Introduction to Programming and Data Structures

Python – Control Flow

Malay Bhattacharyya

Associate Professor

MIU, CAIML, TIH
Indian Statistical Institute, Kolkata

August, 2023

Conditional – if-else

```
if <Condition>:
    statement 1
    statement 2
else:
    statement 3 # Execute if Condition fails
```

Note: Boundary of the conditional block is demarcated by indentation.

Iterative – if-elif-else

```
if <Condition 1>:
    statement 1
elif <Condition 2>:
    Statement 2
else:
    statement 3 # Execute if Condition 1 and 2 fails
```

Note: Boundary of the conditional block is demarcated by indentation.

Iterative – `for` loop

```
for <variable> in <container>:
    statement 1
    statement 2
```

Note: Boundary of the iterative block is demarcated by indentation.

Iterative – `for` loop

We can create a list of consecutive integers using the `range()` function as follows.

```
for <variable> in range(<value>):
    statement 1
    statement 2
```

- `range(x)` returns a list whose items are consecutive integers from [0, x).
- `range(x, y)` returns a list (feasible when x $<$ y) whose items are consecutive integers from [x, y).
- `range(x, y, step)` returns a list of integers from [x, y), such that the difference between each two adjacent items in the list is step. If step is less than 0, it counts down from x to y. If step equals 0, it raises an exception.

Iterative – `for` loop

As a popular practice, you can use a general purpose throwaway variable (wildcard character) as the loop variable if it is not to be used iteratively. However, they are still accessible within the `for` block.

```
for i in range(3):
    print(i)
for _ in range(3):
    print('Hi', _)
```

Output:

```
0
1
2
Hi 0
Hi 1
Hi 2
```

Efficient `for` loop

```
list = []
for i in range(6):
    list.append(i*2)
print(list)
```

Efficient `for` loop

```
list = []
for i in range(6):
    list.append(i*2)
print(list)
```

Output: [0, 2, 4, 6, 8, 10]

Efficient for loop

```
list = []
for i in range(6):
    list.append(i*2)
print(list)
```

Output: [0, 2, 4, 6, 8, 10]

```
list = [i*2 for i in range(6)]
print(list)
```

Efficient for loop

```
list = []
for i in range(6):
    list.append(i*2)
print(list)
```

Output: [0, 2, 4, 6, 8, 10]

```
list = [i*2 for i in range(6)]
print(list)
```

Output: [0, 2, 4, 6, 8, 10]

Iterative – `while` loop

```
while <Condition>:
    statement 1
    statement 2
```

Note: Boundary of the iterative block is demarcated by indentation.

break and continue

- **break:** Immediately jump to the next operation after the loop
- **continue:** Do the operation, if applicable, and continue with the next iteration of the loop

break and continue

- **break:** Immediately jump to the next operation after the loop
- **continue:** Do the operation, if applicable, and continue with the next iteration of the loop

```
for i in range(1, 100):
    print(i)
    if i%10 != 0:
        break
```

Prints only 1

break and continue

- **break:** Immediately jump to the next operation after the loop
- **continue:** Do the operation, if applicable, and continue with the next iteration of the loop

```
for i in range(1, 100):
    print(i)
    if i%10 != 0:
        break
```

Prints only 1

```
i = 0
while i < 100:
    i = i+1
    if i%10 != 0:
        continue
    print(i)
```

Prints 10, 20, ..., 100

Let's try solving a problem

Given a positive integer n as user input, find out the number of trailing zeros in $n!$.

Let's try solving a problem

Given a positive integer n as user input, find out the number of trailing zeros in $n!$.

Hint: This can be done with $\log_5 n$ number of divisions.

Let's try solving a problem

Given a positive integer n as user input, find out the number of trailing zeros in $n!$.

Hint: This can be done with $\log_5 n$ number of divisions.

```
n = input("Enter n: ")
n = int(n)
count = 0
while n:
    n //= 5
    count += n
print('Number of trailing zeros:', count)
```

Problems

1. An n-digit number is `SPECIAL` if the addition of its sum of the digits and the product of its digits equals to the original number. E.g., 19 is a `SPECIAL` 2-digit number. Write a program to verify whether a given number is `SPECIAL` or not. Extend this program to verify whether there exists any `SPECIAL` number for a given value of number of digits n.
2. Consider an n-digit number. Square it and add the right n digits to the left n or $n-1$ digits. If the resultant sum is same as the original number, then it is called a `Kaprekar` number. E.g., 45 is a `Kaprekar` number. Write a program to verify whether a given number is `Kaprekar` or not.

Problems

3. Let us define a string, comprising English alphabets, as `NICE` if each vowel within it is equidistant from its successor and predecessor vowel, if applicable. E.g., "rhythm", "cool", "malayalam" are `NICE` strings. Write a program to verify whether a given string is `NICE` or not. You are required to take the string as a direct input without asking for its length.
4. Implement the Chatterjee Correlation Coefficient (Chatterjee, JASA 2021) in python. Note that, your inputs are two sets of real values and the output is also a real value.
   Link to the paper: `https://www.tandfonline.com/doi/full/10.1080/01621459.2020.1758115`