

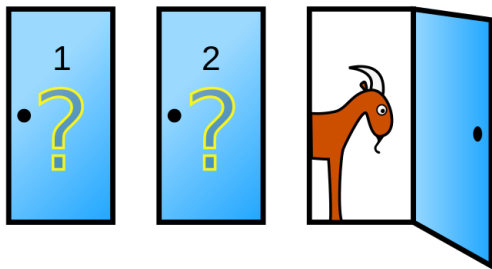
Řetězce, náhodná čísla

IB113

Radek Pelánek

2023

Rozcvička 1: Monty Hall Problem



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

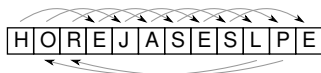
pozpátku



trojice pozpátku



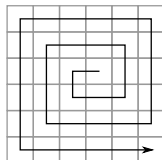
ob tři



dopředu dozadu

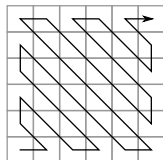


šnek



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	H → 7	+ →	25 → Z
SLONSLONSLONSL	S → 18		
ZWSQWUDBVWCCGF			

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

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

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>

- 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]      # první 3 znaky  
text[3:]      # od 3 znaku dále  
text[1:8:2]   # od 2. znaku po 7. krok po 2  
text[::-3]    # od začátku do konce po 3
```

- 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:
 - C E S K A T R E B O V A , 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 → 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: MPKTWTDVLEVELMZCF
- jak to udělat, aby program vrátil jen jednoho kandidáta?

Caesarova šifra – rozlomení

k	Kandidát	b_s	b_f	k	Kandidát	b_s	b_f
0	MPKWTVDLVELMZCF	0	21	13	ZCXGJGQIYIRYZMPS	0	-13
1	NQLUXUEWMWFMNADG	13	0	14	ADYHKHRJZJSZANQT	0	16
2	ORMVYVFXNXGNOBEH	24	9	15	BEZILISKAKTABORU	67	59
3	PSNWZWGYOYHOPCFI	5	-3	16	CFAJMJTLBLUBCPV	0	11
4	QTOXAXHZPZIPQDGJ	10	-6	17	DGBKNKUMCMVCDQW	5	-4
5	RUPYBYIAQAJQREHK	0	9	18	EHCLOLVNDNWDERUX	17	31
6	SVQZCZJBRBKRSFIL	0	3	19	FIDMPMWEOXEFVY	5	22
7	TWRADAKCSCSLTGJM	32	26	20	GJENQNXFPYFGTWZ	4	-23
8	UXSBEBLDTDMTUHKN	0	24	21	HKFOROYQGQZGHUXA	16	-17
9	VYTFCFCMEUENUVILO	11	46	22	ILGPSPZRHRAHIVYB	28	18
10	WZUDGDNFVFOVWJMP	0	-6	23	JMHQTQASISBIJWZC	9	0
11	XAVEHEOGWGPWXKNQ	5	-2	24	KNIRURBTJTCJKXAD	5	24
12	YBWFIFPHXHQXYLOR	0	-28	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:

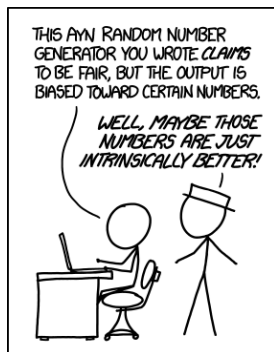
```
for i in text:  
    print(i)
```

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

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

Příklady naivních metod:

- Gregoryho-Leibnizova řada:

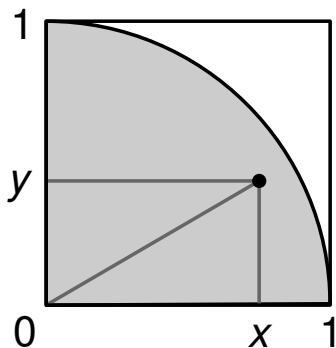
$$\pi = 4 \cdot \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \dots$$

- Monte Carlo metoda – házení šipek do čtvrtedisku, 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

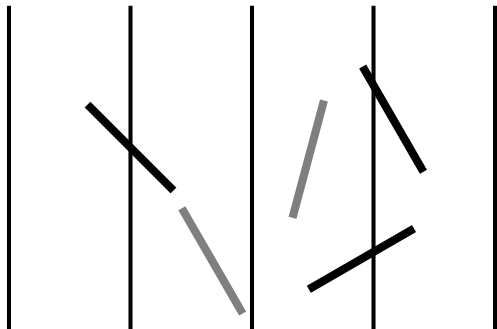


- obsah čtvrtedisku: $\pi/4$
- obsah čtverce: 1

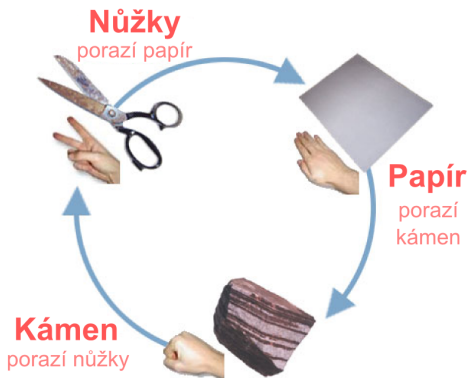
Výpočet π – Monte Carlo

```
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? → 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>

⇒ sada „Řetězce“

- řetězce, znaky
- náhodná čísla
- ukázky, příklady

příště: seznamy