Functions & Modular Code

Python Course · Module 5

ElnurBDa

Functions — Introduction

Functions group reusable code into named blocks.

Benefits:

- Avoid repetition
- Improve readability
- Easier debugging
- Modular structure

Basic structure:

```
def name():
    pass
```

Creating Functions

```
def greet():
    print("Hello!")
```

Call:

```
greet()
```

Functions must be defined **before** calling.

Parameters

```
def greet(name):
    print("Hello,", name)
```

Multiple parameters:

```
def add(a, b):
    return a + b
```

Return Values

```
def square(x):
    return x * x
```

Functions without `return` return `None`.

Multiple Returns

```
def values():
    return 10, 20, 30
```

Unpacking:

```
a, b, c = values()
```

Variable Scope

```
x = 10      # global

def func():
    y = 5   # local
```

Local variables override global ones inside functions.

---

global Keyword

```
count = 0

def inc():
    global count
    count += 1
```

Use sparingly; global state makes code harder to manage.

Lambda Functions

```
square = lambda x: x * x
```

Often used inline:

```
add = lambda a, b: a + b
```

Higher-Order Functions

```
nums = [1, 2, 3, 4]

squares = list(map(lambda x: x*x, nums))
```

Using normal function:

```
def sq(x): return x*x
```

Modules — Importing

```
import math
math.sqrt(16)
```

Selective:

```
from math import sqrt
```

Rename:

```
import math as m
```

Built-in Modules

Examples:

- math
- random
- os
- sys
- datetime

Example:

```
import random
x = random.randint(1, 10)
```

Creating Your Own Module

File:

```
mymath.py
```

Content:

```
def add(a, b):
    return a + b
```

Usage:

```
import mymath
mymath.add(2, 3)
```

Organizing Code

```
project/
├── main.py
├── utils.py
└── math_ops.py
```

```python
from utils import greet
from math_ops import add
```

Example

```python
# utils.py
def greet(name):
    return f"Hello {name}"

# math_ops.py
def multiply(a, b):
    return a * b

# main.py
from utils import greet
from math_ops import multiply

print(greet("Elnur"))
print(multiply(4, 5))
```

Mini Task

Create a project with:

1. main.py
2. converter.py with:
   - km_to_miles(km)
   - celsius_to_fahrenheit(c)
3. Import into main.py
4. Ask user input and print converted values
