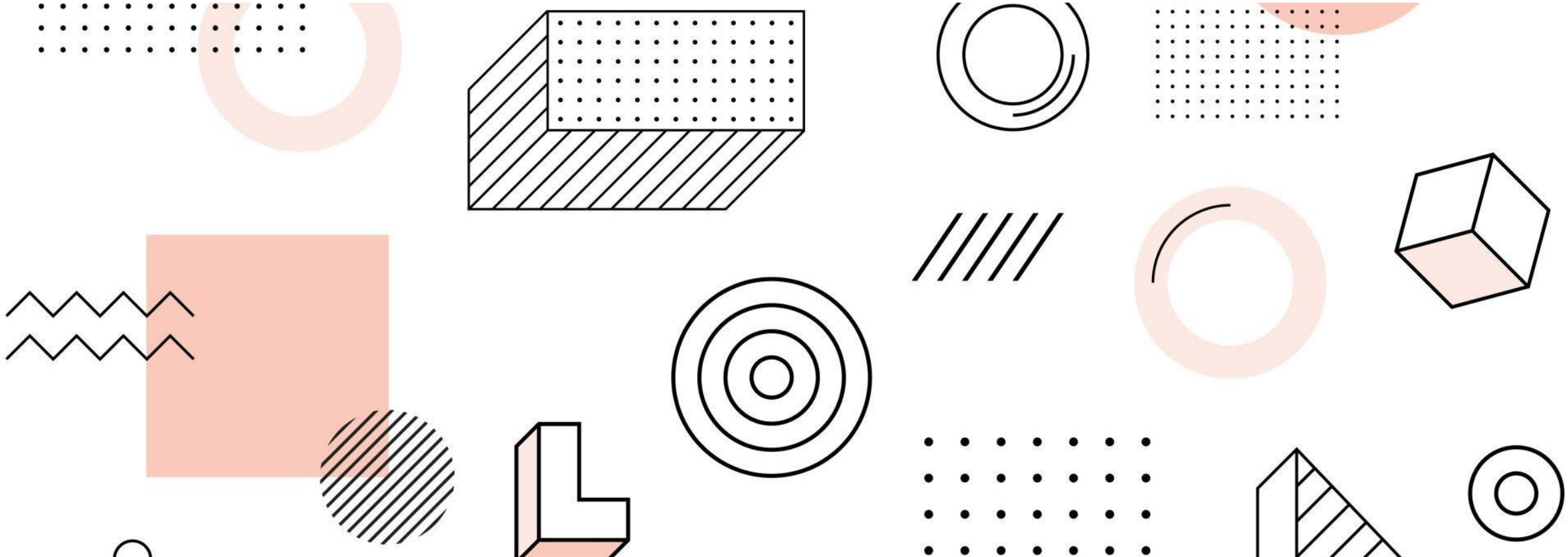
PYTHON PRO INŽENÝRSKÉ VÝPOČTY

Týden 5. Funkce - pokračování, struktura programu



Tak začneme!

Dnes budeme prohlubovat znalostí funkcí v Pythonu.

- **Argumenty:** podíváme se podrobněji na různé typy argumentů funkcí, včetně pozičních argumentů, argumentů klíčových slov, výchozích argumentů a argumentů proměnné délky.
- **Lambda funkce**: tzv. *lambda*, nebo-li anonymní funkce jsou používány pro jednoduché jednořádkové definice funkcí.
- Proč používáme funkci main()?

ARGUMENTY

Hlavní typy argumentů funkcí:

- **Poziční argumenty** - jsou definovány svou pozicí a pořadím. Při volání funkce musí být zadány ve stejném pořadí, v jakém jsou definovány.
- **Klíčová slova** - jsou identifikovány jménem parametru, za kterým následuje symbol **=** a hodnota. Umožňují předávání argumentů funkci uvedením jména parametru bez ohledu na jejich pozici.
- **Výchozí argumenty** - jsou parametry funkce s předdefinovanými hodnotami. Není-li při volání funkce uvedena žádná hodnota výchozího argumentu, použije se výchozí hodnota uvedená v definici.
- **Argumenty proměnné délky** - Argumenty proměnné délky umožňují funkci přijmout libovolný počet argumentů.
  - *args umožňuje předat funkci proměnný počet pozičních argumentů jako tuple.
  - **kwargs umožňuje předat funkci proměnný počet argumentů klíčových slov ve formě slovníku.

```
# Function with positional arguments
def greet(name, age):
    print("Hello, {}! You are {} years old.".format(name,
age))

greet("Alice", 25)

# Function with keyword arguments
def greet(name, age):
    print("Hello, {}! You are {} years old.".format(name,
age))

greet(name="Bob")

# Function with default arguments
def greet(name, age=30):
    print("Hello, {}! You are {}
years old.".format(name, age))

greet("Bob", 25)

# Function with multiple arguments
def calculate_sum(*args):
    total = sum(args)
    print("The sum is:", total)

calculate_sum(1, 2, 3, 4, 5)

# Function with keyword arguments
def print_details(**kwargs):
    for key, value in kwargs.items():
        print(key + ": " + value)

print_details(name="Alice", age="25", city="London")
```

LAMBDA FUNKCE

Lambda funkce, nebo tzv. anonymní funkce, jsou malé jednořádkové funkce, které nevyžadují příkaz `def` ani pojmenovanou funkci. Obvykle se používají, když je potřeba malá funkce na krátkou dobu nebo jako parametr jiné funkce.

```python
# Adding two numbers
add = lambda x, y: x + y
result = add(3, 4)
print(result) # Output: 7



# Squaring numbers
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x ** 2, numbers))
print(squared) # Output: [1, 4, 9, 16, 25]
```

VOLÁNÍ FUNKCE

Chcete-li zavolat funkci, použijte její název následovaný závorkami. Pokud funkce přijímá nějaké argumenty, můžete je předat uvnitř závorek. Funkcím vrací více hodnot jako *tuple*, které lze při volání funkce rozbalit do jednotlivých proměnných.

```
#Define a function that adds two
numbers
def add_and_multiply(x, y):
    sum = x + y
    prod = x * y
    return sum, prod



# Call the function and print the
result
s, p = add_numbers(3, 4)
print("The sum is: ", s)
print("The product is: ", p)
```

FUNKCE MAIN()

Podle konvence se funkce **main()** často volá na konci skriptu. Tím je zajištěno, že se funkce *main()* vykoná pouze při přímém spuštění skriptu. Když je skript Pythonu importován jako modul, funkce *main()* se automaticky nespustí. To umožňuje importovat funkce a třídy definované ve skriptu a používat je v jiných modulech, aniž by se spustila hlavní logika.

```
def main():
    # Main logic of the program
    return



# Execute the main function if the script is run directly
if __name__ == "__main__":
    main()
```