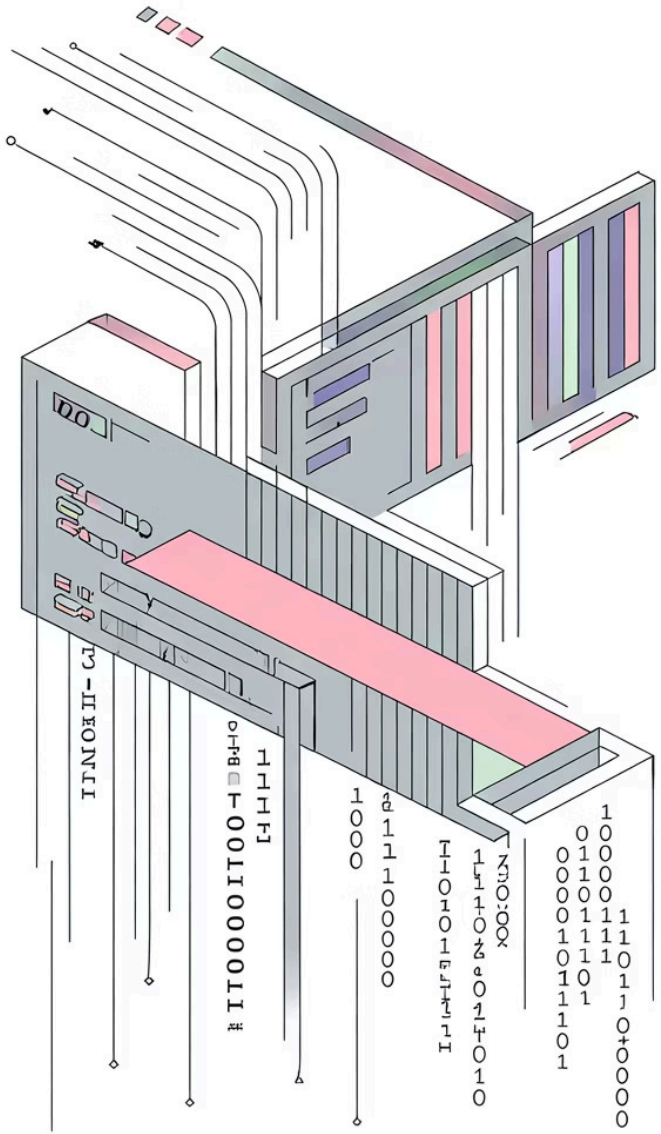


# ASCII tabulka, práce se znaky a číselné soustavy v Pythonu

V této lekci se ponoříte do fascinujícího světa znakové reprezentace v počítači a naučíte se programovat různé převody mezi znaky, dekadickými čísly a dalšími číselnými soustavami. Současně si zdokonalíte techniky zápisu a čtení dat, práci s funkcemi a ukázněné formátování výstupu do tabulek.

# Cíle této lekce

## ASCII reprezentace

Porozumět základní tabulce kódování znaků a její důležitosti v programování

## Převody znaků

Naučit se převádět znaky na čísla a naopak pomocí funkcí ord() a chr()

## Číselné soustavy

Realizovat převod mezi binární, osmičkovou a šestnáctkovou soustavou

## Formátování tabulek

Sestavovat a formátovat tabulky textově bez použití speciálních knihoven

## Parametry funkcí

Seznámit se s rozsahy a uživatelsky měnitelnými parametry

## Práce s funkcemi

Procvičit práci s funkcemi, parametry a návratovými hodnotami

# Co je ASCII a proč je důležité?

## Základní informace o ASCII

ASCII (American Standard Code for Information Interchange) je základní tabulka, která přiřazuje každému znaku jedinečné číslo. Toto kódování vzniklo v 60. letech 20. století a stalo se základem pro reprezentaci textu v počítačích.

Každé písmeno, číslo nebo speciální znak je v počítači reprezentováno nějakou číselnou hodnotou v rozsahu 0-127. To znamená, že když napíšete písmeno "A", počítač ve skutečnosti pracuje s číslem 65.

### Základní převody v Pythonu

Python poskytuje jednoduché funkce pro práci s ASCII:

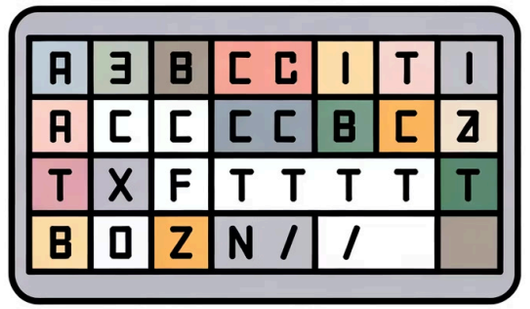
```
# Znak na číslo (ASCII kód)
ord("A")  # Výstup: 65

# Číslo na znak
chr(65)   # Výstup: 'A'
```

Tyto dvě funkce jsou klíčové pro veškerou práci s převody mezi znaky a jejich číselnou reprezentací.

### Pro českou abecedu

Pro češtinu a další jazyky s diakritikou existují rozšířená kódování jako UTF-8, která podporují mnohem více znaků než základní ASCII.



# Vytvoření jednoduché ASCII tabulky

## Základní výpis v jednom sloupci

Nejjednodušší způsob, jak vypsat ASCII tabulku, je projít všechny hodnoty v rozsahu 1-127 a pro každou z nich zobrazit číslo a odpovídající znak:

```
for i in range(1, 128):
    print(f"{i:3} {chr(i):4}")
```

### Vysvětlení formátování

- **chr(i)** - získá znak z jeho ASCII kódu
- **:3** - určuje šířku prvního sloupce (číslo) na 3 znaky
- **:4** - určuje šířku druhého sloupce (znak) na 4 znaky
- Zarovnání zajišťuje přehledný výstup

Tímto způsobem získáte základní přehled všech ASCII znaků včetně kontrolních znaků, písmen, číslic a speciálních symbolů.

## Praktický příklad výstupu

```
 65 A
 66 B
 67 C
 68 D
 69 E
...
 90 Z
 97 a
 98 b
 99 c
```

### Tip pro začátečníky

Zkuste si nejprve vypsat jen rozsah 65-90 (velká písmena) a 97-122 (malá písmena), abyste lépe pochopili strukturu ASCII tabulky.

# Převody do různých číselných soustav

V programování často potřebujeme pracovat s různými číselnými soustavami. Python nám poskytuje vestavěné funkce pro snadný převod mezi nimi.

## Binární soustava (2)

```
bin(65)
# '0b1000001'

# Bez prefixu:
bin(65)[2:]
# '1000001'
```

Binární zápis používá pouze cifry 0 a 1. Je to základ veškeré digitální technologie.

## Osmičková soustava (8)

```
oct(65)
# '0o101'

# Bez prefixu:
oct(65)[2:]
# '101'
```

Osmičková soustava byla populární v raných počítačích a stále se občas používá.

## Šestnáctková soustava (16)

```
hex(65)
# '0x41'

# Bez prefixu:
hex(65)[2:].upper()
# '41'
```

Hexadecimální zápis je velmi užitečný pro práci s barvami, pamětí a binárními daty.

## Kompletní tabulka s převody

Pro vytvoření přehledné tabulky se všemi převody použijeme následující kód:

```
for i in range(32, 128):
    print(f"{i:<5}{chr(i):<5}{bin(i)[2:]:<10}{oct(i)[2:]:<5}{hex(i)[2:].upper():<5}")
```

Tento kód vypíše každý znak spolu s jeho reprezentací v dekadické, binární, osmičkové a šestnáctkové soustavě. Začínáme od 32, protože znaky 0-31 jsou kontrolní a netisknutelné.

# Organizace kódu pomocí funkcí

## Funkce pro výpis ASCII tabulky

Pro lepší organizaci a znovupoužitelnost kódu doporučujeme vytvořit funkce, které zapouzdří logiku výpisu a převodů. Funkce umožňují parametrizaci a snadnou modifikaci chování programu.

```
def ascii_table(start=32, end=127):
    print(f"Dec  Char Bin       Oct  Hex")
    print("-" * 35)
    for i in range(start, end+1):
        dec = i
        char = chr(i)
        binary = bin(i)[2:]
        octal = oct(i)[2:]
        hexa = hex(i)[2:].upper()
        print(f"{dec:<5}{char:<5}{binary:<10}{octal:<5}{hexa:<5}")
```

### Výhody tohoto přístupu

- Flexibilní rozsah - můžete snadno změnit, které znaky chcete vypsat
- **Znovupoužitelnost** - funkci lze volat vícekrát s různými parametry
- **Čitelnost** - kód je přehlednější a srozumitelnější
- **Údržba** - změny stačí provést na jednom místě

## Příklady volání funkce

```
# Celá tabulka
ascii_table()

# Pouze velká písmena
ascii_table(65, 90)

# Malá písmena
ascii_table(97, 122)

# Číslice
ascii_table(48, 57)

# Vlastní rozsah
ascii_table(35, 64)
```

### Výchozí hodnoty parametrů

Použití výchozích hodnot (start=32, end=127) umožňuje volat funkci bez parametrů pro standardní výpis, ale zároveň zachovává flexibilitu.

# Převod znaku na zadanou soustavu

Vytvoříme univerzální funkci, která dokáže převést jakýkoliv znak do zvolené číselné soustavy. Tato funkce bude užitečná pro interaktivní práci s uživatelem.

## Implementace funkce

```
def char_to_base(char, base):
    """
    Převede znak na reprezentaci
    v zadané číselné soustavě.

    Args:
        char: znak k převodu
        base: 'bin', 'oct', nebo 'hex'

    Returns:
        řetězec s číslem v dané soustavě
    """
    value = ord(char)

    if base == 'bin':
        return bin(value)[2:]
    elif base == 'oct':
        return oct(value)[2:]
    elif base == 'hex':
        return hex(value)[2:].upper()
    else:
        return "Neplatný typ soustavy!"
```

## Ukázky použití

```
# Převod znaku # do různých soustav
print(char_to_base("#", "bin"))
# Výstup: '100011'

print(char_to_base("#", "hex"))
# Výstup: '23'

print(char_to_base("#", "oct"))
# Výstup: '43'

# Převod písmene A
print(char_to_base("A", "hex"))
# Výstup: '41'

print(char_to_base("A", "bin"))
# Výstup: '1000001'
```

Funkce nejprve převede znak na jeho ASCII hodnotu pomocí `ord()` a poté aplikuje požadovaný převod do zvolené soustavy.

Tuto funkci lze snadno rozšířit o další soustavy nebo o kontrolu vstupů, například ověření, zda je zadaný znak platný ASCII znak.

# Formátovaný vícesloupcový výstup

Pro větší přehlednost a úsporu místa můžeme vytvořit tabulku rozdělenou do více sloupců. Tento přístup je obzvláště užitečný při výpisu velkého množství dat.

```
def ascii_table_multicolumn(start=32, end=127, cols=4):
    """
    Vypíše ASCII tabulku ve více sloupcích vedle sebe.

    Args:
        start: začátek rozsahu ASCII kódů
        end: konec rozsahu ASCII kódů
        cols: počet sloupců v tabulce
    """
    # Hlavička tabulky opakovaná pro každý sloupec
    header = f"Dec  Char Bin        Oct  Hex   | "
    print(header * cols)
    print("-" * (len(header) * cols))

    # Procházíme řádky
    for i in range(start, end+1, cols):
        # V každém řádku procházíme sloupce
        for j in range(i, min(i+cols, end+1)):
            dec = j
            char = chr(j)
            binary = bin(j)[2:]
            octal = oct(j)[2:]
            hexa = hex(j)[2:].upper()
            print(f"{dec:<5}{char:<5}{binary:<10}{octal:<5}{hexa:<5}", end="| ")
    print()  # Nový řádek po dokončení všech sloupců
```

## Výhody vícesloupcového formátu

- **Kompaktní zobrazení** - více informací na obrazovce najednou
- Lepší využití prostoru - zejména na širších displejích
- **Přehlednost** - snadnější porovnání hodnot
- **Flexibilita** - počet sloupců lze snadno měnit

## Příklady volání

```
# 4 sloupce (výchozí)
ascii_table_multicolumn()

# 3 sloupce, omezený rozsah
ascii_table_multicolumn(35, 62, 3)

# 2 sloupce, pouze písmena
ascii_table_multicolumn(65, 90, 2)
```

# Praktická cvičení a rozšíření

01

### Interaktivní převodník

Upravte program tak, že uživatel zadá znak a program automaticky vypíše všechny jeho reprezentace - dekadickou hodnotu, binární, osmičkový i hexadecimální zápis. Použijte funkci input() pro získání vstupu.

02

### Obousměrný převod

Přidejte možnost zadat nejen znak, ale i číslo (ASCII kód) a program vypíše odpovídající znak spolu se všemi jeho reprezentacemi v různých soustavách. Implementujte kontrolu, zda uživatel zadal znak nebo číslo.

03

### Validace vstupů

Zaveďte důkladnou kontrolu vstupů - uživatel nesmí zadat číslo mimo platný rozsah (1-127 pro základní ASCII). Program by měl vypsat srozumitelnou chybovou zprávu a umožnit opakovaný pokus.

04

### Rozšířené informace

Rozšiřte tabulku o dodatečné informace, které by mohly být užitečné - například textovou reprezentaci netisknutelných znaků (TAB, ENTER, BACKSPACE) nebo kategorii znaku (číslo, písmeno, speciální znak).

## Příklad hlavní funkce programu

Pro srozumitelnost používejte hlavní blok, kde organizujete volání všech funkcí:

```
if __name__ == "__main__":
 import os
 os.system("cls") # Vyčištění konzole (Windows)

 # Základní tabulka
 print("=== Základní ASCII tabulka ===")
 ascii_table()

 print("\n=== Omezený výpis ===")
 ascii_table(40, 60)

 print("\n=== Více sloupců ===")
 ascii_table_multicolumn(35, 62, 4)

 # Interaktivní část
 char = input("\nZadejte znak pro převod: ")
 print(f"\nBinárně: {char_to_base(char, 'bin')}")
 print(f"Osmičkově: {char_to_base(char, 'oct')}")
 print(f"Hexadecimálně: {char_to_base(char, 'hex')}")
```

# Kontrolní otázky a shrnutí

## Otázky k zamyšlení

### 1 Co znamená ASCII a proč je důležité?

ASCII je standardní systém pro kódování znaků, který přiřazuje každému znaku jedinečné číslo. Je základem pro reprezentaci textu v počítačích a komunikaci mezi systémy.

### 2 Jak převádíte znaky na čísla v Pythonu?

Používáme funkce `ord()` pro převod znaku na číslo a `chr()` pro převod čísla zpět na znak.

### 3 Které číselné soustavy se v praxi používají?

Binární (základ digitálních systémů), osmičková (práce s právy v Unixu), hexadecimální (barvy, paměť, ladění) a samozřejmě dekadická soustava.

## Klíčové poznatky lekce

### Co jste se naučili

- Porozumět reprezentaci znaků v počítači pomocí ASCII
- Používat funkce ord() a chr() pro převody
- Převádět čísla mezi různými číselnými soustavami
- Formátovat výstup do přehledných tabulek
- Organizovat kód pomocí funkcí s parametry
- Pracovat s rozsahy a kontrolovat vstupy

## Další kroky

Procvičte si tyto koncepty na vlastních projektech. Zkuste vytvořit komplexnější aplikaci, která kombinuje všechny naučené techniky - například ASCII art generátor nebo šifrovací program založený na posunutí ASCII kódů.