

Python language: Control Flow

The FOSSEE Group

Department of Aerospace Engineering
IIT Bombay

Mumbai, India

Outline

1 Control flow

- Basic Conditional flow

2 Control flow

- Basic Looping

3 Exercises

Outline

1 Control flow

- Basic Conditional flow

2 Control flow

- Basic Looping

3 Exercises

Control flow constructs

- **if/elif/else** : branching
- **while** : looping
- **for** : iterating
- **break, continue** : modify loop
- **pass** : syntactic filler

Outline

1 Control flow

- Basic Conditional flow

2 Control flow

- Basic Looping

3 Exercises

if...elif...else example

Type the following code in an editor & save as **ladder.py**

```
x = int(input("Enter an integer: "))
if x < 0:
    print('Be positive!')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

- Run in IPython: `%run ladder.py`
- Run on terminal: `python ladder.py`

if...elif...else example

Type the following code in an editor & save as **ladder.py**

```
x = int(input("Enter an integer: "))
if x < 0:
    print('Be positive!')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

- Run in IPython: `%run ladder.py`
- Run on terminal: `python ladder.py`

Ternary operator

- `score_str` is either `'AA'` or a string of one of the numbers in the range 0 to 100.
- We wish to convert the string to a number using `int`
- Convert it to 0, when it is `'AA'`
- `if-else` construct or the ternary operator

```
In []: if score_str != 'AA':  
.....:     score = int(score_str)  
.....: else:  
.....:     score = 0
```


Ternary operator

With the ternary operator you can do this:

```
In []: ss = score_str
```

```
In []: score = int(ss) if ss != 'AA' else 0
```

Outline

1 Control flow

- Basic Conditional flow

2 Control flow

- Basic Looping

3 Exercises

Outline

1 Control flow

- Basic Conditional flow

2 Control flow

- Basic Looping

3 Exercises

while: motivational problem

Example: Fibonacci series

Sum of previous two elements defines the next:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

How do you solve this?

- Task: Give computer, instructions to solve this
- How would you solve it?
- How would you tell someone to solve it?
- Assume you are given the starting values, 0 and 1.

while: Fibonacci

Example: Fibonacci series

Sum of previous two elements defines the next:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

- 1 Start with: **a**, **b** = 0, 1
- 2 Next element: **next** = **a** + **b**
- 3 Shift **a**, **b** to next values
 - **a** = **b**
 - **b** = **next**
- 4 Repeat steps 2, 3 when **b** < 30

while: Fibonacci

Example: Fibonacci series

Sum of previous two elements defines the next:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

- 1 Start with: **a, b = 0, 1**
- 2 Next element: **next = a + b**
- 3 Shift **a, b** to next values
 - **a = b**
 - **b = next**
- 4 Repeat steps 2, 3 when **b < 30**

while: Fibonacci

Example: Fibonacci series

Sum of previous two elements defines the next:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

- 1 Start with: **a, b = 0, 1**
- 2 Next element: **next = a + b**
- 3 Shift **a, b** to next values
 - **a = b**
 - **b = next**
- 4 Repeat steps 2, 3 when **b < 30**

while: Fibonacci

Example: Fibonacci series

Sum of previous two elements defines the next:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

- 1 Start with: **a, b = 0, 1**
- 2 Next element: **next = a + b**
- 3 Shift **a, b** to next values
 - **a = b**
 - **b = next**
- 4 Repeat steps 2, 3 when **b < 30**

while: Fibonacci

Example: Fibonacci series

Sum of previous two elements defines the next:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

- 1 Start with: **a, b = 0, 1**
- 2 Next element: **next = a + b**
- 3 Shift **a, b** to next values
 - **a = b**
 - **b = next**
- 4 Repeat steps 2, 3 when **b < 30**

while

```
In []: a, b = 0, 1
In []: while b < 30:
...:     print(b, end=' ')
...:     next = a + b
...:     a = b
...:     b = next
...:
...:
```

- Do this manually to check logic
- Note: Indentation determines scope

while

We can eliminate the temporary **next**:

```
In []: a, b = 0, 1
```

```
In []: while b < 30:
```

```
    ...:     print(b, end='  ')
```

```
    ...:     a, b = b, a + b
```

```
    ...:
```

```
    ...:
```

```
1 1 2 3 5 8 13 21
```

Simple!

for ... range ()

Example: print squares of first 5 numbers

```
In []: for i in range(5):  
      ....:     print(i, i * i)
```

```
      ....:  
      ....:
```

```
0 0  
1 1  
2 4  
3 9  
4 16
```

range()

`range([start,] stop[, step])`

- **range()** returns a sequence of integers
- The **start** and the **step** arguments are optional
- **stop** is not included in the sequence

Documentation convention

- Anything within **[]** is optional
- Nothing to do with Python

for ... range ()

Example: print squares of odd numbers from 3 to 9

```
In []: for i in range(3, 10, 2):  
.....:     print(i, i * i)  
.....:  
.....:  
3 9  
5 25  
7 49  
9 81
```

Exercise with `for`

Convert the Fibonnaci sequence example to use a **`for`** loop with range.

Solution

```
a, b = 0, 1
for i in range(10):
    print(b, end=' ')
    a, b = b, a + b
```

Note that the **while** loop is a more natural fit here

break, continue, and pass

- Use **break** to break out of loop
- Use **continue** to skip an iteration
- Use **pass** as syntactic filler

break example

Find first number in Fibonnaci sequence < 100 divisible by 4:

```
a, b = 0, 1
while b < 500:
    if b % 4 == 0:
        print(b)
        break
    a, b = b, a + b
```

continue

- Skips execution of rest of the loop on current iteration
- Jumps to the end of this iteration
- Squares of all odd numbers below 10, not multiples of 3

```
In []: for n in range(1, 10, 2):  
.....:     if n%3 == 0:  
.....:         continue  
.....:     print(n*n)
```

pass example

Try this:

```
for i in range(5):  
    if i % 2 == 0:  
        pass  
    else:  
        print(i, 'is Odd')
```

- **pass**: does nothing
- Keep Python syntactically happy

Another example:

```
while True:  
    pass
```

Outline

1 Control flow

- Basic Conditional flow

2 Control flow

- Basic Looping

3 Exercises

Problem 1.1: *Armstrong* numbers

Write a program that displays all three digit numbers that are equal to the sum of the cubes of their digits.

That is, print numbers abc that have the property

$$abc = a^3 + b^3 + c^3$$

For example, $153 = 1^3 + 5^3 + 3^3$

Hints

- Break problem into easier pieces
- How would you solve the problem?
- Can you explain to someone else how to solve it?

Some hints

- What are the possible three digit numbers?
- Can you split 153 into its respective digits,
 $a = 1, b = 5, c = 3$?
- With a, b, c can you test if it is an Armstrong number?

Solution: part 1

```
x = 153
```

```
a = x//100
```

```
b = (x%100)//10
```

```
c = x%10
```

```
(a**3 + b**3 + c**3) == x
```

Solution: part 2

```
x = 100
while x < 1000:
    print(x)
    x += 1    # x = x + 1
```

Solution

```
x = 100
while x < 1000:
    a = x//100
    b = (x%100)//10
    c = x%10
    if (a**3 + b**3 + c**3) == x:
        print(x)
    x += 1
```

Problem 1.2: Collatz sequence

- 1 Start with an arbitrary (positive) integer.
- 2 If the number is even, divide by 2; if the number is odd, multiply by 3 and add 1.
- 3 Repeat the procedure with the new number.
- 4 It appears that for all starting values there is a cycle of 4, 2, 1 at which the procedure loops.

Write a program that accepts the starting value and prints out the Collatz sequence.

What did we learn?

- Conditionals: **if elif else**
- Looping: **while** & **for**
- **range**
- **break, continue, pass**
- Solving simple problems