LEARN TO PROGRAM

- gets — geta $\overset{\text{i čeka}}{}$ moj input iz komandne linije

- name = gets.chomp — chomp miče zadnji ENTER koji se spremi iz getsa

- METODE
  .upcase
  .downcase
  .swapcase } povećava, smanjuje ili zamjenjuje slova
  .capitalize — povećava samo prvi znak u stringu

- puts string.center(40) — centrira string uzevši da je širina reda 40 znakova

- puts string.ljust(40) — aligna string lijevo ili desno
  puts string.rjust(40)

- EKSPONENCIRANJE ** puts 5**2 (5^2)

  MODUL % puts 7%3 ostatak dijeljenja
  ↳ 1

- .abs — apsolutna vrijednost broja (-3 je 3)

RANDOM BROJEVI

puts ~~rand~~ rand → float 0.0 do 1.0

puts rand(5) → integeri od Ø do 4 (5 mogućih vrijednosti)

puts rand(1) → uvijek vraća Ø

- srand 1985
  puts rand(100)
  puts rand(100)
  → daje seed za rand. Nakon ovoga će nam random brojevi ići u istom nizu
  → za poništi srand, utipkat samo srand

- MATEMATIČKE FUNKCIJE

```
Math::PI        - konstanta

Math::E

Math.cos (Math::PI/3)
Math.tan (..)
Math.log (...)
Math.sqrt (5)
```

- Array

```
array = [ 'nesto', 'drugo', 'trece' ]

puts array.join(',')      → spaja elemente u nizu
  ↳ nesto, drugo, trece
```

POP & PUSH

```
niz = [1, 2, 3]
niz.push 8                → PUSH DODAJE element na zadnje
    ↳ [1, 2, 3, 8]          mjesto u arrayu

niz.pop                   → POP SKIDA zadnji element iz
    ↳ 8                     niza (niz se smanji)
```

SPREMANJE OUTPUTA PROGRAMA

COMMAND LINE

- `ruby program.rb > output.txt`

IZ RUBY-a

-
```
filename = 'filename.txt'
string = 'nesto'

File.open filename, 'w' do |f|
    f.write string
end
```

- ČITANJE IZ FILEA

```
read_str = File.read filename
```

YAML

```
require 'yaml'

niz = ['...', '...', '...']

niz_string = niz.to_yaml

File.open 'filename.txt', 'w' do |f|
    f.write niz_string
end

read_string = File.read 'filename.txt'
read_niz = YAML::load read_string
```

DEFINIRANJE STRINGOVA

- double quote "..." → \n je newline char
- single quote '...' → prelazak u novi red i nastavak pisanja

INTERPOLACIJA

```
puts "#{2* 10 **4 +1} nešto nešto"
```

Dir[ ] → traži fileove na disku
↳ vraća array sa nazivima ako nađe
↳ vraća prazan array ako ne nađe ništa

PR.

Dir['*.jpg'] ili Dir['*.{JPG, jpg}']

TRAŽENJE U PARENT DIRU

```
Dir['../*.{JPG,jpg}']
```

MIJENJANJE PATHA CURRENT DIR-a

```
Dir.chdir '~/ovo/ono/nešto'
```

File.exist?

MOVE-anje odnosno RENAMEANJE FILE-a

```
File.rename name, new_name
```

VRIJEME

```
time = Time.now      #.trenutno vrijeme

                          MJESEC
puts Time.local (2000, 1, 1)      # 2000 godina, daje vrijeme za to
                             ↓
                            DAN
```

KLASE

```
puts (42.class)  → Fixnum
puts ("nesto".class)  → String
```

.class vraća klasu od objekta

DEFINIRANJE NOVE METODE ZA KLASU

```
class Integer
    def nesto
        :
        :
    end
end

puts 5.nesto
puts 8.nesto
```

STVARANJE NOVE KLASE

```
class Die
   def initialize     → metoda koja se 'izvrši' prilikom instanciranja objekta
      roll                                  tipa  Die.new
   end
   def roll
     @number_showing = 1 + rand (6)     → @ → instance varijabla – dostupna i
   end                                      svim drugim metodama u klasi
   def showing
     @ numbers_showing
   end
end
```

```
class Dragon
   def hungry?   → ako metoda vraća false ili true
      @stuff <= 2
   end
  :
end
```

PROCS

-
  ```
  toast = Proc.new do
      puts 'Cheers!'
  end
  ```
  → Proc sadrži block koda. Block je između do i end

  ↳ Proc je objekt → toast je instanca

  ```
  toast.call
      ↳ Cheers!
  ```

-
  ```
  do_you_like = Proc.new do |good_stuff|
      puts "Volim #{good_stuff}"
  end

  do_you_like.call 'chocolate'
          ↳ Volim chocolate
  ```

- Procevi se mogu definirati kao atributi za <u>metodu!</u>

  ```
  def nesto neki_proc
      put "..."
      neki_proc.call
  end

  proc1 = Proc.new do
      puts "Proc 1"
  end
  proc2 = Proc.new do
      puts "Proc 2"
  end

  nesto.(proc 1)
    ↳ ...
      Proc1
  ```
  → poziva proc 1 unutar metode nesto

  ```
  nesto (proc 2)
    ↳ ...
      Proc 2
  ```
  → poziva proc 2 unutar metode nesto

METODA MOŽE VRACÁT PROC

```
def compoze proc 1 , proc 2
      Proc . new do |x|
            proc 2 . call (proc 1. call(x))
      end
end
```