Programming with Python - Open Elective

March 07, 2025

Revision: variables (containers for data), input(), print(),
datatype: int, float, bool, str, None
arithmetic, relational and logical operator, operator associativity, operator precedence (arithmetic > relational > logical)
if-else, for loop, while loop,
strings, string indexing, string slicing
function, function with arguments, function returning values

Functions, break and continue operations

When writing loops in Python (such as for and while), sometimes we need to stop the loop early or skip certain iterations. This is where the break and continue statements come in.

`break` statement: The `break` statement **immediately exits the loop** when encountered, skipping all remaining iterations.

```
#Stop the loop when a vowel is found
sentence = 'Cn nt wrt wtht using vwl'

for ch in sentence:
    if ch 'aeiou' :
        print("vowel found! Stopping the loop.")
        break  # Stop the loop immediately
    print(ch)
```

`continue` statement: The `continue` statement **skips the current iteration** and moves to the next one, without exiting the loop.

```
#Skip printing even numbers

numbers = '12345'

for num in numbers:

    if int(num) % 2 == 0:

        continue  # Skip this iteration if the number is even

    print(num)
```

Practice Examples:

1. Inflation impact(for loop): Inflation increases the price of a product by 5% per year. Write a Python program that takes an initial price and calculates the price for the next 5 years.

   Input:

   Enter initial price: 1000

   Output:

   Year 1: 1050.0

   Year 2: 1102.5

   Year 3: 1157.63

   Year 4: 1215.51

   Year 5: 1276.28

```
initial_price = float(input("Enter initial price: "))

for year in range(1, 6):  # 5 years

    initial_price *= 1.05  # Increase price by 5%

    print(f"Year {year}: {initial_price:.2f}")
```

2. Loan Repayment (while loop)

A person takes a loan of Rs 10,000 and repays Rs 1,500 per year.

Use a while loop to calculate how many years it will take to repay the full loan.

Output:

```
Year 1: Remaining Loan = 8500
Year 2: Remaining Loan = 7000
Year 3: Remaining Loan = 5500
Year 4: Remaining Loan = 4000
Year 5: Remaining Loan = 2500
Year 6: Remaining Loan = 1000
Year 7: Loan fully repaid!
```

```
loan_amount = 10000
year = 0
repayment = 1500


while loan_amount > 0:
    year += 1
    loan_amount -= repayment
    if loan_amount > 0:
        print(f"Year {year}: Remaining Loan = {loan_amount}")
    else:
        print(f"Year {year}: Loan fully repaid!")
```

3. GDP Growth Rate Analysis (for and if-else)

   Given a list of GDP growth rates, identify whether each year had positive growth or a recession (negative growth).

   For the GDP growth rates [-2.3, 3.5, 1.8, -0.5, 2.0], print:

   "Recession" if the rate is negative

   "Growth" if the rate is positive

   Output:

   ```
   Year 1: Recession
   Year 2: Growth
   Year 3: Growth
   Year 4: Recession
   Year 5: Growth
   ```

```
gdp_growth_rates = [-2.3, 3.5, 1.8, -0.5, 2.0]


for year, rate in enumerate(gdp_growth_rates,
start=1):
    if rate < 0:
        print(f"Year {year}: Recession")
    else:
        print(f"Year {year}: Growth")
```

4. Extract Country Names from Economic Reports (strings)

An economist has a list of reports with country names: "GDP_Report_USA_2024", "Inflation_Report_UK_2023", "Trade_Report_India_2025".

Write a Python program that extracts and prints the country names from these strings.

Output:

USA

UK

India

```
reports = ["GDP_Report_USA_2024",
"Inflation_Report_UK_2023", "Trade_Report_India_2025"]


for report in reports:
    parts = report.split("_")  # Split by underscore
    country = parts[-2]  # Extract second last element
    print(country)
```

Exercises:

Notes:

Practice these built-in functions:

Strings: split, join, replace, strip, len, count