CS305 Lab2
Introduction to Python & WireShark

Dept. Computer Science and Engineering
Southern University of Science and Technology

SUSTech
Southern University
of Science and Technology

Thanks to HHQ. Zhang

Part. A

Introduction to Python

Python

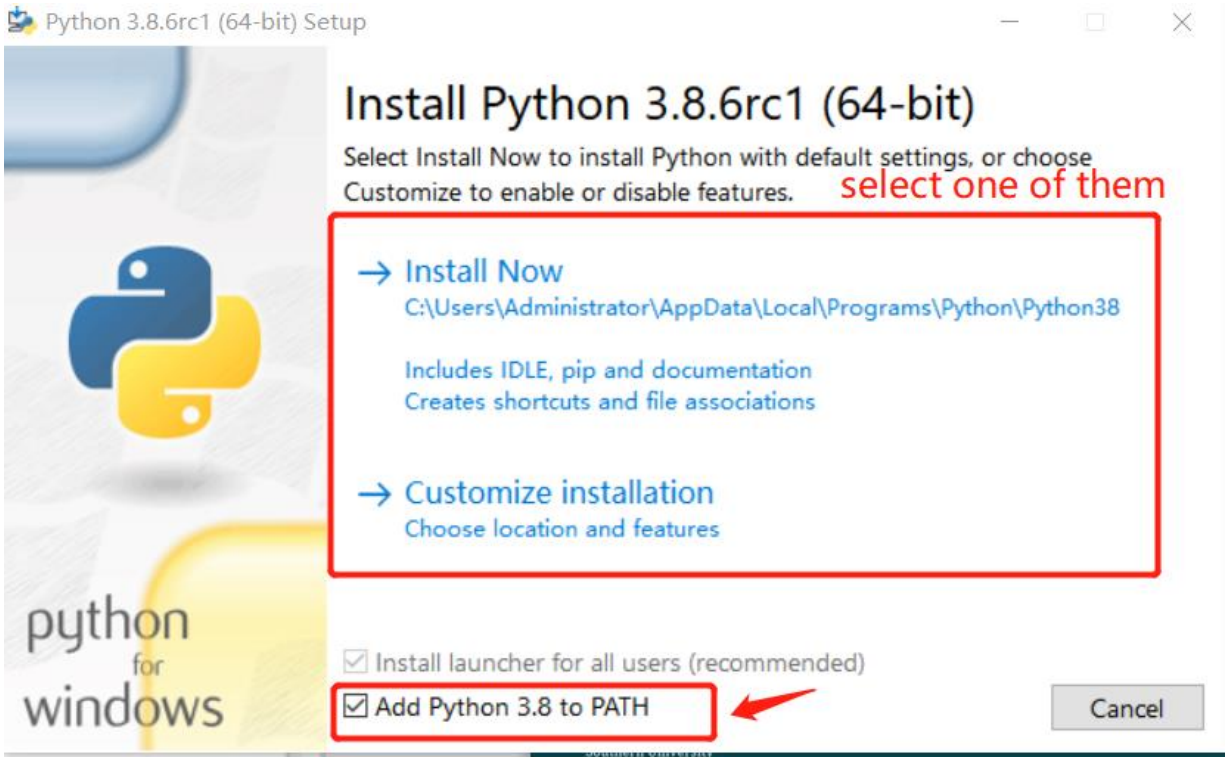
- Python is an interpreted high-level object-oriented programming language.
- First release in 1991.
- Official Tutorial: https://docs.python.org/3/tutorial/

Install python(1)

The Installation package can be got from **https://www.python.org/downloads/**

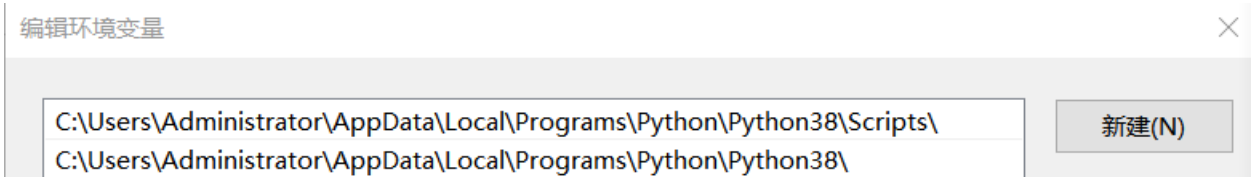
You can choose install it by **default settings** or **customize installation**.

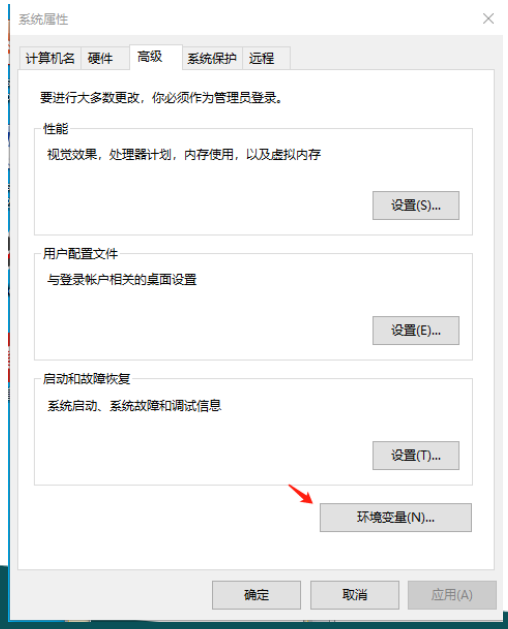
It is highly recommend that choose 'Add Python xx to PATH', or you need to set PATH by hand.



Install python(2)

- If the ‘Add Python xx to PATH’ is not set while installing, configure ‘Path’ manually according to the following steps after the installation.
  - Right click ‘my computer’ on the desktop
  - select ‘attribute’-> ‘advanced attribute’->environment variable
  - configure ‘Path’ with the path where python.exe belonges and it’s subdirectory ‘Scripts’





Read-Eval-Print Loop

- Python has an REPL playground.
- Type and get feedback.

```
C:\Users\Administrator>python
Python 3.8.6rc1 (tags/v3.8.6rc1:08bd63d, Sep  7 2020, 23:10:23) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello World!')
Hello World!
>>>
```

Basic Types and Operations

- The following standard types are built in the interpreter:
  - Numeric Types — int, float, complex
  - Boolean Type — True, False
  - Text Sequence Type — str
  - Sequence Types — list, tuple, range
  - Set Type & Dict Type
  - Binary Sequence Types — bytes, byte array
- There are predefined operations on each type
- Ref: https://docs.python.org/3/library/stdtypes.html

Sequence Types

- List

```
animals = ['dog', 'cat', 'bird']
animals[0] # => 'dog'
animals[0] = 'puppy'
```

- Tuple

```
animals = ('dog', 'cat', 'bird')
animals[0] # => 'dog'
animals[0] = 'puppy'
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Unpacking from Sequence Types

- List

```
foo, bar = ['dog', 'cat']
foo # => 'dog'
bar # => 'cat'
```

- Tuple

```
foo, bar = ('dog', 'cat')
foo # => 'dog'
bar # => 'cat'
```

Set & Dict

- Set

```
animals = set()
animals.add('dog')
animals  # => {'dog'}
```

- Dict

```
alias = dict()
alias['dog'] = 'puppy'
alias[['pig']] = ['hog']
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

Immutable & Mutable

- mutable: it is possible to change its content
- Immutable Type: Numeric, Boolean, str, tuple, bytes, etc.
- Mutable Type: list, dict, set, etc.
- Example:

```
>>> cubes = [1, 8, 27, 65, 125]  # something's wrong here
>>> 4 ** 3  # the cube of 4 is 64, not 65!
64
>>> cubes[3] = 64  # replace the wrong value
>>> cubes
[1, 8, 27, 64, 125]
```

Boolean Values

- Following values are treated as False:
  - None, False
  - 0, 0.0, 0j, Decimal(0), Fraction(0, 1)
  - '', (), [], {}, set(), range(0)
- Otherwise they are True

Flow Control — if

- Example:

```
foo = []
if foo:
    print(foo)
else if foo == []:
    print('100% sure foo is empty')
else:
    print('what hell?')
```

Flow Control — if

- Example:

```
foo = [1, 2, 3, 4]
if foo:
    print(foo)
else if foo == []:
    print('100% sure foo is empty')
else:
    print('what hell?')
```

Flow Control — for

- Example:

```
foo = ['dog', 'cat', 'bird']
for bar in foo:
    print(bar)
for index, value in enumerate(foo):
    print('%d: %s' % (index, value))
    print('{0}: {1}'.format(index, value))
for i in range(10):
    print(i)
```

Flow Control — while

- Example:

```
foo = 10
while foo > 0:
    print(foo)
    foo -= 1
```

Defining Functions

- Example:

```
def fib(n):  # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
```

Defining Functions

- Example:

```
def fib2(n): # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)    # see below
        a, b = b, a+b
    return result
```

Closure

- A closure is an inner function that has access to the outer (enclosing) function's variables.
- Example:

```
def add(x):
def addX(y):
        return y + x
return addX
foo = add(1)
print(foo(2)) # => 3
```

Defining Classes

```
class Animal:
    def __init__(self, name):
        self.name = name

class Duck(Animal):
    def __init__(self, name):
        super(Duck, self).__init__(name)
@staticmethod
    def quack():
        print('Quack')
```

Duck Type

- "If it walks like a duck and it quacks like a duck, then it must be a duck"

```
class Dog(Animal):
    def __init__(self, name):
        super(Duck, self).__init__(name)
    @staticmethod
    def quack():
        print('Quack')
```

Duck Type

- "If it walks like a duck and it quacks like a duck, then it must be a duck"

```
func testDuck(duck):
        duck.quack()


duck = Duck('Tommy')
dog = Dog('Fox')
testDuck(duck)
testDuck(dog)
```

Module

- Save our fib functions into fib.py

```
import fib
fib.fib(5) # => 0 1 1 2 3
result = fib.fib2(5) # => [0, 1, 1, 2, 3]
```

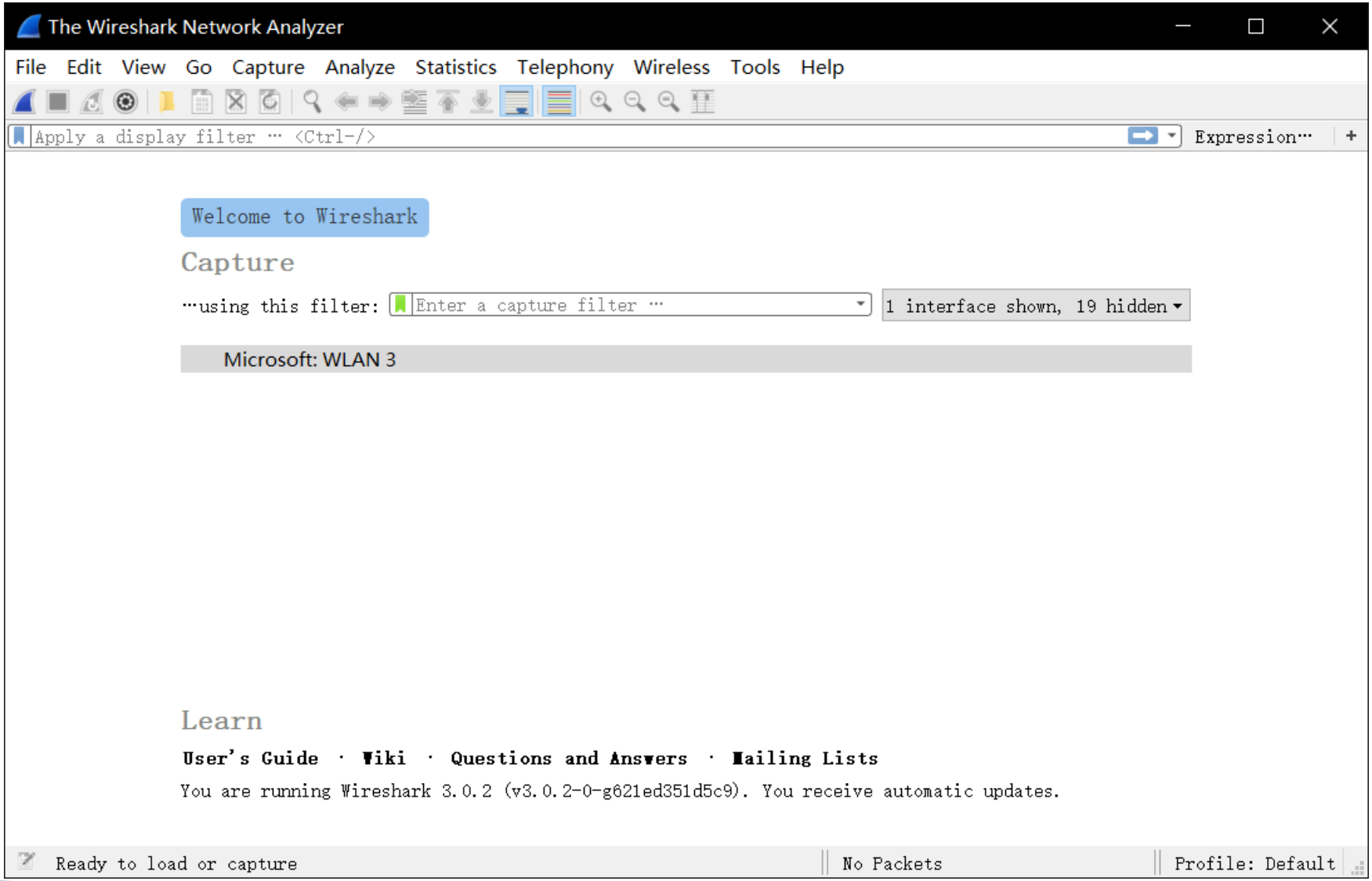
Part. B

Packet Capture and Analysis

Wireshark

- Wireshark is a free and open-source packet analyzer.
  It is used for network trouble shooting, analysis, software and communications protocol development, and education.
- Alternative utilities:
  - tcpdump
  - Tshark

*Tip: new version of Wireshark uses Npcap instead of Winpcap.


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ··· <Ctrl-/>
Expression···
Welcome to Wireshark
Capture
···using this filter: Enter a capture filter ···
1 interface shown, 19 hidden
Microsoft: WLAN 3
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
You are running Wireshark 3.0.2 (v3.0.2-0-g621ed351d5c9). You receive automatic updates.
Ready to load or capture
No Packets
Profile: Default

Capture Filter

- Capture filter allows you to select the packets you want from all the packets captured by Wireshark.
- A proper capture filter can reduce the workload of Wireshark and the size of raw packets.

Capture Filter

- Example:
  - host 172.18.5.4
  - port 53
  - tcp port 80
- For more syntax explanation: http://www.tcpdump.org/manpages/pcap-filter.7.html

Display Filter

- After the capture starts, the display filter can be set to accurately hide the packet you don’t care.
- Display filter can be change at anytime on the fly.

Assignment 2.1
Find Narcissistic Number

- filename: narcissistic_number.py
- requirement:

  implement a function to find all the narcissistic numbers in a range

  function signature:

  ```
  def find_narcissistic_number(start: int, end: int) -> list
  ```

Narcissistic Number

An n-Digit number which is the Sum of the nth Powers of its Digits is called an n-narcissistic number, or sometimes an Armstrong Number or Perfect Digital Invariant (Madachy 1979). The smallest example other than the trivial 1-Digit numbers is

$$153 = 1^3 + 5^3 + 3^3.$$

Assignment 2.2

Use Wireshark to capture packets and answer the questions with your screenshots:

1. launch a http session between your host and “www.example.com”

   1-1. what 's the filter used for the HTTP session between your host and “www.example.com”?

   1-2. Find a HTTP response packet in this http session.

   what's the decimal and hexdecimal representation of the src ip addr, src port, dst ip addr and dst port of this packet.

2. launch a http session between your host and “www.baidu.com”

   2-1. answer the question 1-2 based on the new http session between your host and “www.baidu.com”

   2-2. list the items which value is same in the answers of both question 1-2 and 2-1.

Assignment 2.3

Using ICMPv4 to trace route between your computer(source) and “www.163.com”(destionation). Using a proper capture filter/display filter to capture/display this session.

Answer the following questions with words and screenshots on both the execution result of command and capture result of wireshark:

1. How many 'time-to-live exceed' and 'echo reply' response messages are received ? What’s the source IP address of the 1st received 'time-to-live exceed' message, What’s the source IP address of the 1st received ‘echo reply' message?

2. Calculate the RTT (round-trip time) between your host and www.163.com based on the packets captured. Are they same with RTT from command execution result?

3. Add the value of hops(between source and destination) and TTL value of ICMPv4 messages received by source(which send ICMPv4 echo request). Is it the initial value of TTL from ICMPv4 message send by source or the ICMPv4 message send by destination? how to prove this conclusion.