COL1000: Introduction to Programming

Nuts & Bolts of Python — Nested Loops ..

Subodh Sharma | Lec 9 | Aug 20

Reminders!

- **Refer to the other instructors' slides and lecture code for more practice and ideas!**
- Under Lecture Code —> SVS find the `lec8.py` and play with it!
- **Help sessions in CSC lab from 5-6 pm on all working days! (Use only if you need it)**
- Talk to me or send me a personal email re:course feedback!
- **More practice questions will be given out today in the class slides**

Loops: Nested Loops (RECAP)

- **Loop within a loop:**
  - **Semantics:** For each iteration of outer loop, the entire inner loop (i.e. all the inner loop iterations) executes

```
i,j = 0,0
while i < 3:
  while j < 4:
    print(f”{i}:{j}:Inner Loop”}
  print(f”{i}:Outer Loop”)
```

What should be the output?

Perform hand tracing of the execution!

What is the fix?

Loops: Nested Loops (RECAP)

- **Loop within a loop:**
  - **Semantics:** For each iteration of outer loop, the entire inner loop (i.e. all the inner loop iterations) executes

```
i,j = 0,0
while i < 3:
  print(f”{i}:Outer Loop”)
  while j < 4:
    print(f”{i}:{j}:Inner Loop”}
    j += 1
  i += 1
```

What should be the output?

Can you explain?

Loops: Nested Loops

- **Loop within a loop:**
  - **Semantics:** For each iteration of outer loop, the entire inner loop (i.e. all the inner loop iterations) executes

```
i = 0

while i < 3:

    print(f”{i}:Outer Loop”)

    j = 0

    while j < 4:

        print(f”{i}:{j}:Inner Loop”}

        j += 1

    i += 1
```

What should be the output, now?

Can you explain?

Loops: Nested Loops (RECAP)

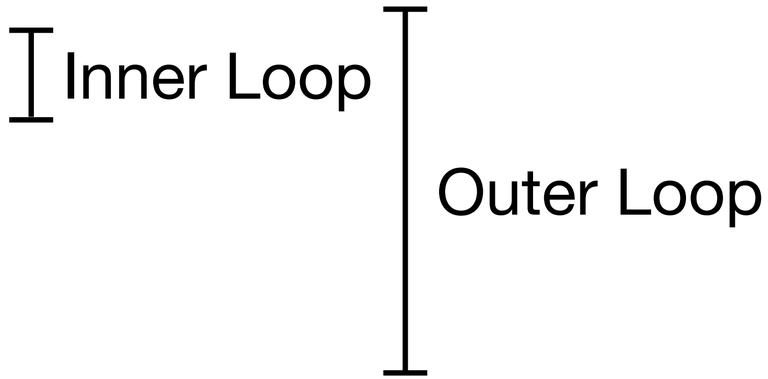
- **Given an input number `n` , test if it is prime**
    1. **Key steps, PRIME(n):**
        1. Set divisor `div = 2`
        2. Check if `div == n`
            1. `True` —> then stop and declared n to be prime
        3. Else check of `n % div == 0`
        4. If `True`, then n is certainly NOT prime
        5. If `False`, then increment div by 1 and repeat from step 2

Loops: Nested Loops (RECAP)

- **Given an input number `n`, print all primes unto `n`**

  1. **Key steps:**

     1. Set i = 2
     2. **If `(PRIME(i))` —>** Print (f“{i} is prime”)
     3. Else —> `i += 1` and repeat from Step (2) until `i == n`



Simple Loops: Best Practices

- **(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

For Loops: Syntax

- **iterable:** An object that can be iterated on
  - Eg: lists, tuples, pairs, strings, dictionaries, **range()**

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

For Loops: Semantics

- 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**

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

Equivalently

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