COL1000: Introduction to Programming

Nuts & Bolts of Python — Nested Loops ..

Subodh Sharma | Lec 10 | Aug 22

Simple Loops: Best Practices (RECAP)

- **(Correctness) Double check loop's condition (or the continuation condition)**
  - Ensure that iteration variable is changing in a way that will eventually **falsify** the loop **continuation** condition
- **(Error-prone) Avoid making side effects in Loop conditions**

```
while (item := get_next()) is not None:
    process(item)
```

Expr changes the value of item

- **(Interpretability) Avoid making too many exit conditions in the loop body**
  - **effectively use break for early termination**
- **(Optimisation) Avoid recomputing invariant expressions in the loop**

```
nums = [1, 2, 3, 4, 5]
while i in range(len(numbers)):
    length = len(numbers)         # ❌ – Invariant expr computed every time
    print(…)
```

(Nested) Loops & Conditions: Practice Problems

- **(Simple)** Read n integers and print only those that are not divisible by 3.
- **(Simple)** Read n integers and find the first negative number.
- **(Fun)** Keep reading the integers until the user enter -1 (sentinel) or 20 numbers have been processed — whichever comes first. Print the sum of all non-zero positive numbers.
- **(Fun)** Search for a given substring in a user input string without using built-in functions. Print either “Found at position i“ or “Not found”.
- (**Challenge**) Find all primes up to **n** in an optimised way (better than $O(n\sqrt{n})$)
- (**Challenge**) Given a list $[a_0, a_1, \ldots, a_{n-1}]$, and a target **T,** find indices i and j, such that $a_i + a_j = T, i \neq j$

For Loops: Syntax (RECAP)

- **iterable:** An object that can be iterated on
    - Eg: lists, tuples, pairs, strings, dictionaries, **range()**
    - **No explicit update of the target value**

```
for target in iterable:
    # loop body
```

For Loops: Semantics (RECAP)

```
for target in iterable:
    # loop body
```

- The loop calls the function **iter()** on the **iterable** object
- `iter_obj = iter (iterable)`
- Assigns the item to the **target**
- Runs the body
- **Repeatedly call `next(iter_obj)` to get the next item**

Equivalently

```
it = iter(iterable)
while True:
    x = next(it, None)
    if x is None:
        break
    # loop body
```

Loop Invariants

- Predicates (or conditions) that hold true at the start of the loop
- AND, after the execution of the loop body — continue to remain true

- How are they connected to **Post-conditions**?
    - **Loop invariant + Loop termination condition => postcondition after the loop**

Loop Invariants

```
sum = 0
# Loop Invariant: sum $= \Sigma_{j=0}^{i-1} j$
for i in range(1, n+1):
    sum += i
# Post condition: sum $= \Sigma_{j=0}^{i-1} j \wedge i = n + 1$
```