda je izostavljen zbog čitljivosti slidea



# Križić – kružić – Igra

## Scratch blok



## Python kod

... Dio kôda je izostavljen zbog čitljivosti slidea

```
while status_igre == -1:
```

... Dio kôda je izostavljen zbog čitljivosti slidea

```
if izabrano_polje == 1 and polje_na_ploci[1] == 1:  
    polje_na_ploci[1] = oznaka_igraca  
  
elif izabrano_polje == 2 and polje_na_ploci[2] == 2:  
    polje_na_ploci[2] = oznaka_igraca  
  
elif izabrano_polje == 3 and polje_na_ploci[3] == 3:  
    polje_na_ploci[3] = oznaka_igraca
```

... Dio kôda je izostavljen zbog čitljivosti slidea

```
elif izabrano_polje == 9 and polje_na_ploci[9] == 9:  
    polje_na_ploci[9] = oznaka_igraca  
  
else:  
    print('POGRESAN POTEZ ' )  
    programPause = input('Za nastavak pritisni <ENTER> tipku ...')  
  
    igrac -= 1
```

... Dio kôda je izostavljen zbog čitljivosti slidea

# Križić – kružić – Igra

## Scratch blok



## Python kod

... Dio kôda je izostavljen zbog čitljivosti slidea

```
while status_igre == -1:
```

... Dio kôda je izostavljen zbog čitljivosti slidea

```
status_igre = Status_Igre()  
igrac += 1
```

```
Iscrtaj_Plocu()  
print('\n\nREZULTAT\n')  
if status_igre == 1:  
    print('Igrac', igrac-1, 'je pobijedio!\n\n')  
else:  
    print('Nerijeseno\n\n')
```

... Dio kôda je izostavljen zbog čitljivosti slidea

# Križić – kružić – Funkcija *Status\_Igre()*

## Scratch blok



## Python kod

```
def Status_Igre():
```

```
    if polje_na_ploci[1] == polje_na_ploci[2] and polje_na_ploci[2] == polje_na_ploci[3]: return 1
    elif polje_na_ploci[4] == polje_na_ploci[5] and polje_na_ploci[5] == polje_na_ploci[6]: return 1
    elif polje_na_ploci[7] == polje_na_ploci[8] and polje_na_ploci[8] == polje_na_ploci[9]: return 1
    elif polje_na_ploci[1] == polje_na_ploci[4] and polje_na_ploci[4] == polje_na_ploci[7]: return 1
    elif polje_na_ploci[2] == polje_na_ploci[5] and polje_na_ploci[5] == polje_na_ploci[8]: return 1
    elif polje_na_ploci[3] == polje_na_ploci[6] and polje_na_ploci[6] == polje_na_ploci[9]: return 1
    elif polje_na_ploci[1] == polje_na_ploci[5] and polje_na_ploci[5] == polje_na_ploci[9]: return 1
    elif polje_na_ploci[3] == polje_na_ploci[5] and polje_na_ploci[5] == polje_na_ploci[7]: return 1
```

```
    elif (polje_na_ploci[1] != 1 and
          polje_na_ploci[2] != 2 and
          polje_na_ploci[3] != 3 and
          polje_na_ploci[4] != 4 and
          polje_na_ploci[5] != 5 and
          polje_na_ploci[6] != 6 and
          polje_na_ploci[7] != 7 and
          polje_na_ploci[8] != 8 and
          polje_na_ploci[9] != 9):
        return 0
```

```
    else:
        return -1
```



# Križić – kružić – Cijeli Python kôd

```
import os

polje_na_ploci = [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]

def Status_Igre():
    if polje_na_ploci[1] == polje_na_ploci[2] and polje_na_ploci[2] == polje_na_ploci[3]:
        return 1

    elif polje_na_ploci[4] == polje_na_ploci[5] and polje_na_ploci[5] == polje_na_ploci[6]:
        return 1

    elif polje_na_ploci[7] == polje_na_ploci[8] and polje_na_ploci[8] == polje_na_ploci[9]:
        return 1

    elif polje_na_ploci[1] == polje_na_ploci[4] and polje_na_ploci[4] == polje_na_ploci[7]:
        return 1

    elif polje_na_ploci[2] == polje_na_ploci[5] and polje_na_ploci[5] == polje_na_ploci[8]:
        return 1

    elif polje_na_ploci[3] == polje_na_ploci[6] and polje_na_ploci[6] == polje_na_ploci[9]:
        return 1

    elif polje_na_ploci[1] == polje_na_ploci[5] and polje_na_ploci[5] == polje_na_ploci[9]:
        return 1

    elif polje_na_ploci[3] == polje_na_ploci[5] and polje_na_ploci[5] == polje_na_ploci[7]:
        return 1
```

# Vježba

- Potapanje brodova (Battleship)
- Poveži 4 (Connect Four)
- Vješala (Hangman)

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4			X							
5						X	X			
6		X						X		X
7				X						X
8	X	X						X		
9										
10										

[Battleship game board - Battleship \(game\) - Wikipedia](#)

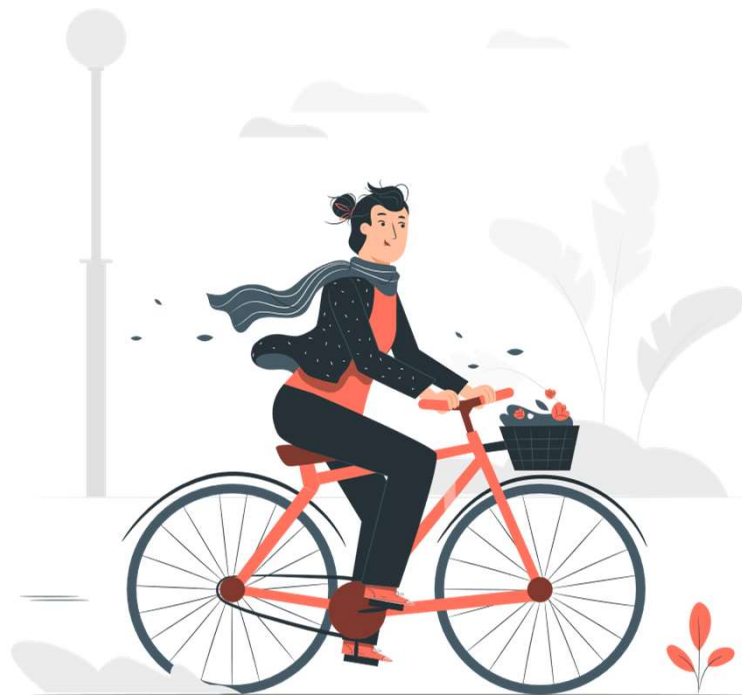
# Vježba

- Mobilni telefon



# Vježba

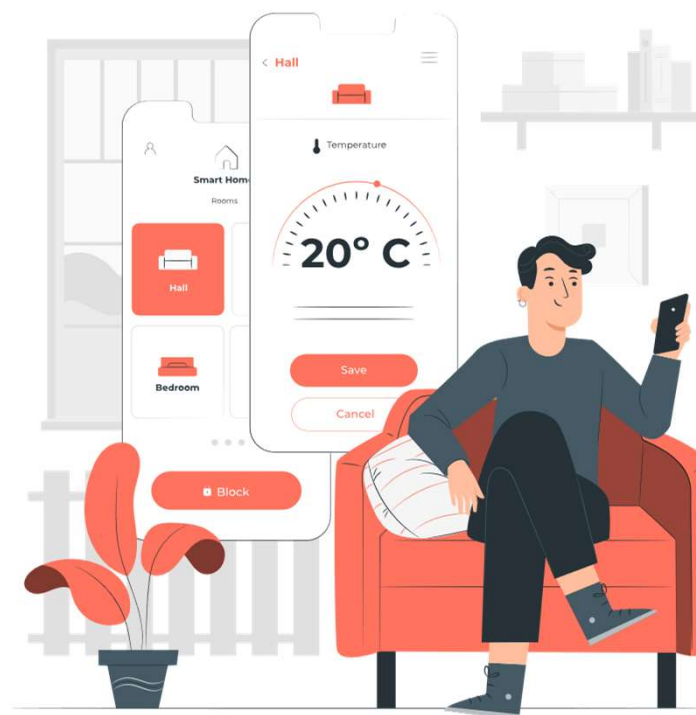
- Električni bicikl
- Zadatak: Primjernom elemenata računalnog razmišljanja osmislite Električni bicikl
  - Možda ima i softver + app, ne samo hardver ;-)





# Vježba

- Pametni dom „My Smart Home”
- Zadatak: Primjernom elemenata računalnog razmišljanja osmislite My Smart Home sustav (softver i hardver)



Hvala na  
pažnji!

