ML Job Ready Course

Week 1 Day 2

Md. Mehedi Hasan
Senior Machine Learning Engineer
Minerva Analytics Inc

Python Control Flow and Loops - Practice Sheet

Following Python Crash Course by Eric Matthes

Learning Objectives

By the end of this practice sheet, you'll be able to:

- Use conditional statements to make decisions in your code
- Work with comparison and logical operators
- Create loops to repeat code efficiently
- Control loop execution with break, continue, and pass
- Use range() and enumerate() functions effectively

---

1. Conditional Statements (if, elif, else)

Basic if Statement

```
age = 19
if age >= 18:
    print("You are old enough to vote!")
```

if-else Statement

```
age = 17
if age >= 18:
    print("You are old enough to vote!")
else:
    print("Sorry, you are too young to vote.")
```

if-elif-else Chain

```
age = 12
if age < 4:
    price = 0
elif age < 18:
    price = 25
else:
    price = 40

print(f"Your admission cost is ${price}.")
```

2. Comparison Operators

Equality and Inequality

```
car = 'bmw'
print(car == 'bmw')   # True
print(car == 'audi')  # False
print(car != 'audi')  # True
```

Case-Sensitive Comparisons

```
car = 'Audi'
print(car == 'audi')        # False
print(car.lower() == 'audi')  # True
```

Numerical Comparisons

```
age = 18
print(age == 18)   # True
print(age != 18)   # False
print(age < 21)    # True
print(age <= 18)   # True
print(age > 16)    # True
print(age >= 18)   # True
```

Try It Yourself 2.1: Conditional Tests

Create at least 10 tests with True and False results:

```
# Example tests
car = 'subaru'
print("Is car == 'subaru'? I predict True.")
print(car == 'subaru')

print("Is car == 'audi'? I predict False.")
print(car == 'audi')

# Write 8 more tests here
# Include tests for strings, numbers, and lists
```

---

3. Logical Operators

and Operator

```
age_0 = 22
age_1 = 18
print(age_0 >= 21 and age_1 >= 21)  # False
```

print(age_0 >= 21 and age_1 >= 18) # True

or Operator

age_0 = 22

age_1 = 18

print(age_0 >= 21 or age_1 >= 21) # True

print(age_0 >= 25 or age_1 >= 25) # False

4. Sequence Learning

In Python, a sequence is an ordered collection of items, where each item is stored at a specific index starting from 0. Sequences support operations like:

- Indexing (accessing items by position)
- Slicing (extracting a subpart)
- Iteration (looping through items)
- Length checking with len()
- Membership testing using in
- Some support concatenation and repetition

Mutable = Changeable (যা ঘটার পরেও পরিবর্তন যোগ্য. Example: তরকারিতে লবণ বাড়ানো : রান্না চলছে, একটু কম লবণ হয়েছে — তুমি চাইলেই এখন লবণ বাড়াতে পারো)
Immutable = Unchangeable (যা ঘটার পরে আর কোনো দিন পরিবর্তন করা যায় না, Example: কাঁচা ডিম ফেটিয়ে ভাজি করে ফেললে, তুমি আবার সেটা কাঁচা ডিম বানাতে পারবে না

Examples of immutable types:

- int
- float
- str
- tuple
- bool
- frozenset ex: frozenset([1, 2, 3])
- bytes ex: bytes ([65, 66, 67]) #is a **built-in data type** used to store **binary data**

Here is a **list of mutable data types** in Python:

- list
- dict (dictionary)
- set

- bytearray #data = bytearray([65, 66, 67]) # The bytearray is a **mutable sequence of bytes**.

---

for Loops in Python

Theory:

A for loop in Python is used to iterate over a sequence (like a list, tuple, string, etc.) and perform actions for each item in the sequence.

It automatically selects each item from the sequence one by one and allows you to use it within the loop body.

Basic Syntax:

```
for variable in sequence:
    # code block (loop body)
```

```
# Define a list of fruits
fruits = ["apple", "banana", "cherry", "date"]
# Use a for loop to iterate through each fruit in the list
for fruit in fruits:
    # Print the current fruit
    print(fruit)
```

---

5. Using range() Function

Theory:

The range() function generates a sequence of numbers, which is often used in for loops or to create number lists.

Syntax of range()

range(start, stop, step)

start → (optional) the starting number (default is 0)

stop → (required) the end value (but not included in result)

step → (optional) how much to increase each time (default is 1)

1. Simple range() in a for loop

```
for value in range(1, 5):
    print(value)
```

Explanation: range(1, 5) gives numbers from 1 to 4 (5 is excluded).

2. Creating a list using range()

numbers = list(range(1, 6))

print(numbers)

Explanation: list(range()) converts the number sequence into a list.

Simple Statistics

digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]

print(min(digits))

print(max(digits))

print(sum(digits))

---

6. List Comprehensions (বোধগম্যতা)

[expression for item in iterable]

expression: What you want to do with each item (e.g., value**2)

item: A variable representing each element in the iterable.

iterable: A sequence like range(), list, tuple, etc.

Normal:

```
squares = []
for value in range(1, 11):
   squares.append(value**2)

print(squares)  # [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

comprehensions:

```
squares = [value**2 for value in range(1, 11)]
print(squares)  # [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Some idea:

1. Convert words to uppercase from a list.
2. Create a list of first letters.

---

7. While Loops

Theory:

A while loop repeats a block of code as long as a condition is True.

Unlike a for loop (which runs a fixed number of times), a while loop is useful when you don't know in advance how many times to repeat — for example, when waiting for user input.

Sample of increment and decrement

```
x = 5
x = x + 1  # Increment by 1
print(x)  # Output: 6

x = 5
x += 1  # Same as x = x + 1
print(x)  # Output: 6
```

for decrement use – instead of +

Basic Syntax:

```
while condition:
    # Code block (runs repeatedly as long as condition is True)
```

1. Simple while Loop

```
current_number = 1
while current_number <= 5:
    print(current_number)
    current_number += 1
```

---

8. Loop Control: break, continue, pass

Using break

```
prompt = "\nPlease enter the name of a city you have visited:"
prompt += "\n(Enter 'quit' when you are finished.) "

while True:
    city = input(prompt)
    if city == 'quit':
        break
    else:
        print(f"I'd love to go to {city.title()}!")
```

Using continue

```
current_number = 0
while current_number < 10:
    current_number += 1
    if current_number % 2 == 0:
        continue
    print(current_number)
```

Using pass

```
for number in range(10):
  if number < 5:
    pass  # Placeholder - do nothing for now
  else:
    print(number)
```

---

9. enumerate() Function

An enumerate object, which yields pairs of (index, value) when iterated over.
Why use enumerate()?
Normally, if you want both index and value from a list, you'd write:

Basic enumerate()

```
fruits = ['apple', 'banana', 'orange']
for index, fruit in enumerate(fruits):
  print(f"{index}: {fruit}")
```

Starting enumerate() at Different Number

```
fruits = ['apple', 'banana', 'orange']
for index, fruit in enumerate(fruits, start=1):
  print(f"{index}. {fruit}")
```