Branching, iterations, and functions in Python

Class 3

04/2/2025

Acknowledgement

The series of the IT & Japanese language course is Supported by AOTS and OEC.

Ministry of Economy, Trade and Industry

Overseas Employment Corporation

What you have Learnt Last Week

We were focused on following points.

- Introduction of Python, Variables & Operators
- Data Types & String Manipulation
- Introduction of Google colab and Jupyternotebook
- Basic Python Syntax and Data Types
- Lower, upper, length (len), random and split function
- Random module
- Upload code on github
- Quiz
- Q&A Session

What you will Learn Today

We will focus on following points.

- Usage of control and loop flow statement
- How to define and use functions
- Usage of lambda function
- Upload code on github
- Quiz
- Q&A Session

Upload code on Github

How you can upload your Python code to GitHub?

[Create a GitHub Repository]

- Go to GitHub and log in.
- Click on the "+" icon in the top-right and select "New repository".
- Create a repository (e.g., Class3-Assignment).
- Choose Public or Private.

[Upload your Code]

- Open Git Bash (Download git)
- Go to code folder
- git add .
- git commit –m “Files are added”
- git push origin main
- Write username and password

Control Flow Statements in Python

These statements allow a program to execute different code blocks based on conditions

[Conditional Statements]

- if statement
- if-else statement
- if-elif-else statement
- Nested if statements

Python: A language for everyone, from beginners to experts!"

if statement

Executes a block of code if a condition is true. if Statement

[Syntax]

- **if condition:**
  # Code to execute if condition is true

[Example]

```
age = 18
if age >= 18:
    print("You are eligible to vote.")
```

if-else Statement

Executes one block of code if a condition is true and another block if it is false.

[Syntax]

```
if condition:
    # Code to execute if condition is true
else:
    #Code to execute if condition is false
```

[Example]

```
num = 5
if num % 2 == 0:
    print("Even number")
else:
    print("Odd number")
```

if-elif-else Statement

Handles multiple conditions

[Syntax]

```
if condition1:
  # Code for condition1
elif condition2:
  # Code for condition2
else:
   # Code if none of the above conditions are true
```

[Example]

```
marks = 85
if marks >= 90:
  print("Grade: A+")
elif marks >= 80:
  print("Grade: A")
elif marks >= 70:
  print("Grade: B")
else:
  print("Grade: C")
```

Nested if Statements & Logical Operators

Allows an if statement inside another if statement for complex conditions

[Example]

```
num = 15
if num > 10:
  if num % 2 == 0:
    print("Even number greater than 10")
  else:
    print("Odd number greater than 10")
```

Logical Operators

Combines multiple conditions in a single if statement

[Example]

```
is_adult = True
has_voter_id = False
if is_adult and has_voter_id:
    print("You can vote.")
else:
    print("You cannot vote.")
```

Ternary Operator

Shorthand way of writing a single-line if-else statement

[Syntax]

variable = value_if_true if condition else value_if_false

[Example]

```
age = 20
message = "Adult" if age >= 18 else "Minor"
print(message)
```

match-case Statement

Simplifies multiple condition handling, replacing if-elif chains

[Syntax]

```
match variable:
  case value1:
    # Code for value1
  case value2:
    # Code for value2
  case _:
    # Code for all other cases
```

[Example]

```
color = "red“
match color:
    case "red":
        print("Stop")
    case "yellow":
        print("Ready")
    case "green":
        print("Go")
    case _:
        print("Invalid color")
```

Loop Flow Statements in Python

These statements are used to execute a block of code multiple times until a specified condition is met.

[Loop Flow Statements]

- For Loop
- While Loops
- Break Statement
- Continue Statement
- Pass Statement
- Else Statement in Loops
- Nested Loop

For Loop

A for loop is used when the number of iterations is known beforehand.
Iterating over sequences (lists, strings, tuples, dictionaries)

[Example]

```
# Iterating over a list
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)

# Iterating over a string
for char in "Python":
    print(char)
```

[Example]

```
# Iterating over a dictionary
person = {"name": "Alice",
                "age": 25,
                 "city": "New York"}
for key, value in person.items():
    print(f"{key}: {value}")
```

For Loop

Using range() in for Loops

[Example]

```
# Iterating through a range of numbers
for i in range(5):
    print(i)  # Outputs: 0, 1, 2, 3, 4

# Custom range with start, stop, and step
for i in range(1, 10, 2):
    print(i)  # Outputs: 1, 3, 5, 7, 9
```

While Loop

When looping based on a condition. Number of iterations is not known beforehand

[Example]

```
count = 0
while count < 5:
    print(f"Count: {count}")
    count += 1
```

[Example]

```
#Infinite Loops and Breaking Conditions:
while True:
    user_input = input("Enter 'quit' to exit: ")
    if user_input == "quit":
        print("Exiting loop.")
        break
```

Loop Control Statement

Exiting a loop prematurely, Skipping the current iteration and Placeholder Statement

[Example]

- Break statement
- Continue Statement
- Pass Statement

Loop Control Statement

Break statement, Continue Statement and Pass Statement

[Example]

```
#Usage of break statement
for i in range(10):
    if i == 5:
        print("Breaking the loop")
        break
    print(i)
```

[Example]

```
#continue statement
for i in range(10):
    if i % 2 == 0:
        continue
    print(i)  # Outputs only odd numbers
```

[Example]

```
#pass statement
for i in range(5):
    if i == 3:
        pass  # Placeholder, does nothing
    print(f"Processing {i}")
```

Break vs Continue vs Pass Statements

Break statement, Continue Statement and Pass Statement

Statement	Effect
pass	Does nothing, placeholder
break	Exits the loop immediately
continue	Skips the current iteration and continues

Nested Loop

Looping inside loops

[Example]

```
#Looping inside loop
for i in range(3):
    for j in range(2):
        print(f"i = {i}, j = {j}")
#Output will be 3x2 matrix
```

(0,0) (0,1)
(1,0) (1,1)
(2,0) (2,1)

[Example]

```
#Practical Example: Multiplication Table
for i in range(1, 6):
    for j in range(1, 6):
        print(f"{i} x {j} = {i * j}")
    print()
```

Loop Else Statement

The else block executed because the loop was not interrupted

The else block did NOT execute because break was encountered

[Example]

```
#Using else with Loops
for num in range(5):
    print(num)
else:
    print("Loop completed successfully.")
```

[Example]

```
#Practical Example: Multiplication Table
for num in range(5):
    if num == 3:
        print("Breaking the loop")
        break
    print(num)
else:
    # This won't execute because of `break`
     print("Loop completed without break.")
```

Loop Else Statement

Real world example: Searching for a Number

[Example]

```
nums = [1, 2, 3, 4, 5]
search = 7

for num in nums:
    if num == search:
        print(f"Found {search}!")
        break
else:
    print(f"{search} not found in the list.")
```

Define and Use Functions in Python

A function in Python is a block of reusable code that performs a specific task

[Basic Syntax]

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

[Adding Parameters]

```
#Adding Parameters
def greet_user(name):
        print(f"Hello, {name}!")
```

Calling Function

How to call function?

[Without Parameters]

```
def say_hello():
    print("Hi there!")
say_hello()
# Output: Hi there!
```

[Passing Arguments]

```
def add(a, b):
    return a + b
print(add(3, 5))
# Output: 8
```

[Handling Multiple Arguments]

```
def display_info(name, age):
    print(f"Name: {name}, Age: {age}")
display_info("Alice", 25)
# Output: Name: Alice, Age: 25
```

Return Statement

Returning single and multiple values

[Single Value]

```
def square(num):
    return num * num
print(square(4))
# Output: 16
```

[Multiple Values with tuples]

```
def calculate(a, b):
    return a + b, a - b
sum_, diff = calculate(10, 5)
print(sum_, diff)
# Output: 15 5
```

No Return Statement

If a function in Python does not have a return statement, it returns None by default

[Example]

```
def my_function():
    pass  # No return statement

result = my_function()
print(result)  # Output: None
```

This confirms that the default return value is *None*

Function Parameter

Default and Positional Argument

[Positional Arguments]

```
def greet(name, greeting):
    print(f"{greeting}, {name}!")
greet("Alice", "Hello")
# Output: Hello, Alice!
```

[Default Arguments]

```
def greet(name, greeting="Hi"):
    print(f"{greeting}, {name}!")
greet("Alice")
# Output: Hi, Alice!
```

Function Parameter

Keyword and arbitrary arguments

[Keyword Arguments]

```
def greet(name, greeting):
    print(f"{greeting}, {name}!")
greet(name="Bob", greeting="Hey")
# Output: Hey, Bob!
```

[Arbitrary Arguments]

```
def print_args(*args):
    for arg in args:
        print(arg)
print_args(1, 2, 3)
# Output: 1 2 3

def print_kwargs(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")
print_kwargs(name="Alice", age=25)
# Output:
# name: Alice
# age: 25
```

Anonymous Function

Using lambda for One-Liner Functions

[lambda Function]

```
square = lambda x: x ** 2
print(square(5))
# Output: 25
```

[Difference Between lambda and def]

- lambda is for small, simple functions.
- def allows for complex, multi-line definitions.

Quiz Section

Quiz

Everyone student should click on submit button before time ends otherwise MCQs will not be submitted

[Guidelines of MCQs]

1. There are 20 MCQs
2. Time duration will be 10 minutes
3. This link will be share on 6:10pm (Pakistan time)
4. MCQs will start from 6:15pm (Pakistan time)
5. This is exact time and this will not change
6. Everyone student should click on submit button otherwise MCQs will not be submitted after time will finish
7. Every student should synchronize there laptop clock with actual time otherwise they cannot solve the MCQs

Q&A Session

ありがとうございます。
Thank you.
شکریا

For the World with Diverse Individualities