The Center for the Development of the Gifted and Talented
香港科技大學資優教育發展中心

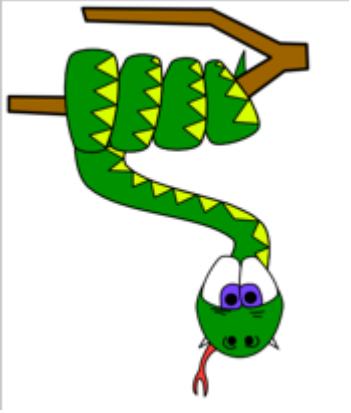
D002 Python for Everyone

Lecture 2 Python Basics II

Dr. Kevin WANG

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

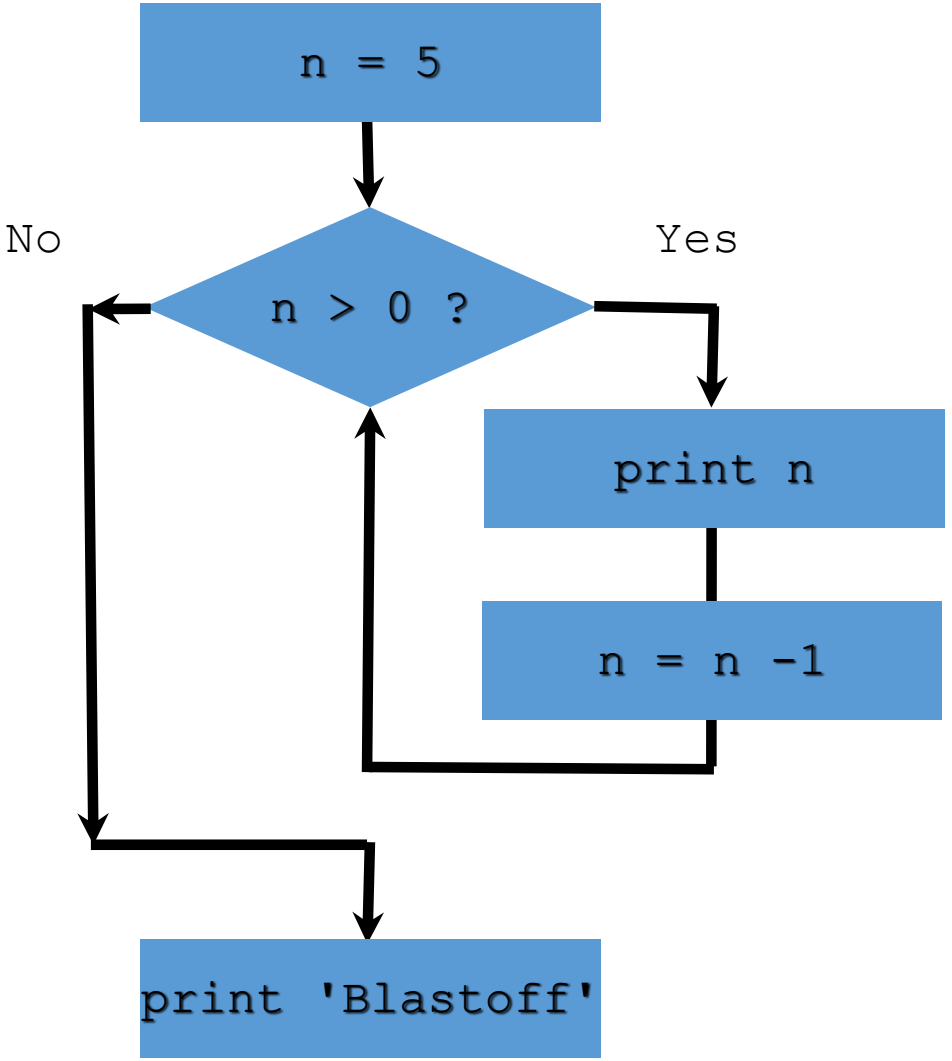


Loops and Iteration

Looping with `while` and `for`

- Another way to alter sequential flow is using *loops*.
- This is helpful when you want to repeat for a few times
  - `for`: when you know exactly how many times to repeat
  - `while`: when you're not sure how many times to repeat, but it will stop when a certain condition is fulfilled
- Loops are a great feature, because we write the loop once, but the computer can repeat the body of the loop many times.
- A little work by the programmer to get a lot of work from the computer!

while loop example: 5 4 3 2 1 … blast off



while_blastoff.py - C:/Users/koala/Desktop/D002Python/Lec2/while_blastoff.py (3.6.5)

File Edit Format Run Options Window Help

```
'''D002 Python for everyone'''
'''Python Program: while example'''
'''Author: Cindy LI'''
'''Date: July 16, 2018'''

n = 5
while n > 0:
    print(n)
    n = n - 1
print("Blast off")
print(n)
```

n here is an iterator that change each time through a loop.
Often the iterator controls how many times a loop run.

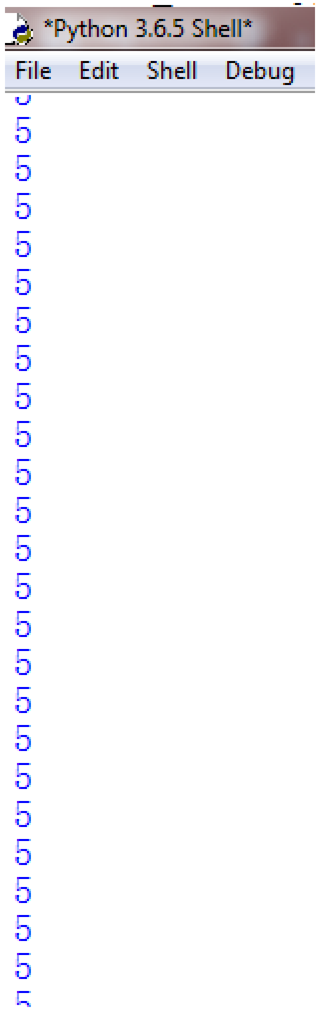
Infinite loop

- This loop will never stop its execution
- We DON'T like it, It is a BUG

while_infinite.py - C:/Users/koala/Desktop/D002Python/Lec2/while_infinite.py (3.6.5)

File Edit Format Run Options Window Help

```python
'''D002 Python for everyone'''
'''Python Program: infinite loop'''
'''Author: Cindy LI'''
'''Date: July 16, 2018'''

n = 5
while n > 0:
    print(n)
print("Blast off")
print(n)
```



`while` loop exercise – Q1

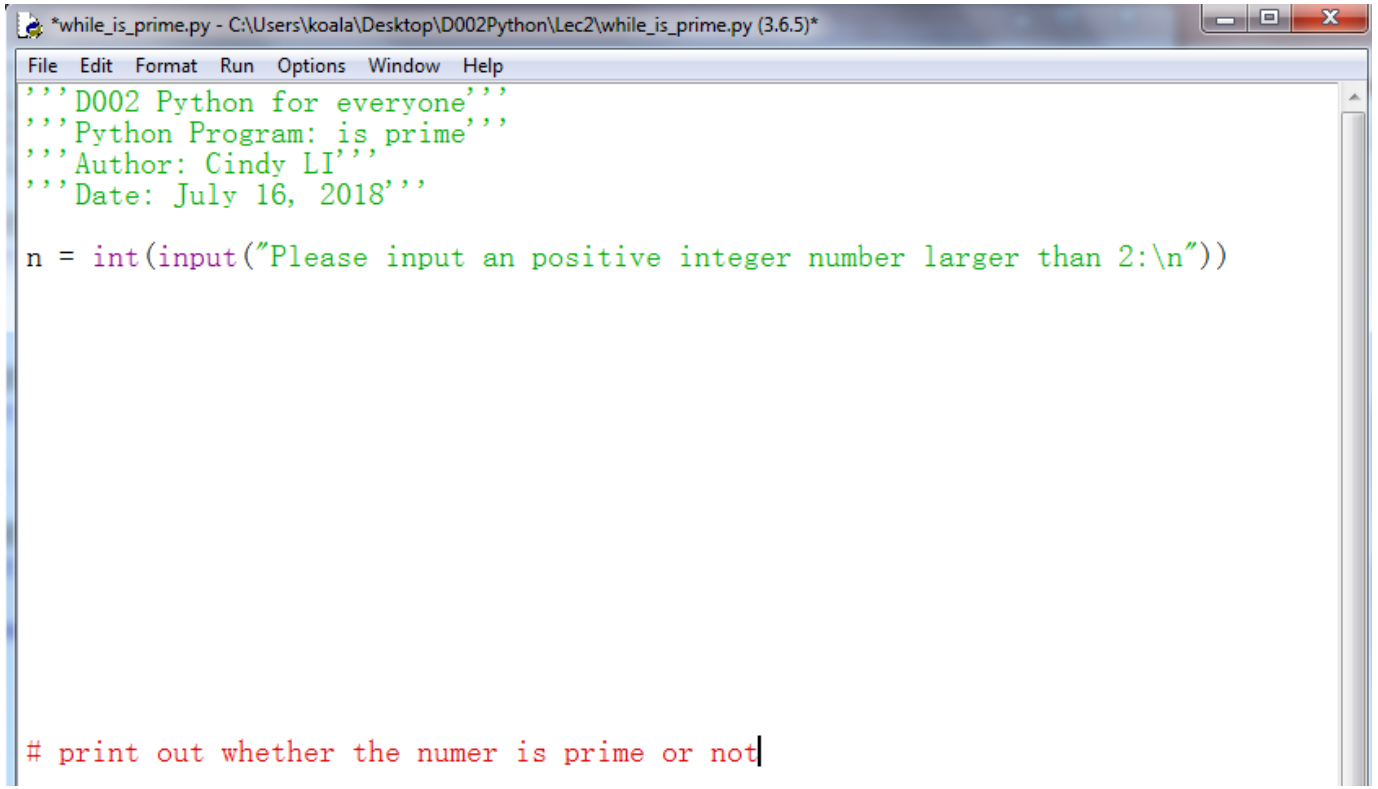
- Ask the user to input an positive integer (e.g. n)
- Find out whether that number is a prime number

Definition of a prime number:

A prime number can only be divided by 1 and itself.

Negative number is not a prime.

0 is not a prime. 1 is a prime.



break, continue, pass

- `break` terminates the innermost executing loop and transfer control after the loop.
- `continue` immediately transfers control to the top of the innermost executing loop.
- `pass` is the no-op statement in Python.

You can't write

```
if age > 18:
        #need "pass" here
else:
        print("no access")
```

`break` example

while_break.py - C:/Users/koala/Desktop/D002Python/Lec2/while_break.py (3.6.5)

File Edit Format Run Options Window Help

```
'''D002 Python for everyone'''
'''Python Program: continue in while loop'''
'''Author: Cindy LI'''
'''Date: July 16, 2018'''

n = 10
while n > 0:
    if ( n > 5):
        print(n)
        n = n - 1
    else:
        break
```

`continue` example

while_continue.py - C:/Users/koala/Desktop/D002Python/Lec2/while_continue.py (3.6.5)

File Edit Format Run Options Window Help

```
'''D002 Python for everyone'''
'''Python Program: continue in while loop'''
'''Author: Cindy LI'''
'''Date: July 16, 2018'''

n = 10
while n > 0:
    if ( n % 2 == 0):
        print(n)
        n = n - 1
    else:
        n = n -1
        continue
```

Nested loop

```
'''D002 Python for everyone'''
'''Python Program: nested while loop'''
'''Author: Cindy LI'''
'''Date: July 16, 2018'''

n = int(input("Please input an positive integer number:\n"))
i = 1
while i <= n:
    j = 1
    while j <= n:
        print("i = %d\tj = %d" % (i, j))
        j = j + 1
    print("\n")
    i = i + 1
```

```
Please input an positive integer number:
3
i = 1   j = 1
i = 1   j = 2
i = 1   j = 3


i = 2   j = 1
i = 2   j = 2
i = 2   j = 3


i = 3   j = 1
i = 3   j = 2
i = 3   j = 3
```

\t means a “tab” symbol

`break` and `while` together – Q2

- what's this program is doing?

```
n = int(input("Please input an positive integer number:\n"))
y = 2
while y < n:
    y = y + 1
    if y % 2 == 0:
        print("%d is even continue" % y)
        continue
    x = int(y/2)
    while x > 1:
        if y % x == 0:
            print("%d has factor %d; break" % (y, x))
            break
            x = x - 1
    if x == 1:
        print("%d is a prime" % y)
```

for loop example

for_blastoff.py - C:/Users/koala/Desktop/D002Python/Lec2/for_blastoff.py (3.6.5)

File Edit Format Run Options Window Help

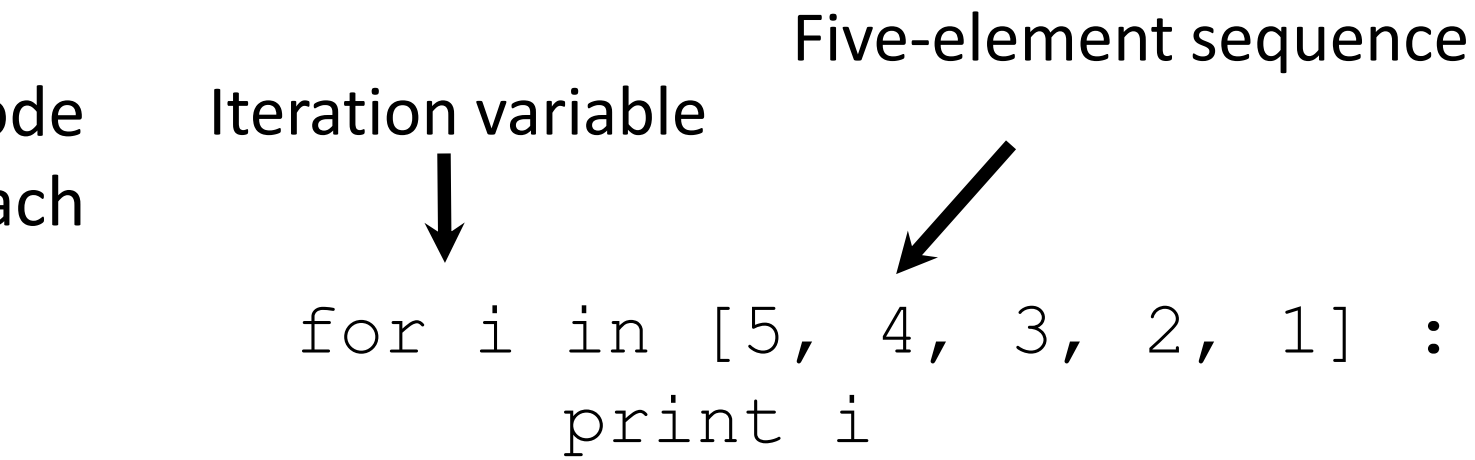
```
'''D002 Python for everyone'''
'''Python Program: for example'''
'''Author: Cindy LI'''
'''Date: July 16, 2018'''

for n in [5, 4, 3, 2, 1]:
    print(n)
print("Blast off")
print(n)

friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends:
    print ("Hello %s" % friend)
print(friend)
```

```
====== RESTART: C:/
5
4
3
2
1
Blast off
1
Hello Joseph
Hello Glenn
Hello Sally
Sally
>>>
```

Iterator in for-loop updates explicitly

Looking at `in`…

- The iterator "iterates" though the sequence (ordered set)
- The block (body) of code is executed once for each value in the sequence
- The iterator moves through all of the values in the sequence



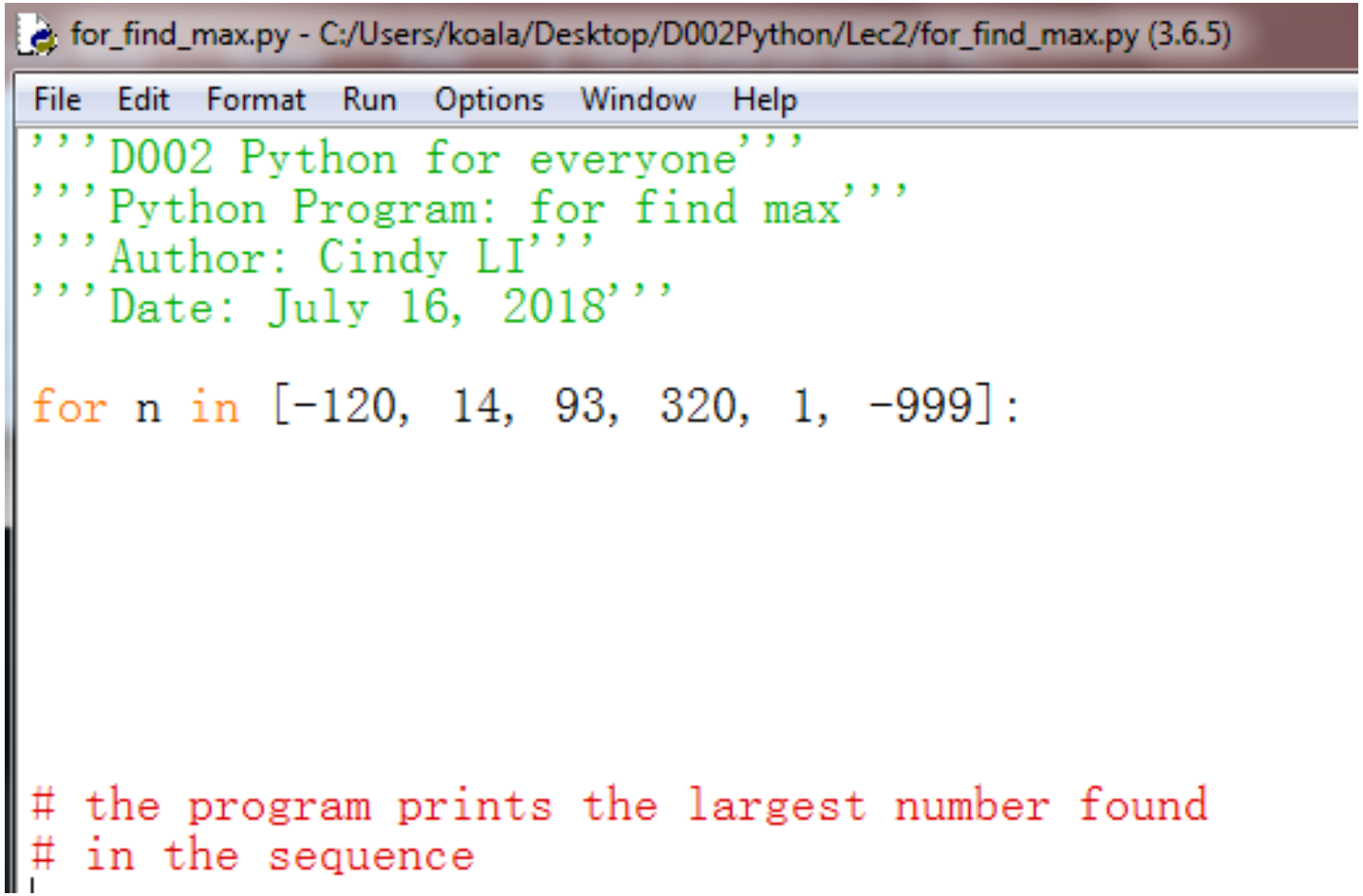
Range with For-loop

- range(start, end) generates the value from start to end - 1.
  - E.g. range(1, 5) gives [1,2,3,4]

```
for i in range(1, 5):
    print("value of i:", i, end=" ")
    '''print(end=" ") will not go to next line'''
    print("\t value of j:", end=" ")
    for j in range(i, 3):
        print(j, end=" ")
    print()
    '''this empty print like press the Enter key -
        move the cursor to next line '''
```

```
====== RESTART: C:/Users/kevinw/Documents/D002Python/Lec2/for_range.py ======
value of i: 1    value of j: 1 2
value of i: 2    value of j: 2
value of i: 3    value of j:
value of i: 4    value of j:
```

loop exercise: find max value in sequence – Q3



Solving a math problem – Q4

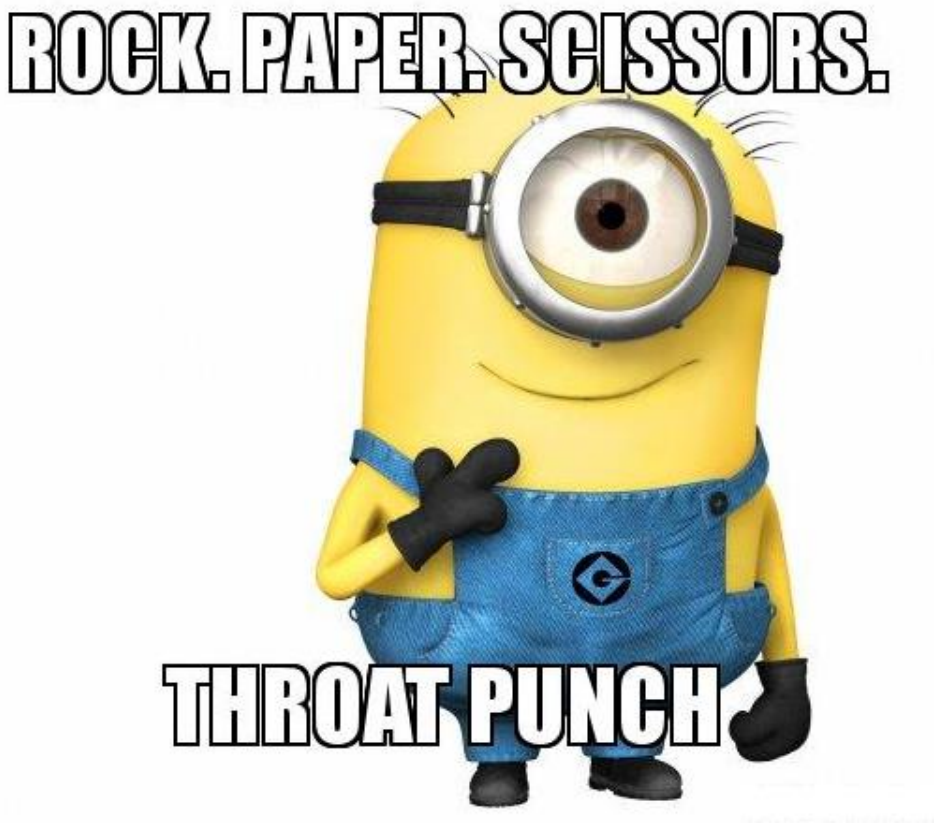
- Given that a,b,c,d are positive integers and a + b + c + d = 60. Find the maximum value of ab + bc + cd.

- Well, too difficult? Why don't we starts with a + b = 10, find max of ab?

- Then we have a + b + c = 20, find max of ab + bc.

Exercise: – Beat the King Q5

- Rock-paper-scissor game among you and minion!
- Similar to L1 Q5 except that you are playing against the “King” and you need to win three times in a row in order to throw him away from his throne.
- Since you are very determined, you will replay the game automatically when you lose.



Exercise: number guessing game v1 – Q6

- The computer thinks a random integer in [1, 100], e.g. 79
- The player tries to guess the number
- E.g. Player guesses 40, print message, your guess is too low
- E.g. Player guesses 92, print message, your guess is too high
- Player guesses 102, print message, wrong input and guess number in [1, 100]
- Allow the player to guess at most 3 times
- When player finally gets the number, print congratulation message, and tell him/her how many times does he/she guessed

Step-by-step number guessing

Generate a random number n in [1, 100]

ask user input x

x == n: user wins, game ends

x in (n, 100]: guess too large

x in [0, n): guess too small

otherwise: guess out-of-range

The user has at most 3 chances

(n, 100] means n to 100,
excluding () n,
Including [] 100

Exercise: number guessing game v2 - Bonus

- Every rule is the same
- But now 2 players are taking turns to guess
- Program needs to print out “Player 1, input your guess”
- Then takes player 1’s guess and gives hint
- If Player 1 gets the number, stop the game and announces the winner
- Otherwise, switch to player 2, print out “Player 2, input your guess”
- Continue above until a winner is there