Řetězce, náhodná čísla

IB113

Radek Pelánek

2022

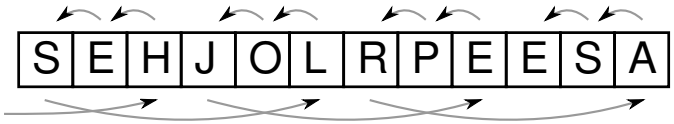
Rozcvička: šifry

1. C S A R B V
   E K T E O A
2. A J L B N O C E
3. O U A G A D O U G O U
4. C S B U J T M B W B

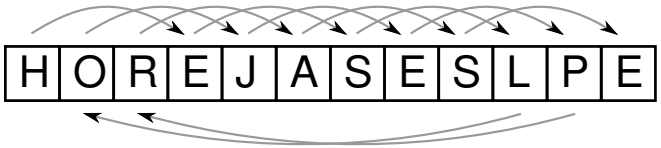
Transpoziční šifry

pozpátku

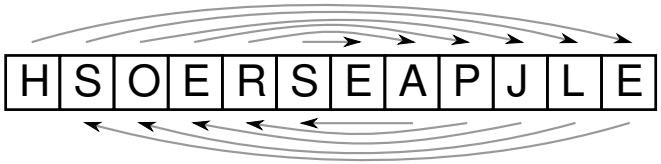
trojice pozpátku



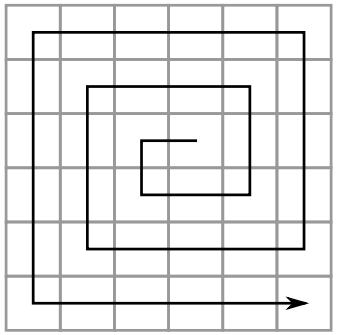
ob tři



dopředu dozadu

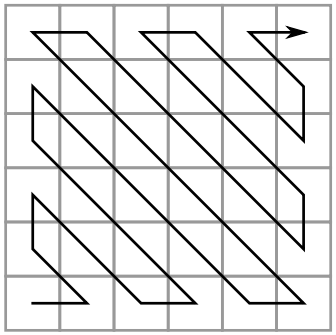


šnek



L	B	A	K	I	N
I	C	S	E	J	B
Z	H	O	P	D	Y
K	O	K	L	A	R
O	V	A	N	Y	U
U	H	R	A	Z	E

cik-cak



N	I	O	U	Z	E
H	B	K	K	H	A
C	O	Y	A	Z	R
L	S	V	R	B	I
K	A	E	A	U	L
P	O	D	J	N	Y

Substituční šifry

Jednoduchá substituce - posun o 3 pozice

	K	O	Z	A
	↓	↓	↓	↓
	10	14	25	0
+3	↓	↓	↓	↓
	13	17	2	3
	↓	↓	↓	↓
	N	R	C	D

Substituce podle hesla

```
HLEDEJPODLIPOU
SLONSLONSLONSL
ZWSQWUDBVWWCGF
```

H → 7

S → 18

7 + 18 → 25 → Z

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Řetězce a znaky – ukázky operací

```
"kos" * 3
"petr" + "klic"
text = "velbloud"
len(text)
text[0]
text[2]
text[-1]
"e" in text
ord('b')
chr(99)
```

`str()` – explicitní přetypování na řetězec

Uvozovky, apostrofy

- jiné jazyky často: uvozovky pro řetězce, apostrofy pro znaky
- Python: lze používat uvozovky i apostrofy
- PEP8: hlavně konzistence

Indexování od nuly

Proč indexujeme od 0?

- částečně „historicky-technické“ důvody
- ale i dobré „matematické“ důvody

Pro zájemce:

Why numbering should start at zero (Edsger W. Dijkstra)
`http://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html`

`http://programmers.stackexchange.com/questions/110804/why-are-zero-based-arrays-the-norm`

`https://www.quora.com/Why-do-array-indexes-start-with-0-zero-in-many-programming-languages`

Kódování

- jak jsou znaky reprezentovány?
  ASCII, ISO 8859-2, Windows-1250, Unicode, UTF-8, ...
  `http://www.joelonsoftware.com/articles/Unicode.html`
- Python3 – Unicode řetězce
- pro tento kurz:
  - `ord`, `chr` – převod znaků na čísla a zpět
  - anglická abeceda má přiřazena po sobě jdoucí čísla

```
for i in range(26):
    print(chr(ord('A')+i))
```

Řetězce – pokročilejší indexování

(specifické pro Python)

```
text = "velbloud"
text[:3]     # prvni 3 znaky
text[3:]     # od 3 znaku dale
text[1:8:2] # od 2. znaku po 7. krok po 2
text[::3]    # od zacatku do konce po 3
```

Řetězce – změny

- neměnitelné (immutable) – rozdíl oproti seznamům a oproti řetězcům v některých jiných jazycích
- změna znaku – vytvoříme nový řetězec

```
text = "kopec"
text[2] = "n" # error
text = text[:2] + "n" + text[3:]
```

Řetězce: další operace

```
text = "i Have a dream."
print(text.upper())
print(text.lower())
print(text.capitalize())
print(text.rjust(30))
print("X", text.center(30), "X")
print(text.replace("dream", "nightmare"))
```

... a mnoho dalších, více později, příp. viz dokumentace

Pozn. objektová notace

Příklad: Transpozice (rozcvička 1)

- úkol: přepis textu do N sloupců
- příklad vstupu a výstupu:
  - `CESKATREBOVA`, 2
  - `C S A R B V`
    `E K T E O A`

Transpozice (rozcvička 1)

```
def cipher_columns(text, n):
    for i in range(n):
        for j in range(len(text) // n + 1):
            position = j * n + i
            if position < len(text):
                print(text[position], end="")
        print()
```

Transpozice (rozcvička 1), kratší varianta

Za využití notace specifické pro Python:

```
def cipher_columns(text, n):
    for i in range(n):
        print(text[i::n])
```

Příklady interaktivně: řetězce, vnořené cykly

```
Oua
    gad
        oug
            ou
```

```
O   a   g
 u   d   o
  a   o   u
   g   u
```

```
O   a   g
 u g d u o
  a   o   u
```

```
O
 u
  a
 g
a
 d
  o
 u
g
 o
  u
```

Caesarova šifra (rozcvička 3)

- substituční šifra – posun v abecedě
- vstup: text, posun
- výstup: zašifrovaný text
- `BRATISLAVA`, $1 \rightarrow$ `CSBUJTMBWB`

Caesarova šifra – řešení

```
def caesar_cipher(text, n):
    result = ""
    text = text.upper()
    for i in range(len(text)):
        if text[i] == " ":
            result = result + " "
        else:
            c = ord(text[i]) + n
            if (c > ord("Z")): c = c - 26
            result = result + chr(c)
    return result
```

Pozn. Řešení má nedostatky – zkuste najít a vylepšit.

Caesarova šifra – rozlomení

- máme text zašifrovaný Caesarovou šifrou (s neznámým posunem)
- jak text dešifrujeme?
- příklad: `MPKTWTDVLVELMZCF`

Caesarova šifra – rozlomení

- máme text zašifrovaný Caesarovou šifrou (s neznámým posunem)
- jak text dešifrujeme?
- příklad: `MPKTWTDVLVELMZCF`
- jak to udělat, aby program vrátil jen jednoho kandidáta?

Caesarova šifra – rozlomení

k	Kandidát	b_s	b_f
0	MPKTWTDVLVELMZCF	0	21
1	NQLUXUEWMWFMNADG	13	0
2	ORMVYVFXNXGNOBEH	24	9
3	PSNWZWGYOYHOPCFI	5	-3
4	QTOXAXHZPZIPQDGJ	10	-6
5	RUPYBYIAQAJQREHK	0	9
6	SVQZCZJBRBKRSFIL	0	3
7	TWRADAKCSCLSTGJM	32	26
8	UXSBEBLDTDMTUHKN	0	24
9	VYTCFCMEUENUVILO	11	46
10	WZUDGDNFVFOVWJMP	0	-6
11	XAVEHEOGWGPWXKNQ	5	-2
12	YBWFIFPHXHQXYLOR	0	-28
13	ZCXGJGQIYIRYZMPS	0	-13
14	ADYHKHRJZJSZANQT	0	16
15	BEZILISKAKTABORU	**67**	**59**
16	CFAJMJTLBLUBCPSV	0	11
17	DGBKNKUMCMVCDQTW	5	-4
18	EHCLOLVNDNWDERUX	17	31
19	FIDMPMWOEOXEFSVY	5	22
20	GJENQNXPFPYFGTWZ	4	-23
21	HKFOROYQGQZGHUXA	16	-17
22	ILGPSPZRHRAHIVYB	28	18
23	JMHQTQASISBIJWZC	9	0
24	KNIRURBTJTCJKXAD	5	24
25	LOJSVSCUKUDKLYBE	4	29

Vigenèrova šifra

- substituce podle hesla – „sčítáme“ zprávu a heslo
- vhodné cvičení
- rozlomení Vigenèrovovy šifry?

Řetězce a for cyklus

Můžeme iterovat přímo přes jednotlivá písmena.

```
text = "prase"
for i in range(len(text)):
    print(text[i])
for letter in text:
    print(letter)
```

Velmi nevhodný hybrid:

```
for i in text:
    print(i)
```

Ladění

- ladící výpisy
  - např. v každé iteraci cyklu vypisujeme stav proměnných
  - doporučeno vyzkoušet na ukázkových programech ze slidů
- použití debuggeru
  - dostupný přímo v IDLE / Thonny
  - sledování hodnot proměnných, spuštěných příkazů, breakpointy, ...

Ladění a funkce

- dobrá dekompozice na funkce usnadňuje ladění
- „hledání chyby v celém programu“ vs. „hledání chyby v dílčí funkci“
- „unit testing“, „test driven development“

Čtení chybových hlášek

```
Traceback (most recent call last):
  File "sorting.py", line 63, in <module>
    test_sorts()
  File "sorting.py", line 59, in test_sorts
    sort(a)
  File "sorting.py", line 52, in insert_sort
    a[j] = curent
NameError: name 'curent' is not defined
```

- kde je problém? (identifikace funkce, číslo řádku)
- co je za problém (typ chyby)

Základní typy chyb

- SyntaxError
  - invalid syntax: zapomenutá dvojtečka či závorka, záměna = a ==, ...
  - EOL while scanning string literal: zapomenutá uvozovka
- NameError – špatné jméno proměnné (překlep v názvu, chybějící inicializace)
- IndentationError – špatné odsazení
- TypeError – nepovolená operace (sčítání čísla a řetězce, přiřazení do řetězce, ...)
- IndexError – chyba při indexování řetězce, seznamu a podobně („out of range“)

Časté chyby

projeví se „rychle“ (program spadne hned):

- zapomenutá dvojtečka, závorka, uvozovka
- překlepy
- použití $=$ tam, kde mělo být $==$
- špatný počet argumentů při volání funkce
- zapomenuté `len` v “`for i in range(alist)`”

Časté chyby

nemusí se projevit rychle / vždy:

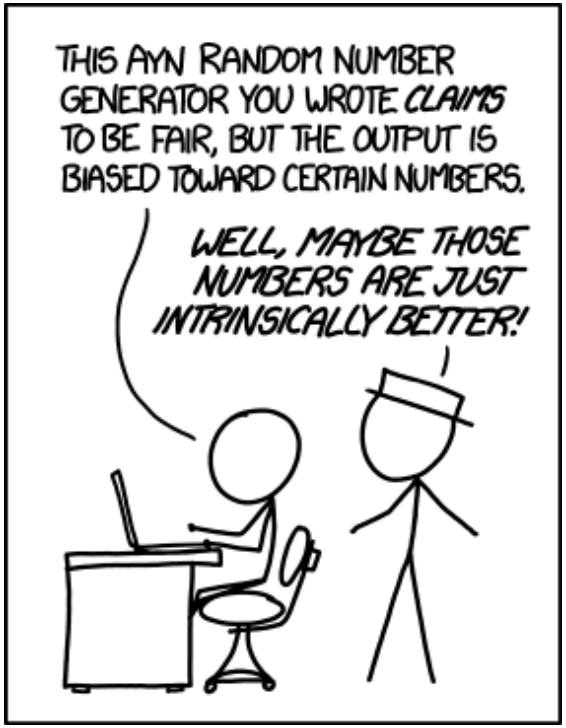
- použití == tam, kde mělo být =
- `"True"` místo `True`
- záměna `print` a `return`
- dělení nulou
- chybné indexování (řetězce, seznamy)

Náhodná čísla

- přesněji: *pseudo-náhodná* čísla
- opravdová náhodná čísla: `https://www.random.org/`
- bohaté využití v programování: výpočty, simulace, hry, ...
- Python
  - `import random`
  - `random.random()` – float od 0 do 1
  - `random.randint(a, b)` – celé číslo mezi `a`, `b`
  - mnoho dalších funkcí

Náhodná čísla: xkcd

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```



https://xkcd.com/221/
https://xkcd.com/1277/

Náhodná čísla: průměr vzorku

Vygenerujeme náhodná čísla a vypočítáme průměrnou hodnotu:

```
def random_average(count, maximum=100):
    total = 0
    for i in range(count):
        total += random.randint(0, maximum)
    return total / count
```

Jakou očekáváme hodnotu na výstupu? Jak velký bude rozptyl hodnot? (Názorná ukázka *centrální limitní věty*)

Simulace volebního průzkumu

- volební průzkumy se často liší; jaká je jejich přesnost?
- přístup 1: matematické modely, statistika
- přístup 2: simulace
- program:
  - vstup: preference stran, velikost vzorku
  - výstup: preference zjištěné v náhodně vybraném vzorku

Simulace volebního průzkumu

```
def survey(size, pref1, pref2, pref3):
    count1 = 0
    count2 = 0
    count3 = 0
    for i in range(size):
        r = random.randint(1, 100)
        if r <= pref1: count1 += 1
        elif r <= pref1 + pref2: count2 += 1
        elif r <= pref1 + pref2 + pref3: count3 += 1
    print("Party 1:", 100 * count1 / size)
    print("Party 2:", 100 * count2 / size)
    print("Party 3:", 100 * count3 / size)
```

Poznámky ke zdrojovému kódu

- uvedené řešení není dobré:
  - „copy & paste“ kód
  - funguje jen pro 3 strany
- lepší řešení – využití seznamů (příště)

Výpočet π

- $\pi = 3.14159265359\ldots$
- Ale jak se na to přišlo?
- Jak vypočítat π?

Výpočet π

Příklady naivních metod:

- Gregoryho-Leibnizova řada:

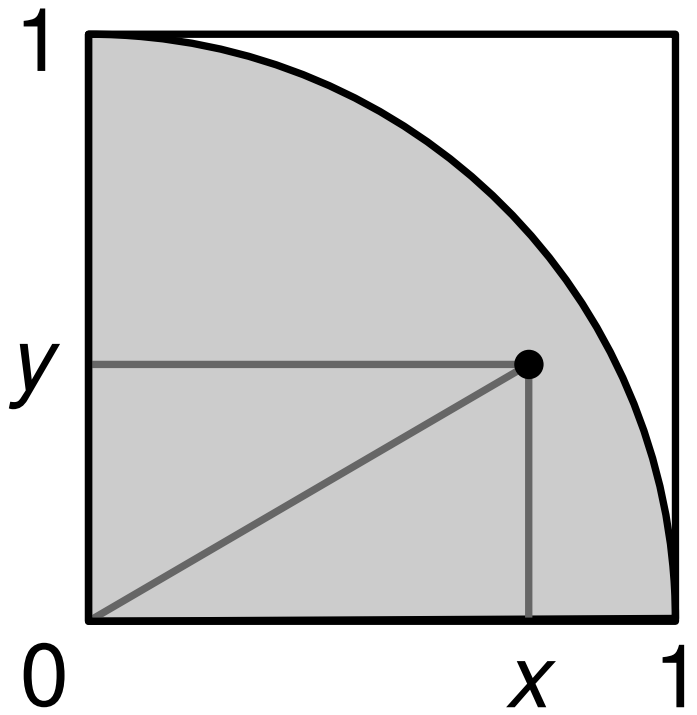
$$\pi = 4 \cdot \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \cdots$$

- Monte Carlo metoda – házení šipek do čtvrtdisku, Buffonova jehla

Výpočet π – Gregory-Leibniz

```
def gregory_leibniz(n):
    total = 0
    sign = 1
    for k in range(1, n+1):
        total += sign / (2*k-1)
        sign *= -1
    return 4*total
```

Výpočet π – Monte Carlo

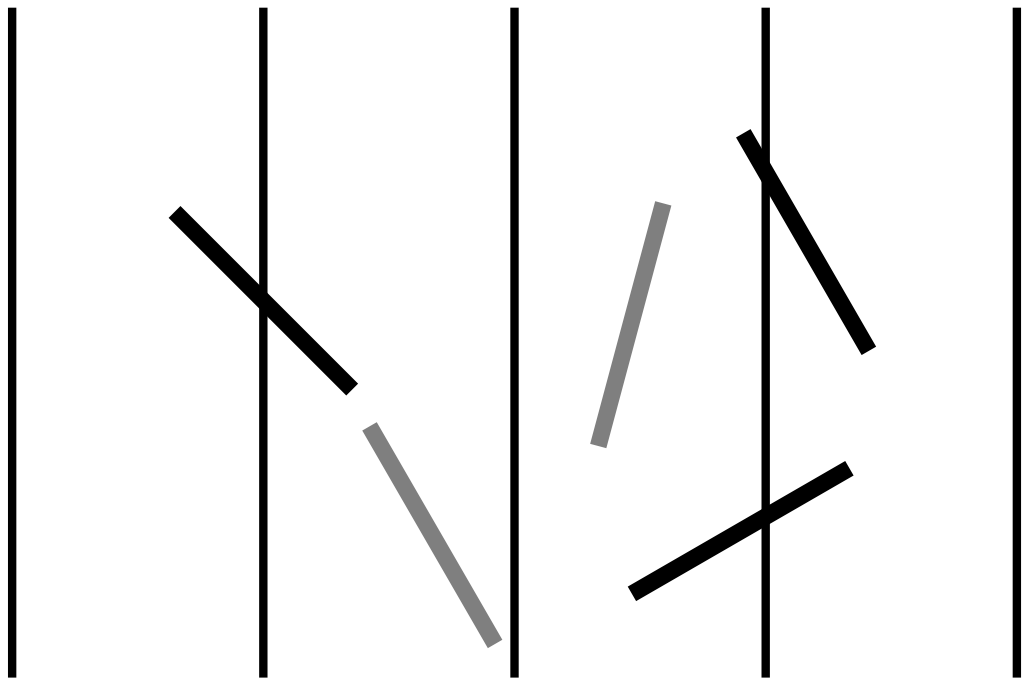


- obsah čtvrtdisku: $\pi/4$
- obsah čtverce: 1

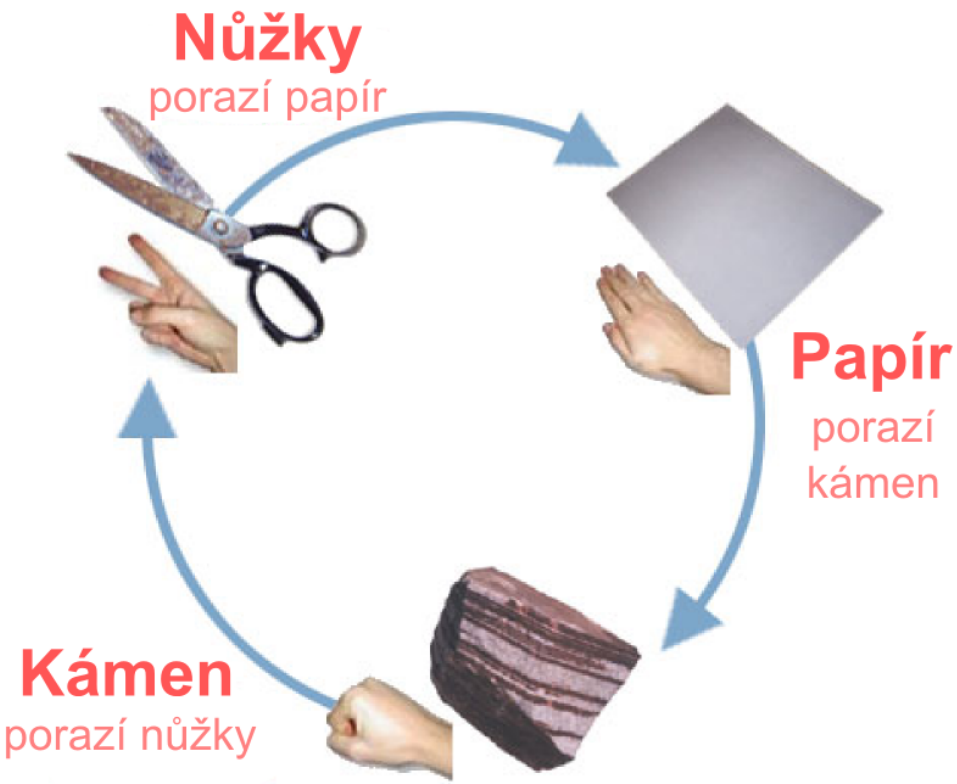
Výpočet π – Monte Carlo

```
def monte_carlo_disc(attempts):
    hits = 0
    for k in range(attempts):
        x = random.random()
        y = random.random()
        if x*x + y*y < 1:
            hits += 1
    return 4 * hits / attempts
```

Buffonova jehla



Kámen, nůžky, papír



Zdroj: Wikipedia

KNP: strategie

```
def strategy_uniform():
    r = random.randint(1, 3)
    if r == 1:
        return "R"
    elif r == 2:
        return "S"
    else:
        return "P"

def strategy_rock():
    return "R"
```

KNP: vyhodnocení tahu

```
def evaluate(symbol1, symbol2):
    if symbol1 == symbol2:
        return 0
    if symbol1 == "R" and symbol2 == "S" or \
       symbol1 == "S" and symbol2 == "P" or \
       symbol1 == "P" and symbol2 == "R":
        return 1
    return -1
```

KNP: sehrání západu

```
def rsp_game(rounds):
    points = 0
    for i in range(1, rounds+1):
        print("Round ", i)
        symbol1 = strategy_uniform()
        symbol2 = strategy_uniform()
        print("Symbols:", symbol1, symbol2)
        points += evaluate(symbol1, symbol2)
        print("Player 1 points:", points)
```

KNP: obecnější strategie

```
def strategy(weightR, weightS, weightP):
    r = random.randint(1, weightR + weightS + weightP)
    if r <= weightR:
        return "R"
    elif r <= weightR + weightS:
        return "S"
    else:
        return "P"
```

KNP: rozšiřující náměty

- turnaj různých strategií
- strategie pracující s historií
  - kopírování posledního tahu soupeře
  - analýza historie soupeře (hraje vždy kámen? $\rightarrow$ hraj papír)
- rozšíření na více symbolů (Kámen, nůžky, papír, ještěr, Spock)

Kontrolní otázky

- Co znamená „indexování od nuly“?
- Jaký je význam funkcí `chr` a `ord`?
- Jak zjistíme délku řetězce?
- Jak zjistíme, zda řetězec obsahuje znak `X`?
- Jak vypíšeme řetězec pozpátku?
- Jaký je význam sčítání řetězců? Můžeme řetězce násobit?
- Jakým způsobem vygenerujeme náhodné číslo?
- K čemu lze využít náhodná čísla?

Doporučené procvičování

`https://www.umimeinformatiku.cz/rozhodovacka`
`https://www.umimeinformatiku.cz/porozumeni`
`https://www.umimeinformatiku.cz/vystup-programu`

$\Rightarrow$ sada „Řetězce“

Shrnutí

- řetězce, znaky
- náhodná čísla
- ukázky, příklady

příště: seznamy