Computing (ES 112)

Yogesh K. Meena
Shouvick Mondal

August 2024

Computer Science & Engineering
IIT Gandhinagar

Recap: Talking to Python Interpreter (Shell mode)

```
~/ES112Test$ python3
Python 3.10.11 (main, Apr  4 2023, 22:10:32) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for
 more information.
>>> x = 1
>>> print(x)
1
>>> x = x + 1
>>> print(x)
2
>>> exit()
```

This is a good test to make sure that you have Python correctly working. Note that quit() also works to end the interactive session.

Lecture 3

Variables, Expressions, and Statements

What do we say in the Python language?

- Vocabulary / Words - Variables and Reserved words
- Sentence structure - valid syntax patterns
- Story structure - constructing a program for a purpose

```
# Variable assignment
x = 10
name = "Alice"

# Variable reassignment
x = x + 5

# Data types
my_list = [1, 2, 3]
pi = 3.14159
is_valid = True

# Variable scope
def my_function():
    local_var = "I am local"

# Global variables
global_var = "I am global"
```

A short “story” about how to count words in a file

```
name = input('Enter file:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

Indentation (spaces, tabs) and Colon ':' MUST be respected syntactically but consistently

```
python words.py
Enter file: clown.txt
the 7
```

the clown ran after the car and the car ran into the tent and the tent fell down on the clown and the car

Reserved Words

You cannot use reserved words as variable names / identifiers

```
False class return is finally
 None if for lambda continue
 True def from while nonlocal
    and del global not with
     as  elif try or yield
    assert else import pass
     break except in raise
```

Sentences or Lines

```
x = 2
x = x + 2
print(x)
```

Assignment statement

Assignment with expression

Print statement

Variable Operator Constant Function

Paragraphs: Python scripts

Interactive Python is good for experiments and programs of 3-4 lines long.

Most programs are much longer, so we type them into a file and tell Python to run the commands in the file.

In a sense, we are "**giving Python a script**".

As a convention, we add "**.py**" as the suffix on the end of these files to indicate **they contain Python**.

Interactive versus Script

Interactive: You **type directly** to Python **one line at a time,** and it **responds**.

```
~/ES112Test$ python3
Python 3.10.11 (main, Apr  4 2023, 22:10:32) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 1
>>> print(x)
1
>>> x = x + 1
>>> print(x)
2
>>> exit()
```

Interactive versus Script

Script: You enter a **sequence of statements (lines) into a file** using a text editor and tell Python to **execute** the statements in the **file**.

```
name = input('Enter file:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

```
python words.py
Enter file: clown.txt
the 7
```

the clown ran after the car and the car ran into the tent and the tent fell down on the clown and the car

Program Steps or Program/Control Flow

Like a recipe or installation instructions, a program is a **sequence** of steps to be done in order.

Some steps are **conditional** - they may be skipped.

Sometimes a step or group of steps is to be **repeated**.

Sometimes we store a set of steps to be used over and over as needed several places throughout the program.

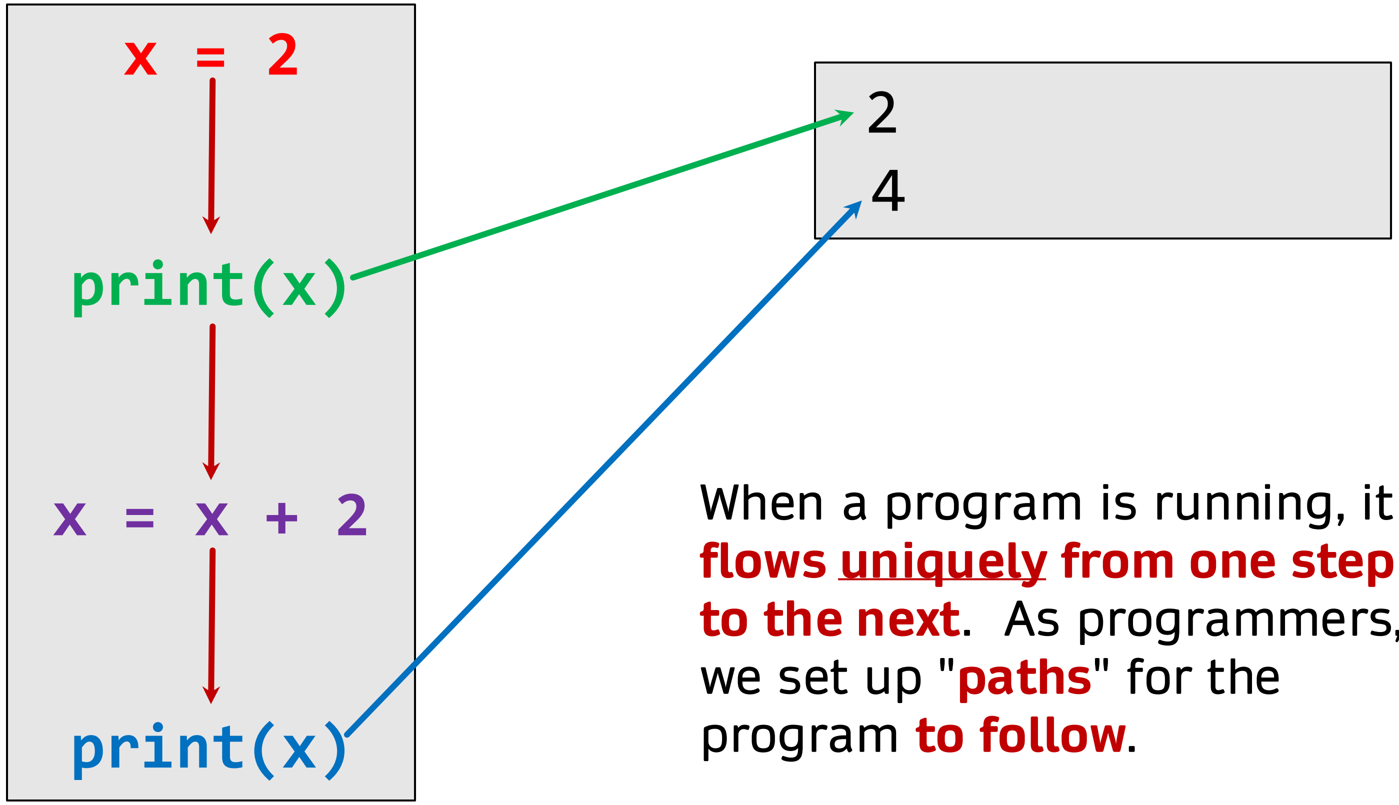
Sequential Steps

Program

```
x = 2
print(x)
x = x + 2
print(x)
```

Control flow

Output



```
x = 2
print(x)
x = x + 2
print(x)
```

```
2
4
```

When a program is running, it **flows <u>uniquely</u> from one step to the next**. As programmers, we set up "**paths**" for the program **to follow**.

Conditional Steps

Program

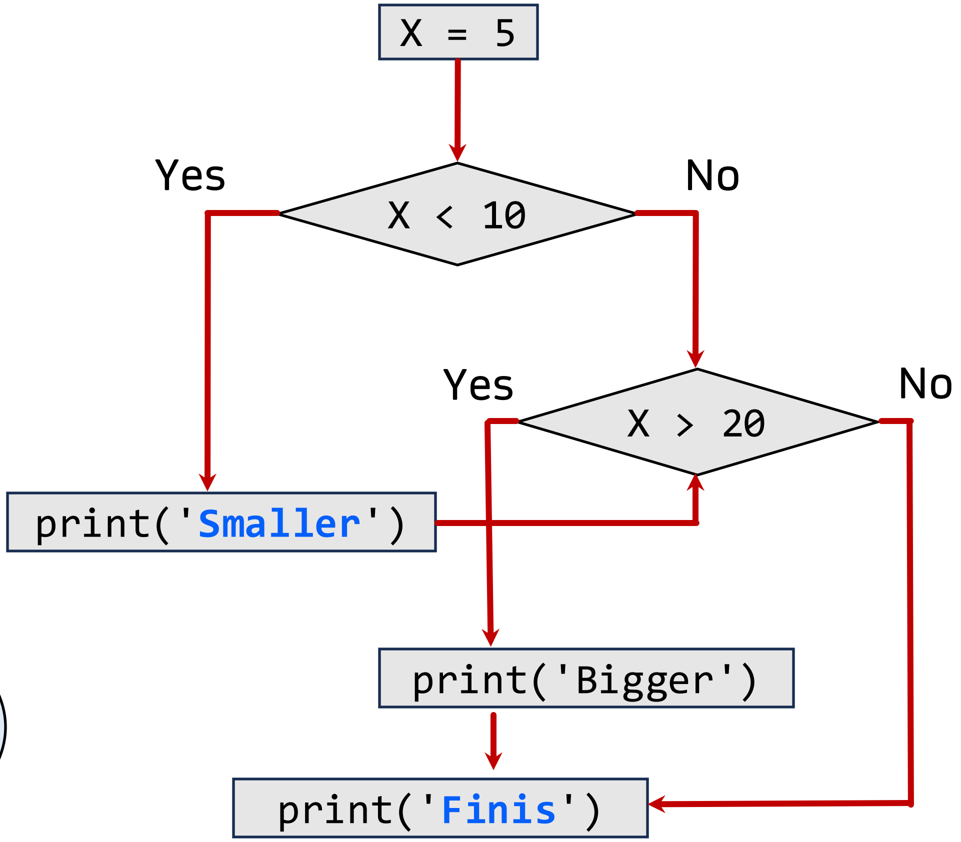
```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Finis')
```

Indentation (spaces, tabs) MUST be consistent.

Colon ':' is a MUST part of the syntax.

Control flow



Output

```
Smaller
Finis
```

When a program is running, **the execution path is not unique**. The non-uniqueness comes from **decision making** on which path to take!

Repeated Steps

$\sum_{i=1}^{10} i$

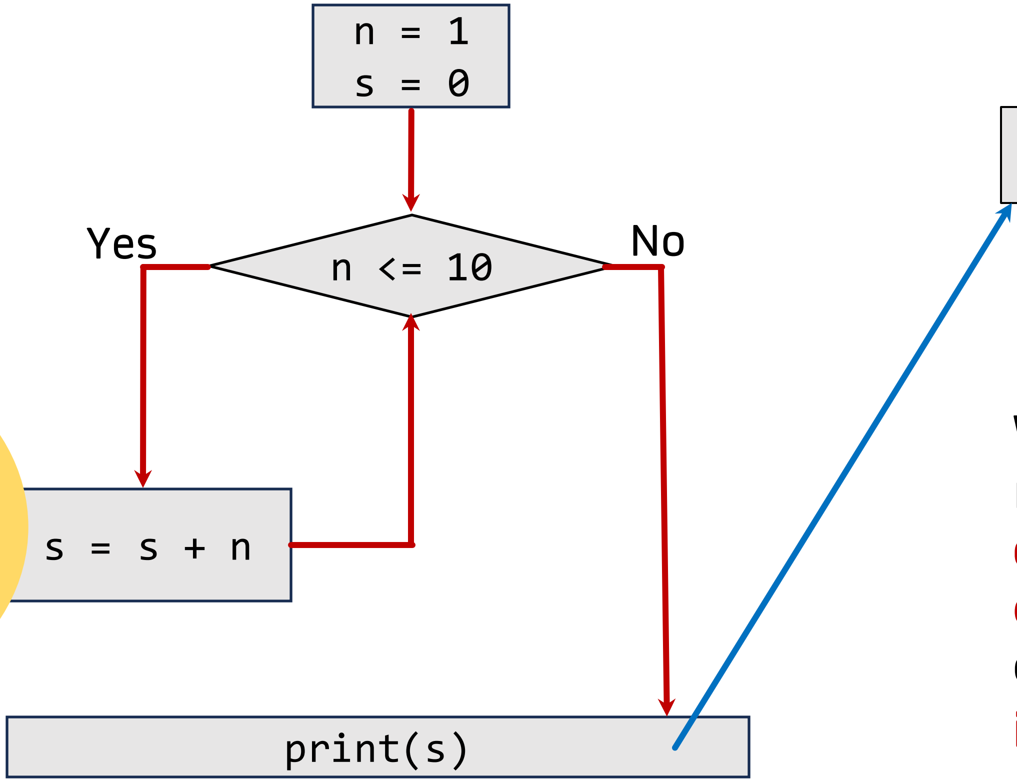
Program

```
n = 1
s = 0
while n <= 10:
    s = s + n
print(s)
```

What will be the output of the program?

Indentation (spaces, tabs) and Colon ':' MUST be respected syntactically

Control flow



Output

?

When a program is running, **the execution of some instruction can be repeated**. This called looping. Here the **iteration variable** is **n**

Repeated Steps

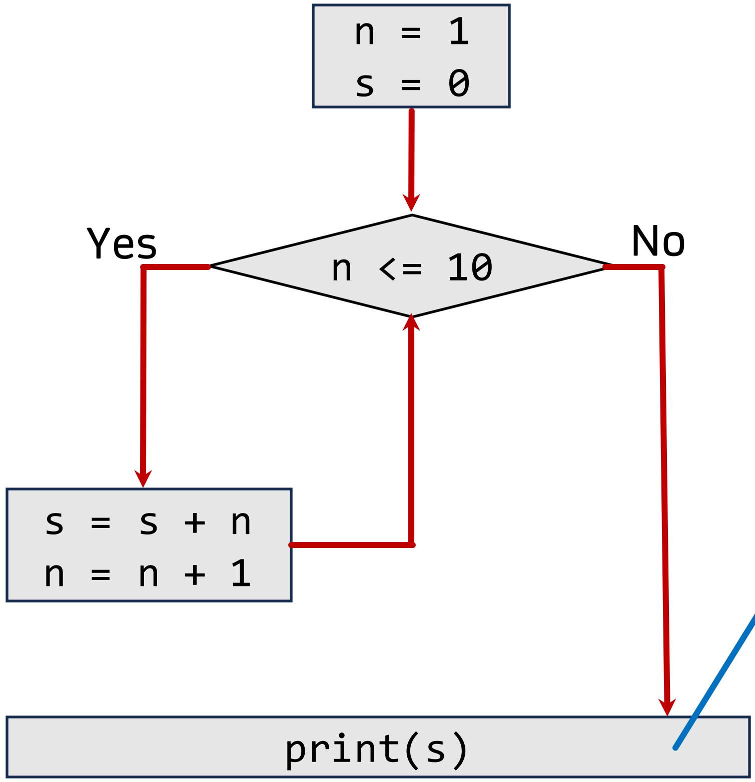
$$\sum_{i=1}^{10} i$$

Program

```
n = 1
s = 0
while n <= 10:
    s = s + n
    n = n + 1
print(s)
```

Indentation (spaces, tabs) and Colon ':' MUST be respected syntactically

Control flow



Output

```
55
```

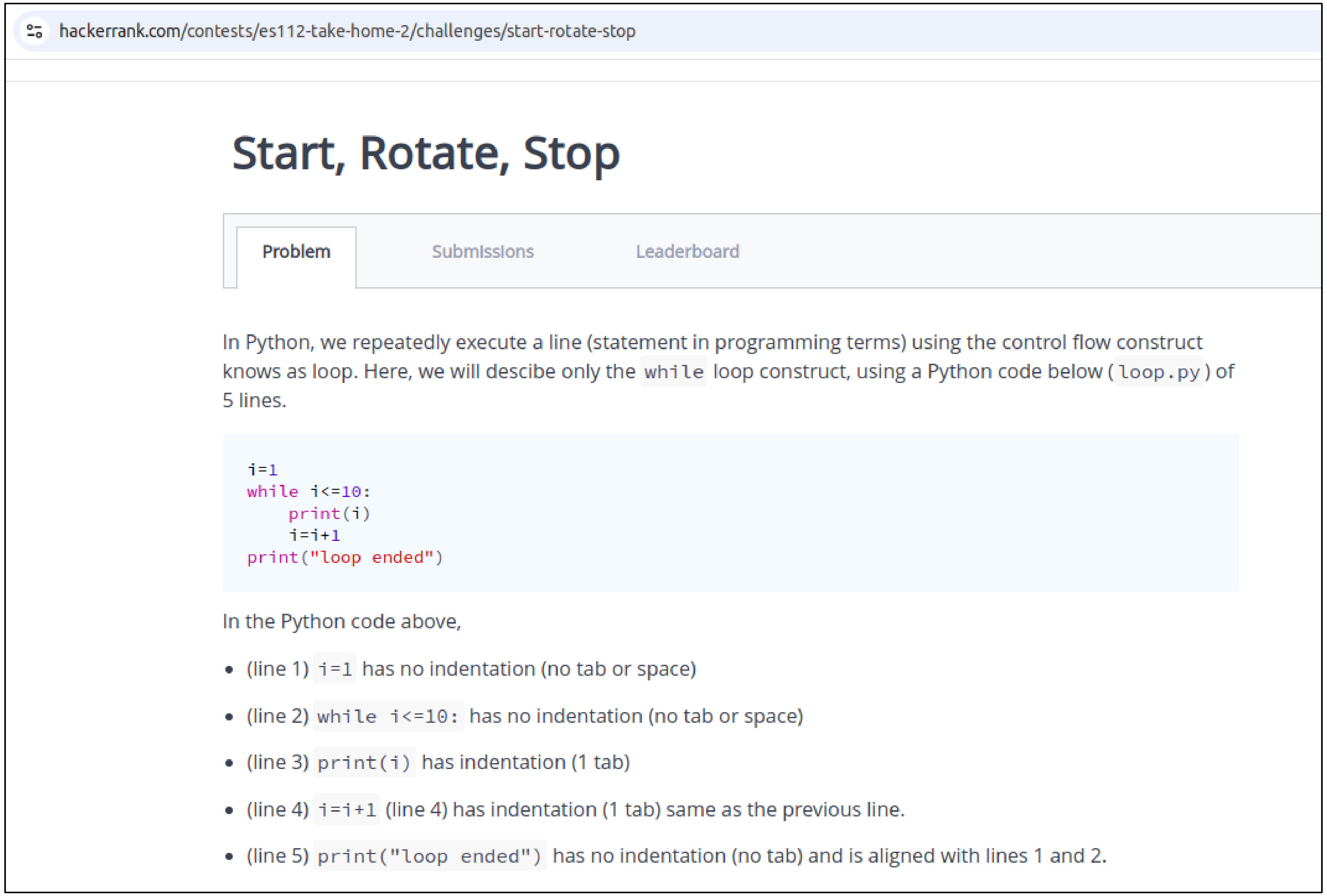
When a program is running, **the execution of some instruction can be repeated**. This called looping. Here the **iteration variable** is **n**

```
i = 0
while i < 6:
  print(i)
  i += 1
```

```
i = 0
for x in
range (0, 6):
  print(x)
```

Take Home Exercise - HackerRank

https://www.hackerrank.com/es112-take-home-2



hackerrank.com/contests/es112-take-home-2/challenges/start-rotate-stop

Start, Rotate, Stop

In Python, we repeatedly execute a line (statement in programming terms) using the control flow construct knows as loop. Here, we will descibe only the `while` loop construct, using a Python code below (`loop.py`) of 5 lines.

```
i=1
while i<=10:
    print(i)
    i=i+1
print("loop ended")
```

In the Python code above,

- (line 1) `i=1` has no indentation (no tab or space)
- (line 2) `while i<=10:` has no indentation (no tab or space)
- (line 3) `print(i)` has indentation (1 tab)
- (line 4) `i=i+1` (line 4) has indentation (1 tab) same as the previous line.
- (line 5) `print("loop ended")` has no indentation (no tab) and is aligned with lines 1 and 2.

Combining the basic constructs

Constructs: Sequential, Conditional, Repeated (iteration)

```
name = input('Enter file:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

Program can be a combination of three constructs. Here conditional and repeated constructs are overlapped.

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Contributors 2024 - Yogesh K. Meena and Shouvick Mondal, IIT Gandhinagar