Chapter 4.
Repetition Structures

Starting out with Python

Code Examples with Jupyter Lab

Kyu Lee. Ph. D.

Computer Science

Condition-Controlled vs Count-controlled

Condition-Controlled vs Count-Controlled loops

- A condition-controlled loop
  - uses a true/false condition to control the number of times that it repeats.
- A count-controlled loop
  - repeats a specific number of times.

Condition-controlled	Count-controlled
while loop	for loop

While loop

While loop

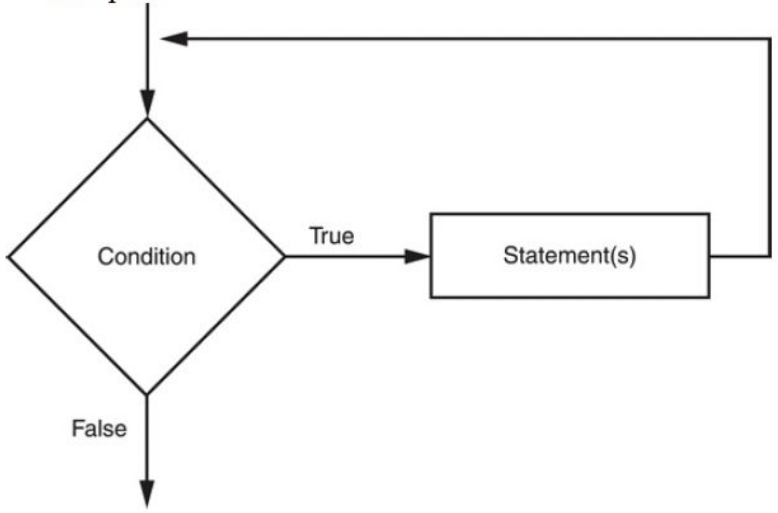
- A condition-controlled loop

```
while condition:
        statement
        statement
        etc.
```

Indentation should be managed
for the inner block

Figure 4-1 The logic of a `while` loop

▶ Description



While loop

- A condition-controlled loop

```
i = 0;
while ( i < 10):
    i = i + 1
    print (i)
# 1 2 3 4 5 6 7 8 9 10
```

```
i = 0;
while ( i < 10):
    print (i)
    i = i + 1
# 0 1 2 3 4 5 6 7 8 9
```

3 things to be careful

1) initial value
2) condition
3) increase/decrease

Caution!
Infinite loop

While loop

- Loop until the character ‘q’ is entered.

```
user_val = input(‘Enter a character’)
while ( user_val != ‘q’ ):
     print ( user_val)
     user_val = input(‘Enter a character’)
```

For loop

For loop

- A count-controlled loop

```
for variable in [value1, value2, … , value n ] :
    statement
    statement


for number in [1, 2, 3, 4, 5 ] :
    print (number)

// 1 2 3 4 5
```

For loop

- A count-controlled loop

```
for strval in [ ‘Python’, ‘Programming’, ‘DVC’ ]:
    print (strval)

for i in [10, 20, 30, 40, 50]
    print (i*2);

for c in range( ord(‘a’), ord(‘e’)+1):
    print (chr(c))
```

Using the range Function with the for Loop

- Range()

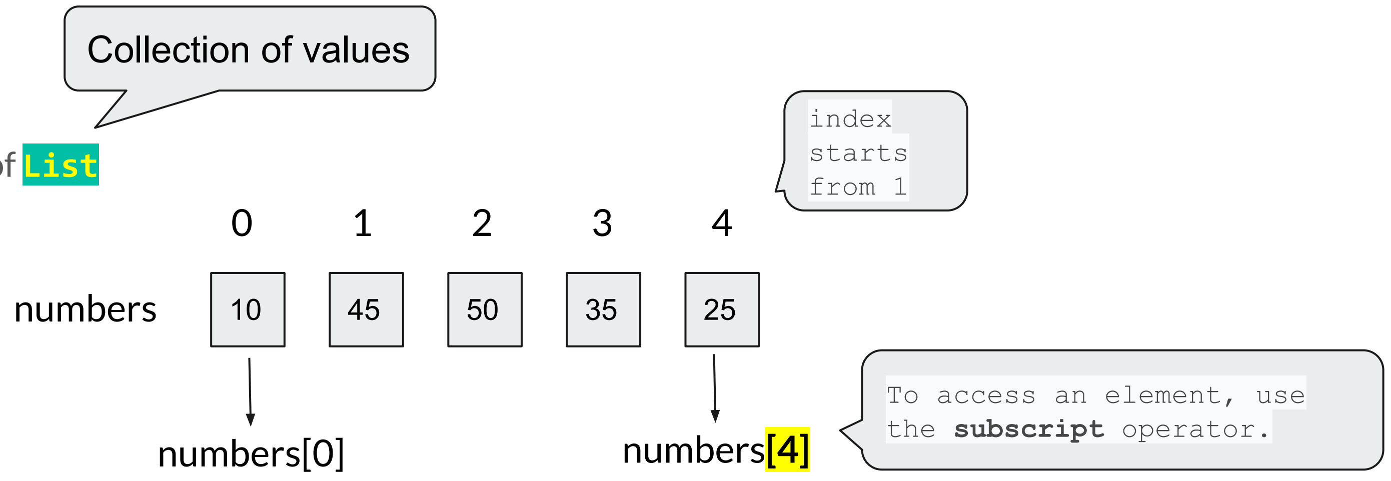
```
for num in range(5):
    print (num) // 0 1 2 3 4

for num in range(5, 10):
    print (num) // 5 6 7 8 9

for num in range(10, 5, -1):
    print (num) // 10 9 8 7 6
```

For Loop

- Basic Usage of List



```
for idx in range(5):
    print (numbers[i])
```

```
for idx in range(len(mylist)):
    print (numbers[i])
```

For loop

- Run all program example code segments and check the results

```
mylist = [45, 56, 77, 88, 100]
for idx in range(5):
    print (mylist[idx])
```

range(5) means 0, 1, 2, 3, 4

```
mylist = [45, 56, 77, 88, 100]
for idx in range(4, -1, -1):
    print (mylist[idx])
```

range(start, end, step)
4 3 2 1 0

```
mylist = [45, 56, 77, 88, 100]
for idx in reversed(range(5)):
    print (mylist[idx])
```

For loop

- List value with for-loop; Run all program example code segments and check the results

```
mylist = [0] * 5
for idx in range(5):
    mylist[idx] = idx
print (mylist)      # print entire elements in the list
```

```
mylist = [ ]
for idx in range(5):
    mylist.append(idx)
print (mylist)      # print entire elements in the list
```

For loop

- **Exercise 1**
    - The program gets all powers of 2 from 0 to N and stores them in a list.
        - N is user input
    - Save all power numbers in the variable `result` as a **list**
        - Expected output if N = 10
        - 1 2 4 8 16 32 64 128 256 512 1024
- Input
    - One integer for power N
- Output
    - 2 to All powers from 0 to N

Use the same variable name `result`

```
[Input]
        10
[Output]
        1 2 4 8 16 32 64 128 256 512 1024

[Input]
        3
[Output]
        1 2 4 8
```

For loop

- **Exercise 2: Calculating a Running Total**
    - Use a for-loop structure
    - A running total is a sum of numbers that accumulates with each iteration of a loop.
    - Make the for-loop with **5** iterations
        - In each iteration, take the user input for integer value
        - Accumulate the user input to the variable "total"
    - After the for-loop,
        - print the variable "total"
    - Run Examples

```
Inputs:
           5
           2
           3
           1
           5
Output:
           16
```

Use the same variable name total

For loop

- **Exercise 3: Calculating a Running Total 2**
  - A running total is a sum of numbers that accumulates with each iteration of a loop.
  - Do the same work as Exercise 2, except the for-loop
    - Use the **while loop**, instead of for-loop

```
Inputs:
      5
      2
      3
      1
      5
Output:
      16
```

Use the same variable name `total`

```python
def main():
    total = 0
    i = 0
    while i < 5:
        num = int(input('Enter your input: '))
        total += num
        i += 1
    print(total)
```

For loop

- **Exercise 3-List: Calculating a Running Total 3**
  - Do the same work as Exercise 2 except the user input
    - In this exercise, we use the **list** “***numbers***”
    - to save 5 user input values
  - Construct the list with 5 user input values
    - `numbers = [0] * 5`
    - `for i in range(len(numbers)):`
      - `numbers[i] = int(input(‘Enter a value ’))`
  -
  - Print the total summation of all elements in the list
    - do not use the library function sum(). Develop your algorithm to get the total of the list

Use the same variable name numbers

```
Inputs:
     5
     2
     3
     1
     5
Output:
    16
```

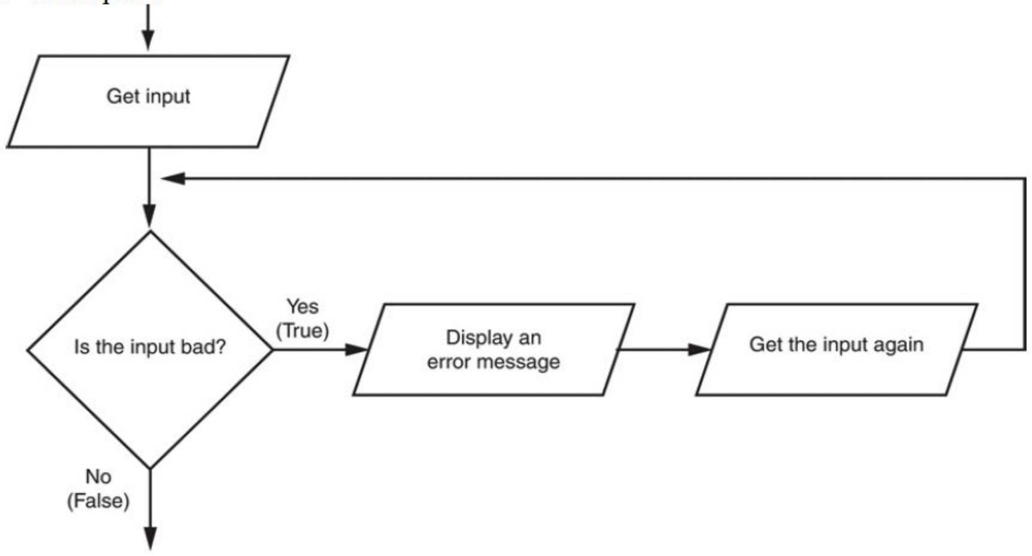
Use the same variable name total

input validation loop

Input Validation

- Input Validation Loop

▶ Description



Input Validation

- Input check
    - The input should be between 0 and 100

```
number = int(input('Enter your input'))

while ( number < 0 or number > 100):
     number = int(input('The input should be between 0 and 100'))

print (number)
```

Input Validation

- Print a character until ‘q’ is entered

```
user_char =  input('Enter your character')

while( user_char != 'q'):
      print (user_char, end=' ')
      user_char =  input('Enter your character')
```

Input Validation

- Print a character until ‘q’ is entered

```
while True:
     user_char =  input('Enter your character')
     if ( user_char == 'q'):
          print ('You entered q.  Program stopped')
          break;
     else:
          print (user_char, end=' ')
```

Input Validation

- try - except

```
try:
      user_num = int(input('Enter a number'))
except ValueError:
      print ('Invalid input: Value Error')

print (user_num)
```

Input Validation

- while loop with try - except

```
while True:
    try:
        user_num = int(input('Enter a number'))
    except ValueError:
        print ('Invalid input: Value Error')
        continue
    else:
        print (user_num)
        break
```

Input Validation

- Exercise 4-1
  - Use the `while loop`
  - Ask the user input (integer value) until it is valid
    - validation condition: greater than 0 and less than 100
    - if the input is invalid, ignore it and
    - ask for another input until it is valid
  - Once you get the valid input,
    - save it to the integer variable `number` and print it

Run Example

```
[Input]
      150
      200
      30
[Output]

      30
```

Use the same variable name `number`

Input Validation

- `Exercise` 4-2
  - Use the **`while loop`** and **`try-except-else(see pages 24, 25)`**
  - Ask the user input until it is a **numeric** value
    - if the input is not a numeric value,
      - take the user input again
    - if the input is a numeric value, save it to the variable `number`.
      - print it and stop

See the slide page 25

Run Example

```
[Input]
      A
      B
      30
[Output]
      Input must be numeric
      Input must be numeric
      30
```

Use the same variable name
`number`

Use the same variable name
`number`

Input Validation

- Exercise 5-1
  - Use the `while loop` for this exercise.
  - Generate 5 random numbers between 0 and 100.
  - Save the random numbers in the list “`numbers`”
  - Get the summation of the list and save it to the “`total`”
  - Print all random numbers in the list “`numbers`” and “`total`”

Run Example

```
[Input]
     None
[Output]
     21 7 61 25 79
     The total sum is 193
```

Use the same variable name
`numbers`
`total`

Input Validation

- Exercise 5-2
  - Write a Python program to generate random numbers until the sum of the numbers is greater than 100.
  - generate random numbers until the sum is greater than 100
    - **Save all random numbers to the list "numbers"**
      - **include the last random number that makes the program stop**
    - **Print** the sum of the random numbers less than 100
  - Run Example

```
[Input]
     None
[Output]
     8 74 15 1 99
     The total sum is 98
```

Use the same variable name `numbers` and `total`

8 + 74 + 15 +1

numbers = [8, 74, 15, 1, 99]
total = 98

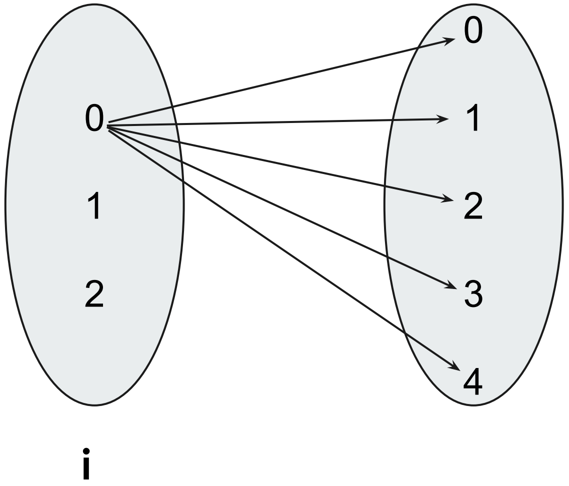
While loop examples

- Exercise 6: How to repeat while loop a certain number of times
  - Make a while loop with 10 iterations and print the iteration number.
- Exercise 7: How to exit while loop on user input
  - Make a while loop that runs until the user input is ‘q’. Print the user input if it is not ‘q’
- Exercise 8: Using while loop with a flag
  - Make a while loop that runs
    - random numbers between 0 and 99
    - until the current random number is greater than the previous random number.
- Exercise 9: How to use while loop with multiple conditions
  - Take the user input(integer value)
    - if the user input is not between 0 and 100, take the user input again.

Nested Loop

Nested Loop

- All combinations of outer and inner for-loop values



```
for i in range(3):
      for j in range(5):
            print (i, j)

            # it will print
                  0     0
                  0     1
                  0     2
                  0     3
                  0     4
                  1     0
                  1     1
                  1     2
                  1     3
                  1     4
                  2     0
                  2     1
                  2     2
                  2     3
                  3     4
```

Nested Loop

- Print pair of numbers with the nested for-loop

```
for i in range(9, 6, -1):
    for j in range(5):
        print (i, j)
```

```
9 0
9 1
9 2
9 3
9 4
8 0
8 1
8 2
8 3
8 4
7 0
7 1
7 2
7 3
7 4
```

Nested Loop

- Nested loop
    - using the index of outer loop in the inner loop

```
for i in range(3):
    for j in range(i, 3):
        print (i, j)
```

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

Nested Loop

- Nested loop
    - using the index of outer loop in the inner loop

      print the pair of number in shaded area with the nested two for-loops

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

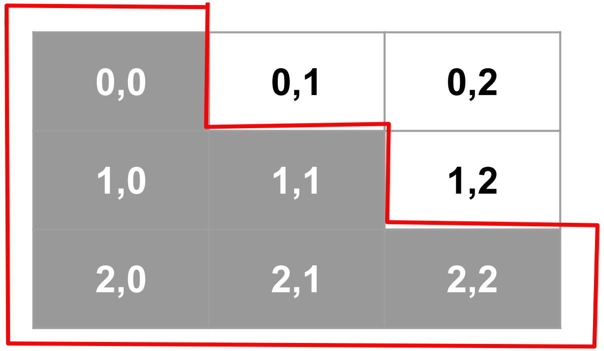
```python
for i in range(3):
    for j in range(i, 3):
        print (i, j)
```

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

Nested Loop

- Exercise 6 : Nested loop
  - Ask the user one integer value N for the dimension
    Print the pair of number in shaded area
    Use the nested for-loops.

```
[Input]
     3
[Output]
        (0, 0)
        (1, 0) (1, 1)
        (2, 0) (2, 1) (2, 2)
[Input]
     5
[Output]
        (0, 0)
        (1, 0) (1, 1)
        (2, 0) (2, 1) (2, 2)
        (3, 0) (3, 1) (3, 2) (3, 3)
        (4, 0) (4, 1) (4, 2) (4, 3) (4, 4)
```



if N = 3, in a 3x3 Matrix,
Print the left-bottom half of the matrix.

```
(0, 0)
(1, 0) (1, 1)
(2, 0) (2, 1) (2, 2)
```

```
for i in range(N):
        for j in """ COMPLETE FOR LOOP STATEMENT """
            print(f'({i}, {j})', end=' ')
            iternum += 1
        print()
```

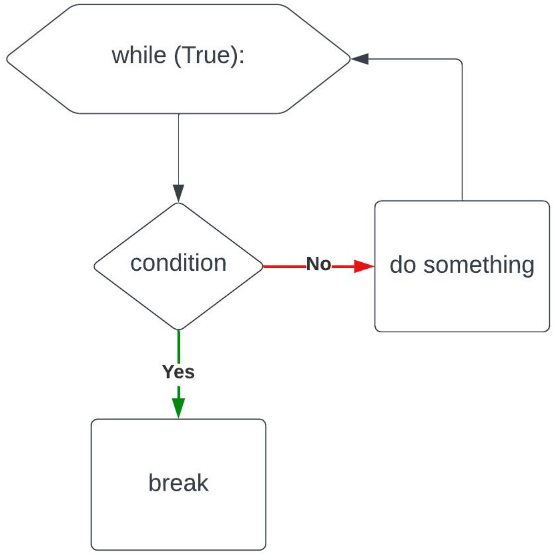
Complete this for loop.
Do not change any other lines

Break and Else

break / continue / else

- break
  - is used to terminate the execution of the loop.

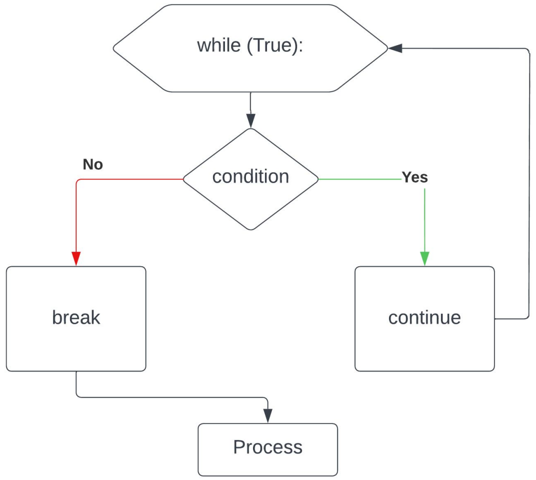
```
while (True):
      if (some condition):
            break
      do something
```



break / continue / else

- **continue**
  - ends the iteration from loops and start to continue the next iteration

```
while (True):
        if (some condition):
                continue
        else
                break
```



break / continue / else

- else
    - is only executed when your while condition becomes false.
    - If you break out of the loop, or if an exception is raised, it won't be executed.

```
while (condition):
     do something
else:
     do something when condition is false
```

not executed when break, except

vs.

```
while (condition):
     do something

do something when condition is false
```

always executed

Exercise 7

- Exercise 7:
  - Ask users for integer value to users **until the current value is greater than the previous value.**
    - All input values except the last one are stored in the list "`numbers`"
      - if current input < the previous input
        - save it to the list `numbers` / continue
      - otherwise
        - `break`
    -
  - input
    - 5 4 3 2 1 5
  - output
    - 5 4 3 2 1

Use the same variable name "`numbers`"

- Print the values in the list `numbers`

```
[Input]
      5
      4
      3
      2
      1
      5
[Output]
      5  4  3  2  1
[Input]
      100
      90
      110
[Output]
      100  90
```

current input > previous value
- break

Exercise 8

- Exercise 8:
  - Write a program that find all prime numbers between two user input values(**Inclusive**).
  - Ask user for two integer values that are greater than 1, and the first value ‘**begin**’ must be less than second value ‘**end**’
  - **Find the prime numbers in the given range.**
  - And save the prime numbers into the list “`plist`”
  - Print all the values in the plist
- input
  - 2
  - 10
- output
  - 2 3 5 7

```
[Input]
      10
      20
[Output]
      11 13 17 19
[Input]
      20
      30
[Output]
      23 29
```

Use the same variable name `plist`

better to use
for-else or while-else

Kyuwoong Lee, Ph. D

Algorithm Development

Introduction to Python Programming

Algorithm Development 1 : Find min value in the list

- There is a list “numbers” that contains 5 integer numbers.
- You will see only one number from number[0] to number[4] at a time.
- When you see the last element in the list, you should determine the least number in the list
- No need to use programming syntax

- Show the all the detail steps to develop your algorithm
    - Pseudo-code can be used to explain the algorithm
    - Draw the flowchart (**draw.io**) to show your algorithm
    - Elaborate on your algorithm

Algorithm Development 2 : Prime number

- Show your algorithm to determine the input value is the prime number or not

- Show the all the detail steps to develop your algorithm
    - Pseudo-code can be used to explain the algorithm
    - Draw the flowchart (**draw.io**) to show your algorithm
    - Elaborate on your algorithm

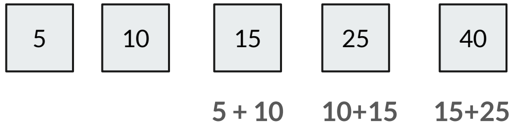
Algorithm Development 3 : Convert to Binary number

- There is an integer value in the variable "number"
- Show your algorithm to convert "number" to the binary number.
  - Do not use any library. Use the loop structure to convert it to binary number.

- Show the all the detail steps to develop your algorithm
  - Pseudo-code can be used to explain the algorithm
  - Draw the flowchart (**draw.io**) to show your algorithm
  - Elaborate on your algorithm

Algorithm Development 4 : series of summation

- Write a algorithm that generates a sequence of numbers where each number is the sum of the previous two numbers.
- Initially, there are two numbers. You will repeat 3 times to make 5 numbers in the list.



- Show the all the detail steps to develop your algorithm
  - Pseudo-code can be used to explain the algorithm
  - Draw the flowchart (**draw.io**) to show your algorithm
  - Elaborate on your algorithm

Assignments

Introduction to Python Programming

Assignment 4-1

- Find the consecutive letters from 'start' to 'end'
  - 'start' and 'end' are the user input (one letter string for each input)
  - e.g., if the user inputs are 'a' and 'f', you should print 'a b c d e f'
  - Save all letters to the list **`'result'`**
- Input
  - a
  - f
- output
  - a b c d e f
- **Requirements**
  - 1) if the 'start' is less than 'end', print error message and **take the user input again**
  - 2) if the 'start' or 'end' is not a alphabet, print error message and **take the user input again**
- Related built-in functions
  - **string.isalpha()**
  - **ord(), chr()**
- Submit:
  - **Elaboration** on your algorithm and troubleshootings
  - Program code, Algorithm **Documentation**, Flow Chart

Use the same variable name `result`

Assumption:
All inputs are lower-case alphabet

```
[Input]
        a
        f
[Output]
        a b c d e f
[Input]
        f
        a
        a
        f
[Output]
        Input Error.
        a b c d e f
```

Assignment 4-2

- Write a program that generates a sequence of numbers where each number is the sum of the previous two numbers. Ask the user for the input N for the number of sequences(N>2).
  - the values in the sequence should be stored to the list “**`result`**”
- Input
  - Two integer values for the starting sequence
  - Ask user the input N for the number of sequences.
- Output
  - all values in this sequence

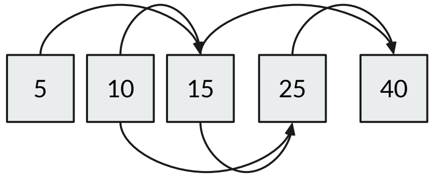
Use the same variable name `result`

- **Run Example**
  - Input
    - 1
    - 2
    - 5
  - Output
    - 1 2 3 5 8

```
[Input]
        5
        8
        3
[Output]
        5  8  13
[Input]
        5
        10
        5
[Output]
        5  10  15  25  40
```

5, 8 is the starting sequence

3 is the number of sequence values



Assignment 4-3

input

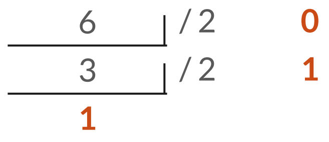
- Write a program that find the remainder of dividing a **number** by 2 repeatedly until the dividend is less than 2.
  - All remainders should be saved to the list "`result`"
  - Do not use any Python Libraries. Develop your code.

Use the same variable name `result`

- Fo

```
As long as x is greater than 0
    Get x % 2 (remainder is either 0 or 1). Append the remainder to the list  result
    Assign x with x divided by 2 (x // 2)
```

- For example,
  - for the input 6
  - the output is
    - 011



Execution Example

```
[Input]
      15
[Output]
      1 1 1 1
```

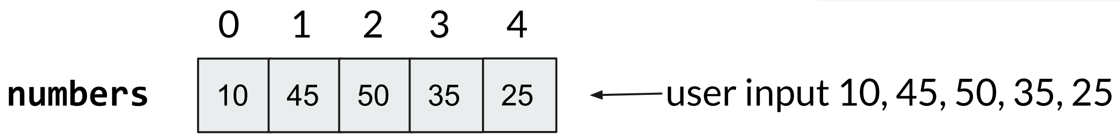
```
[Input]
      7
[Output]
      1 1 1
```

```
[Input]
      64
[Output]
      0 0 0 0 0 0 1
```

Assignment 4-4

- Write a program that finds the least and greatest values among **5** user input values
  - All input values should be saved to the list "**`numbers`**"

Use the same variable name `numbers`



- Find a least and greatest value in the list "**`numbers`**"
  - In this example
    - the least value is 10
    - the greatest value is 50
  - Requirements
    - Do **NOT** use the sorted( ), min( ), or max() functions
  - Output
    - Print all elements in the list on the first line
    - Print the max and min value on the second line

Use the same variable name `minval and maxval`

`Execution Example`

```
[Input]
      10
      45
      50
      35
      25
[Output]
      10 45 50 35 25
      50 10
```