Open Innovation Lab
CUHK

CTF Training Camp for Hackers: Basic Training

CUHK Open Innovation Lab

Cousin
Coordinator of CUHK Open Innovation Lab

CUHKOIL Discord Server

We use this server for

- Announcements
- Resources for Workshops and CTFs
- Discussion
- Chat! :)

Note: This link is only valid for 50 invites.
(For CUHK Students Only)

whoami

- WU Ka Lok (Cousin)
- Coordinator of Open Innovation Lab
- First Year MPhil Student in Information Engineering, CUHK
- Did Undergrad @ HKUST, majored in Mathematics (PMA) and Computer Science
- Former Captain (打雜) of Firebird CTF Team, an Academic Team @ HKUST (2020-2021)
- Interested in Maths (Algebra stuff), cryptography (esp. the math) and cybersecurity in general

Who We Are: CUHK Open Innovation Lab

- Hub for advancing the movement of open source, open data, open culture and technology entrepreneurship among engineering students.
- participate in events like Hackathon, Bootcamps and Capture-The-Flag (CTF) competitions

CTF Workshop Schedule (Tentative)

	Wednesday	Thursday
Oct	6	7: Basic Training
	13: Web Security 1	14: Chung Yeung Festival
	20: Binary Exploitation 1	21: Web Security 2
	27: Binary Exploitation 2	28: Cryptography 1
Nov	3: Binary Exploitation 3	4: Cryptography 2
	10: Binary Exploitation 4	11
		18: HKCERT writeup sharing

- Again all 7:30 pm at SHB801, CUHK
- No need to attend all trainings! You can just attend trainings that you are interested in.
- Think back about the team thing: find teammates that accel at different categories.

HKCERT CTF 2021

- Competition from 12 Nov (Fri) 5pm - 14 Nov (Sun) 5pm
- Online Jeopardy, team of 1-4
- Teritary Division: team up with any full-time students on any tertiary institution (diploma, higher diploma, associate degree and bachelor degree)
- (Online) Workshop on 8 Oct (Tomorrow!) and 22 Oct 2:30pm - 5pm (Recommended)
- https://ctf.hkcert.org/
- After the competition, we will hold a write-up sharing session (17 or 18 Nov)

Basic Training

Agenda

- Very very basic… >_<

- Basic Linux Commands
- Pipeline
- Special commands and files
- Local File Inclusion, Command Injection

- Common Encoding

- Python basics
- pwntools

Operating System

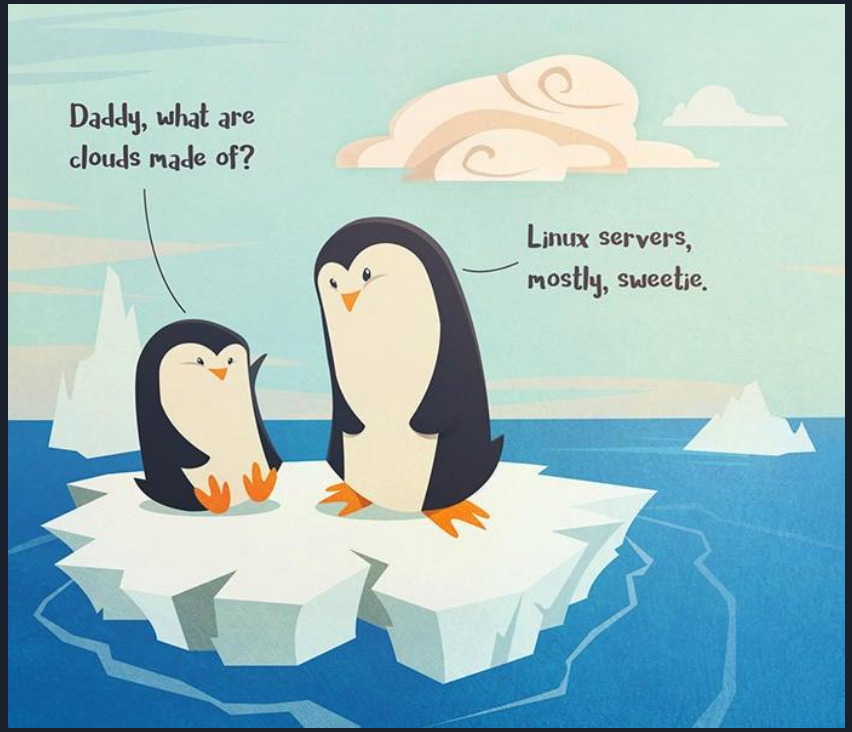
- Everything that are shipped to you when you install one
- Includes the kernel, some systems program and other applications, software bundles
- On a low level, OS allocates resources (CPU time, memory, hard disk access, etc.) to programs
- e.g. Windows, Linux, Unix, Mac (which is based on UNIX), FreeBSD, Solaris, TempleOS (???)
- We will run Linux on a virtual machine. (Although we don't need it today)

Virtual Machine

- Virtual machine emulates a computer system.
- Virtual machines allows operating systems to run as applications within other OSes.
- Use multiple OS (mainly your main OS + Linux)
- Sandbox: will not trash your machine if anything goes wrong on your VM
- WSL is a compatibility layer for running Linux executables, but does not provide the sandbox feature.

Linux/Unix

- Mac and Android are Linux/Unix variants
- Most servers in the world are Linux Servers
- Most server we deal with are Linux Servers
- (Except when we need to deal with Windows machines sometimes)



Ackchyually...

I'd just li is in
fact, GN Linux
is not ar nt of a
fully fun tilities
and vita X.

Many co day,
without NU
which is s are
not awa Project.
There re t of the
system

Linux is he's
resources to the other programs that you run. The kernel is an essential part
of an operating system, but useless by itself; it can only function in the context

Linux Basics

Linux Distributions

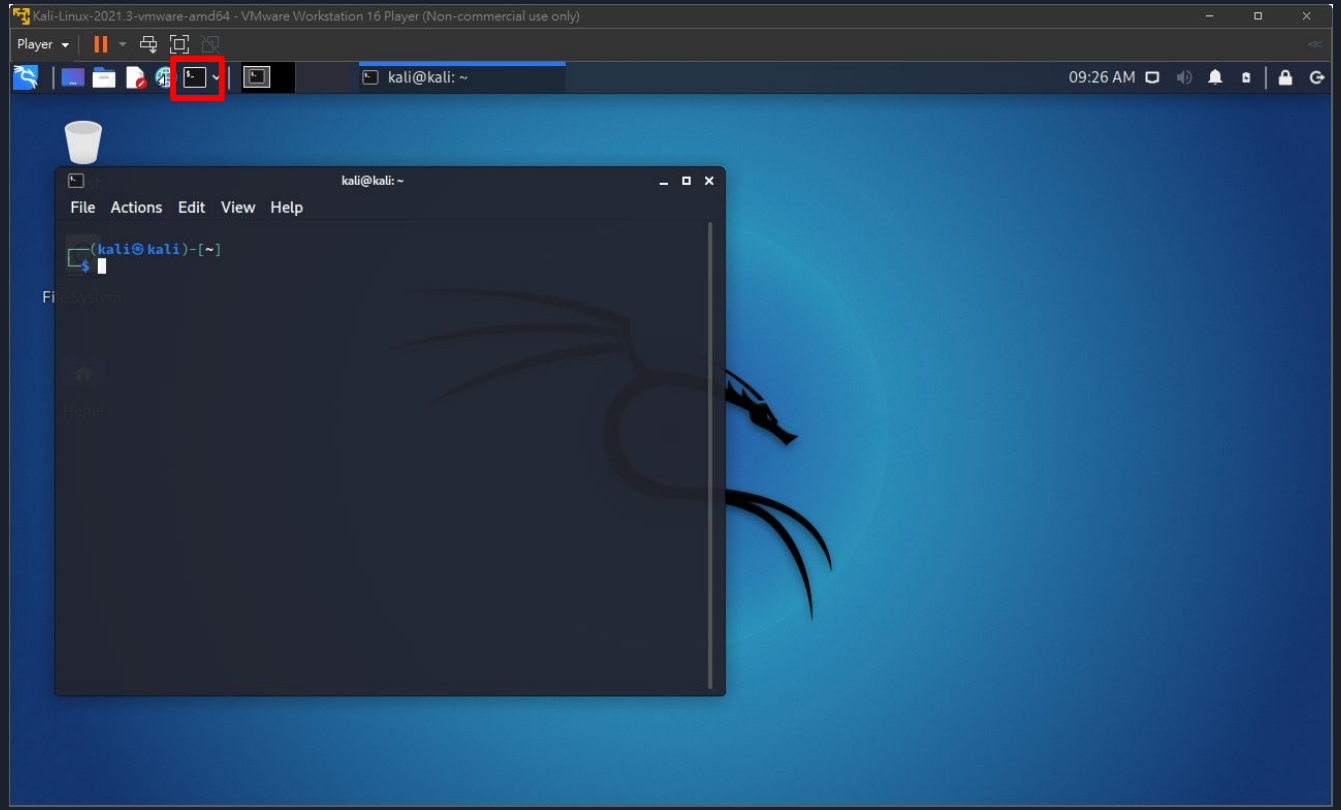
- “distro”
- The “flavor” of Linux you want, different software bundles, GUI, philosophy ...
- Ubuntu (based on “debian”) is the most popular, well-supported and newbie-friendly
- Fedora, Arch Linux, Gentoo (if you like torturing yourself), Linux From Scratch (you don’t need to attend this workshop)…
- We use Kali Linux.

Why Kali?

- Kali for hacking / pentesting
- debian-based so well-supported
- preinstalled with lots of useful tools for cybersecurity
- Don’t do illegal stuff with kali: Ethical hacking :)

username: `kali`
password: `kali`

Using the Terminal



CLI Basics

current working directory
(~ is home directory)

CLI Basics

Let make it print “hello world”

```
echo  "hello world"
```

command

parameters / arguments

Easy!

```
File  Actions  Edit  View  Help
kali@kali:~$ echo "Hello world"
Hello world
kali@kali:~$
```

The Mother of All Commands

- ~~`systemd`~~ `man`
- As we always say: Read The [redacted] Manual
- Everything you want to know about the command (or function)

- Sometimes you can also try `<command> -h` or `<command> --help`

LINUX TERMINAL FOR BEGINNERS

Basic Commands

`cd <dir>`	**C**hange **D**irectory (aka folder)
`ls <dir>`	**LiS**t directory contents
`mv <src> <dest>`	**MoV**e file from src to dest
`cp <src> <dest>`	**CoP**y file from src to dest
`rm <file>`	**ReMove** file
`mkdir <dir>`	**MaKe** a **DIR**ectory
🐱 `<file>`	Print the content of file (cat stands for con🐱enate)
`echo <string>`	Print a string to standard output
`pwd`	**P**rint the absolute path of **W**orking (current) **D**irectory
`touch <file>`	Create blank file or change access and modification time to current time

Basic Commands

`cd <dir>`	**C**hange **D**irectory (aka folder)
`ls <dir>`	**LiS**t directory contents
`mv <src> <dest>`	**MoV**e file from src to dest
`cp <src> <dest>`	**CoP**y file from src to dest
`rm <file>`	**ReMo**ve file
`mkdir <dir>`	**MaK**e a **DIR**ectory
`cat <file>`	Print the content of file (cat stands for con**CAT**enate)
`echo <string>`	Print a string to standard output
`pwd`	**P**rint the absolute path of **W**orking (current) **D**irectory
`touch <file>`	Create blank file or change access and modification time to current time

`ls`

- `ls -l` prints a list format
- `ls -a` lists ALL files
- we usually do `ls -la`

```
total 32388
drwxr-xr-x 20 cousin cousin  4096 Oct  4 21:37 .
drwxr-xr-x  7                     5:19 ..
-rw-------  1                     1:37 .bash_history
-rw-r--r--  1                     5:18 .bash_logout
-rw-r--r--  1                     5:18 .bashrc
drwx------ 15                     6:05 .cache
drwx------ 16                     6:05 .config
drwxr-xr-x  2                     6:55 Desktop
drwxr-xr-x  2                     6:55 Documents
drwxr-xr-x  2                     6:55 Downloads
drwx------  3                     6:56 .gnupg
-rw-------  1                     7:41 .lesshst
drwxr-xr-x  5                     8:31 .local
drwxr-xr-x  2                     6:55 Music
drwxr-xr-x  2                     6:55 Pictures
-rw-r--r--  1                     5:18 .profile
drwxr-xr-x  2 cousin cousin  4096 Aug 27 16:55 Public
-rw-------  1 cousin cousin  9924 Sep 29 13:55 .python_history
```

File permission

File group

Modification time

```
-rw-r--r--  1 cousin cousin  9792634 Sep  7 11:37 openssl-1.1.1e.tar.gz
```

File owner

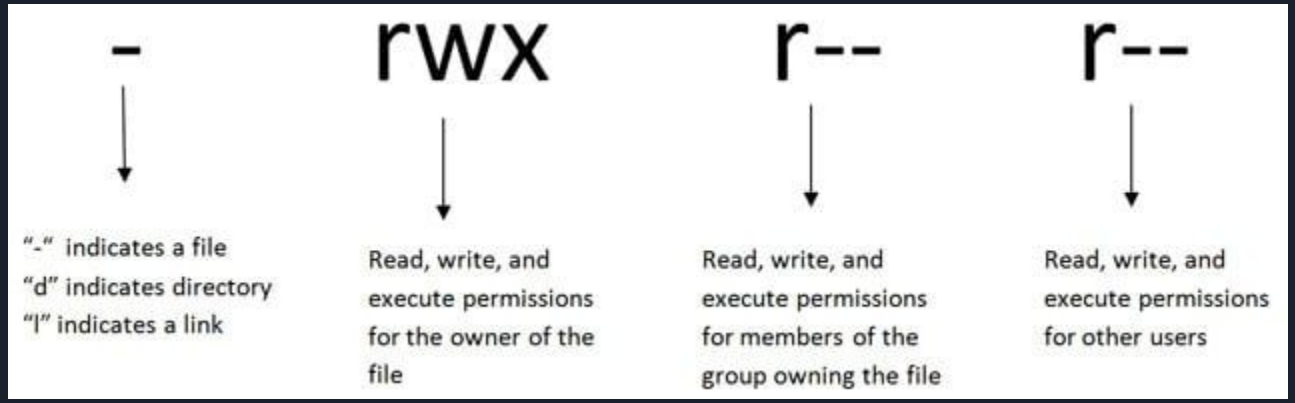
File size
(in bytes)

File Name

- Files that start with . are hidden files.
- . and .. refers to current directory and the parent directory respectively.

File Permission

```
drwxr-xr-x
drwxr-xr-x
-rw-------
-rw-r--r--
-rw-r--r--
```



chmod, chown

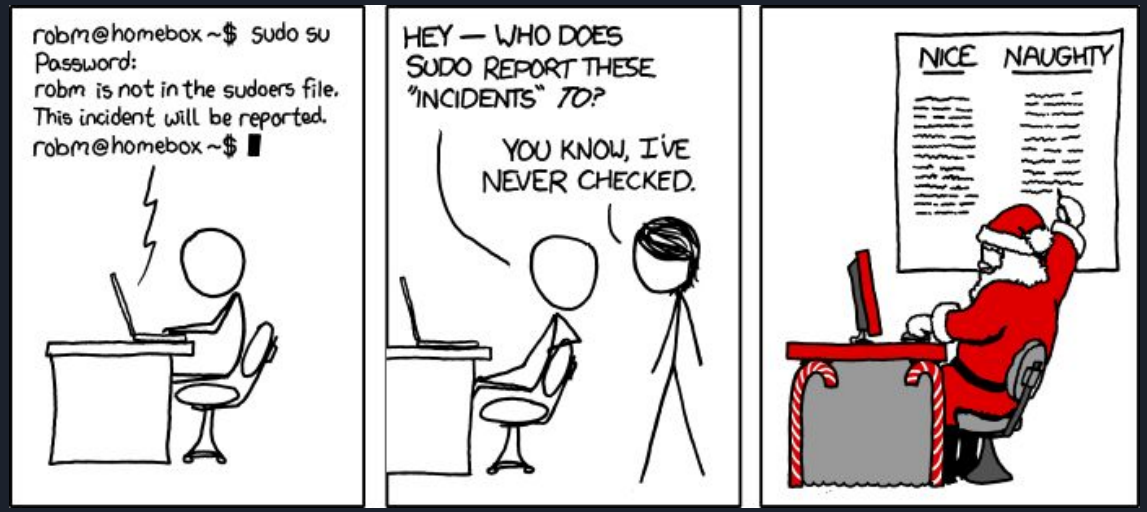
- chown <user>:<group> <file>
- change the file owner to user and (optionally) the user group to group
- chmod <mode> file
- Change file permission of file

- Multiple forms of mode:
- `chmod u+x,go-r haha`
- user add eXecute permission, group and others remove Read permission
- `chmod 751 haha`
- Each number can be decomposed as a sum of 4, 2 and 1 (a bitmask)
- 4 for read, 2 for write and 1 for execute
- read write execute for the user, read and execute for group, only execute for others

User Privilege

The root user is the “administrator”. It has the highest permission that can do virtually anything.

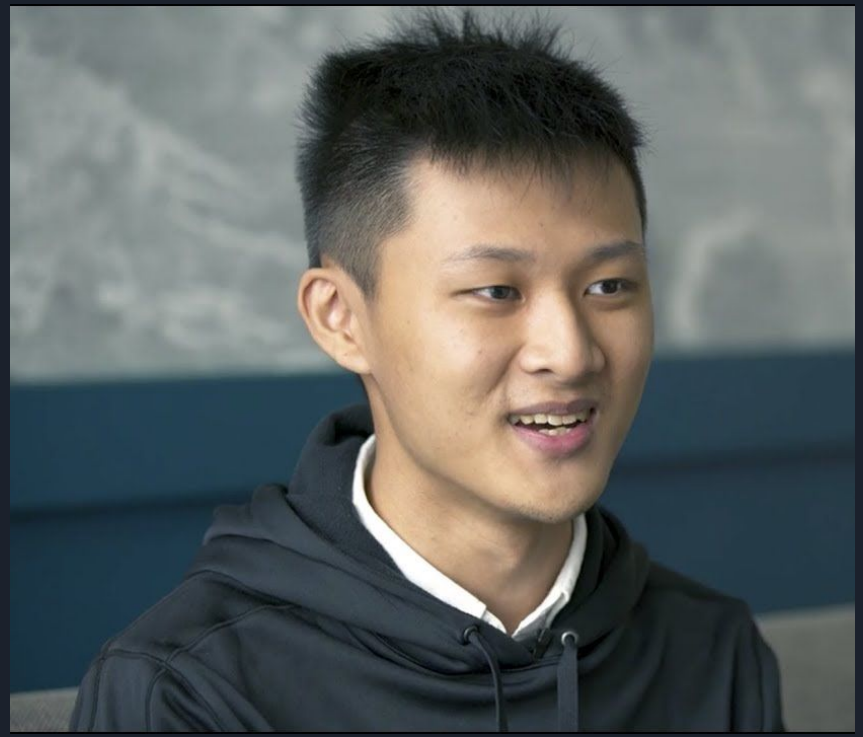
`su`	change user to root
`sudo <command>`	execute command as root if you are sudoer



Basic Commands

`cd <dir>`	**C**hange **D**irectory (aka folder)
`ls <dir>`	**LiS**t directory contents
`mv <src> <dest>`	**MoV**e file from src to dest
`cp <src> <dest>`	**CoP**y file from src to dest
`rm <file>`	**ReMo**ve file
`mkdir <dir>`	**MaK**e a **DIR**ectory
🐱`cat <file>`	Print the content of file (cat stands for **CON**catenate)
`echo <string>`	Print a string to standard output
`pwd`	**P**rint the absolute path of **W**orking (current) **D**irectory
`touch <file>`	Create blank file or change access and modification time to current time

filedescriptor



File Descriptor

- File descriptor is a **unique identifier** for a **file** or other input/output resource. (from wiki)
- There are 3 standard file descriptors (streams):
- 0: Standard input (takes input from your keyboard)
- 1: Standard output (the terminal screen)
- 2: Standard error (outputs error)

Other Useful Commands

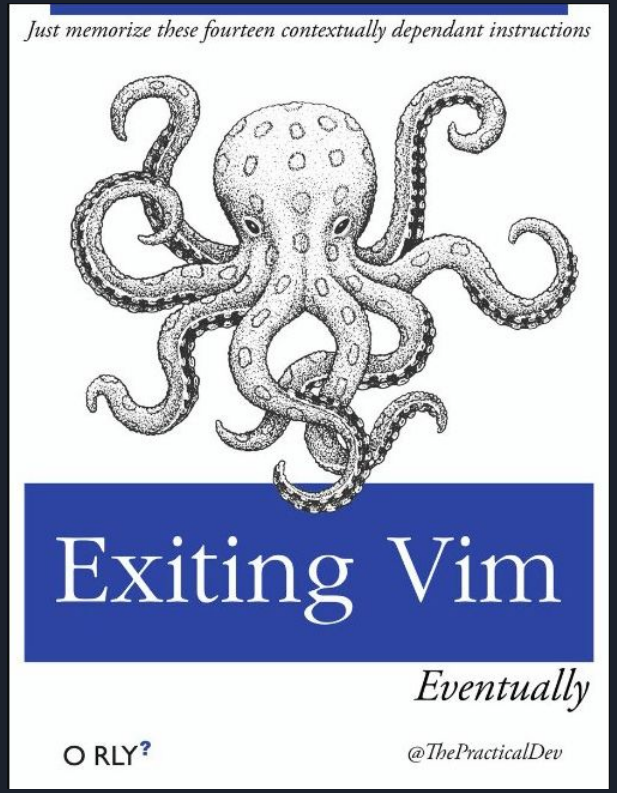
- `clear` — Clear the terminal screen
- `strings <file>` — print any sequence of printable strings in a file
- `head/tail <file>` — print the first/last lines of a file
- `more <file>` — view the contents of a file one screen at a time
- `less <file>` — less is more (but better)
- `file <file>` — Identify the type of file

```
┌──(kali㉿kali)-[~]
└─$ file not_an_executable.png
not_an_executable.png: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux
-x86-64.so.2, BuildID[sha1]=c9603b9489e4f13f892017c03728a7207a0e8b51, for GNU/Linux 3.2.0, not stripped
```

Text Editor in the terminal

- vim, nano, emacs, ~~pico, ed~~…
- I personally use (neo/mac)vim

- vim is very powerful, but can seem cryptic at first
- Enter vim in **normal mode**
- in **normal mode**, press **`i`** to enter **Insert mode** to type
- in **insert mode**, press **`esc`** to get back to **normal mode**
- in **normal mode**, press **`:wq`** to save (w) and quit (q)
- alternatively, press **`ZZ`** in normal also does save quit



- nano is perhaps more beginner-friendly
- type straight away
- press ^X (Ctrl+X) to exit, enter Y to confirm save (and leave)
- The shortcuts are on the bottom of the screen

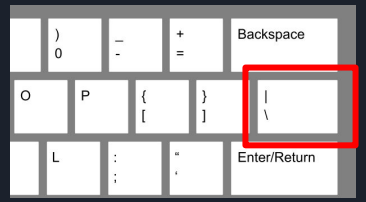
Searching Stuff

`grep "text" <file>`	searches text from file, can also accept regular expressions (won't explain)
`find <format>`	search file using some format

What if I want to search for text in program output?

Redirect

- The pipe operator |
  - Connects the output of the left command to the input of the right command
  - e.g. `curl --help all | grep "POST"`
  - search the help text of `curl` for the word POST




```
┌──(kali㉿kali)-[~]
└─$ curl --help all | grep "POST"
 -d, --data <data>   HTTP POST data
     --data-ascii <data> HTTP POST ASCII data
     --data-binary <data> HTTP POST binary data
     --data-raw <data> HTTP POST data, '@' allowed
     --data-urlencode <data> HTTP POST data url encoded
```

Redirect

- `<command> > <file>`
  - feed the output of the command to `file`
- `<command> >> <file>`
  - APPEND the output of the command to `file`
- `<command> < <file>`
  - Use the `file` as input to the command

- e.g. `ls -la > ls.txt`
- put the output of `ls -la` to the file `ls.txt` (will OVERWRITE the previous ls.txt)

Other Operators

- semicolon ;
  - Run two commands in one line
- and &&
  - take the exit code of two commands and take AND (exit code 0 = success is **true**)
- or ||
  - logical or
- Short Circuit Evaluation
  - For A && B, if A is false, then A && B must be false! So don't bother running B.
  - For A || B, if A is true, then A || B must be true! So don't bother running B.
- Comment #
  - everything after # are regarded as comment
- $(<command>)
  - Run command in the bracket and replaces it with the output
  - e.g. `echo "I am $(whoami)"`

```
┌──(kali㉿kali)-[~]
└─$ echo "I am $(whoami)" # prints who I am
I am kali
```

Package Manager

When you need to install (or update/upgrade stuff…)

- `sudo apt install <packages>`
  - e.g. `sudo apt install python3-pip` to install `python3-pip`
- `sudo apt update`
- `sudo apt upgrade`

Environment Variable

Variables to use in a shell

```
env
cat /proc/self/environ
```

```
$PATH
```

```
test=1
echo “test is ${test}”
```

gone when you close the shell

ssh and nc

- `ssh`: Connect to some servers using the SSH protocol (via TCP port 22)
  - `ssh user@1.2.3.4`
- `nc`: arbitrary TCP and UDP connections
  - `nc 1.2.3.4 1234`
    connect to port 1234 of 1.2.3.4
  - We will use nc to play pwn or crypto challenges.

Communicate with the Internet

- `curl`: transfer a URL
  - mentioned last time
- `wget`: network downloader
- `elinks`: a browser on the command line

Linux file structure

Tree structure, just like Windows/Mac

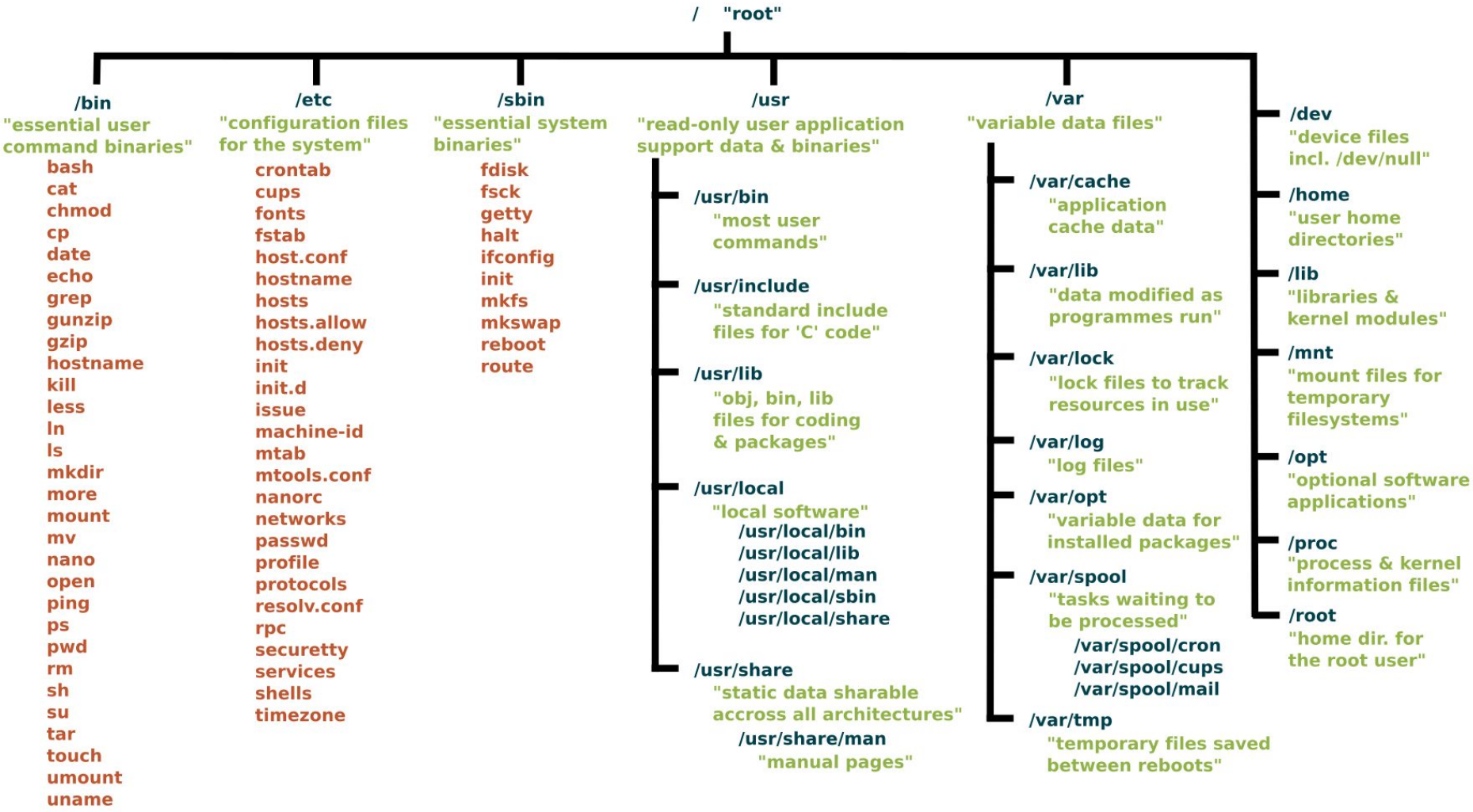
/ <- root, the top of everything

then

e.g. /etc/passwd <- in /etc/ folder file passwd

~ <- this is equivalent to $HOME, the home directory

Hardware file /dev


/ "root"
/bin
"essential user command binaries"
bash
cat
chmod
cp
date
echo
grep
gunzip
gzip
hostname
kill
less
ln
ls
mkdir
more
mount
mv
nano
open
ping
ps
pwd
rm
sh
su
tar
touch
umount
uname
/etc
"configuration files for the system"
crontab
cups
fonts
fstab
host.conf
hostname
hosts
hosts.allow
hosts.deny
init
init.d
issue
machine-id
mtab
mtools.conf
nanorc
networks
passwd
profile
protocols
resolv.conf
rpc
securetty
services
shells
timezone
/sbin
"essential system binaries"
fdisk
fsck
getty
halt
ifconfig
init
mkfs
mkswap
reboot
route
/usr
"read-only user application support data & binaries"
/usr/bin
"most user commands"
/usr/include
"standard include files for 'C' code"
/usr/lib
"obj, bin, lib files for coding & packages"
/usr/local
"local software"
/usr/local/bin
/usr/local/lib
/usr/local/man
/usr/local/sbin
/usr/local/share
/usr/share
"static data sharable accross all architectures"
/usr/share/man
"manual pages"
/var
"variable data files"
/var/cache
"application cache data"
/var/lib
"data modified as programmes run"
/var/lock
"lock files to track resources in use"
/var/log
"log files"
/var/opt
"variable data for installed packages"
/var/spool
"tasks waiting to be processed"
/var/spool/cron
/var/spool/cups
/var/spool/mail
/var/tmp
"temporary files saved between reboots"
/dev
"device files incl. /dev/null"
/home
"user home directories"
/lib
"libraries & kernel modules"
/mnt
"mount files for temporary filesystems"
/opt
"optional software applications"
/proc
"process & kernel information files"
/root
"home dir. for the root user"

Useful location

./ ../
/dev/null /dev/urandom
/etc/passwd /etc/shadow
~/.bashrc
~/.bash_history
~/

/bin/ for program
/lib/ for library
/usr/bin/ for local program
/usr/share/ for some resource
/etc/ for weird use
/tmp/ <- temp

wordlist:
rockyou in /usr/share/wordlists

Command Injection

Execute arbitrary command in other people's shell.

Command Injection 1: http://chal.firebird.sh:35001/
Command Injection 2: http://chal.firebird.sh:35002/
Command Injection 3: http://chal.firebird.sh:35003/

Warning When allowing user-supplied data to be passed to this function, use escapeshellarg() or escapeshellcmd() to ensure that users cannot trick the system into executing arbitrary commands.

```
ping -c 1 -W 1 1.1.1.1
```

Just like last time with SQL injection!

```
ping -c 1 -W 1 ;
```

```
ping -c 1 -W 1 ;cat /flag
```

Local File Inclusion

Local File Inclusion is an attack that allows remote attacker to read any file on the server.

natas7 (Username: `natas7` / Password: `7z3hEENjQtflzgnT29q7wAvMNfZdh0i9`)

The flag is located `/etc/natas_webpass/natas8`, can you try read it?

Encoding

Encoding

Although we can read English, Chinese, C, Python, … Computers can only read 0 or 1s. How does the computer translate from the 0s and 1s to English (or Chinese or whatever) Characters?

- Encoding.
- Encoding is a reversible mapping of characters.
- Encoding is NOT encryption.

ASCII

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

Open up a text file in Hex editor

It is likely encoded in ASCII

xxd <file>

```
00000000: 5368 6172 6b73 2061 7265 2061 2067 726f  Sharks are a gro
00000010: 7570 206f 6620 656c 6173 6d6f 6272 616e  up of elasmobran
00000020: 6368 2066 6973 6820 6368 6172 6163 7465  ch fish characte
00000030: 7269 7a65 6420 6279 2061 2063 6172 7469  rized by a carti
00000040: 6c61 6769 6e6f 7573 2073 6b65 6c65 746f  laginous skeleto
00000050: 6e2c 2066 6976 6520 746f 2073 6576 656e  n, five to seven
00000060: 2067 696c 6c20 736c 6974 7320 6f6e 2074   gill slits on t
00000070: 6865 2073 6964 6573 206f 6620 7468 6520  he sides of the
```

Common encoding

Name	Example	Comment
ASCII	`Open Innovation Lab :)`	
UTF-8	`Open Innovation Lab` 😄	中 あ 😄 <- CJK/Emojis...
Binary (BIN)	`01001111 01110000 01100101 01101110 00100000 01001001 01101110 01101110 01101111 01110110 01100001 01110100 01101001 01101111 01101110 00100000 01001100 01100001 01100010`	ASCII in Binary
Decimal (DEC)	`79 112 101 110 32 73 110 110 111 118 97 116 105 111 110 32 76 97 98`	ASCII in Dec
Hexadecimal (HEX / Base16)	`4f 70 65 6e 20 49 6e 6e 6f 76 61 74 69 6f 6e 20 4c 61 62`	ASCII in Hex
Base32	`J5YGK3RAJFXG433WMF2GS33OEBGGCYQ=`	
Base64	`T3BlbiBJbm5vdmF0aW9uIExhYg==`	
Base58	`FYVbxEooaXJ4Hp6EPZXHRrSB9s`	Used in Bitcoin

See also: UTF-8 with BOM, Big5, GBK, Base32, Base85, ROT13, Morse code, Pigpen cipher....

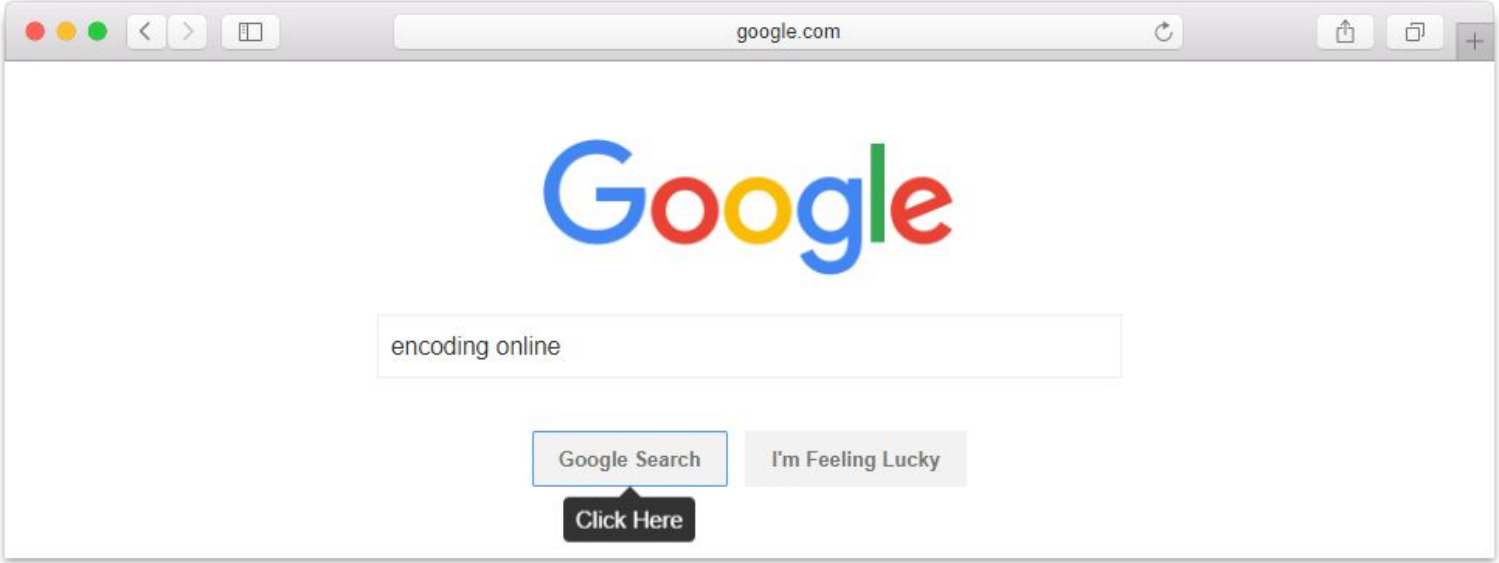
What about other encodings?

Step 1
Visit google.com

Step 2
Type your question.

Step 3
Click the button.

That's it!



Python Basics🐍

Programming Language

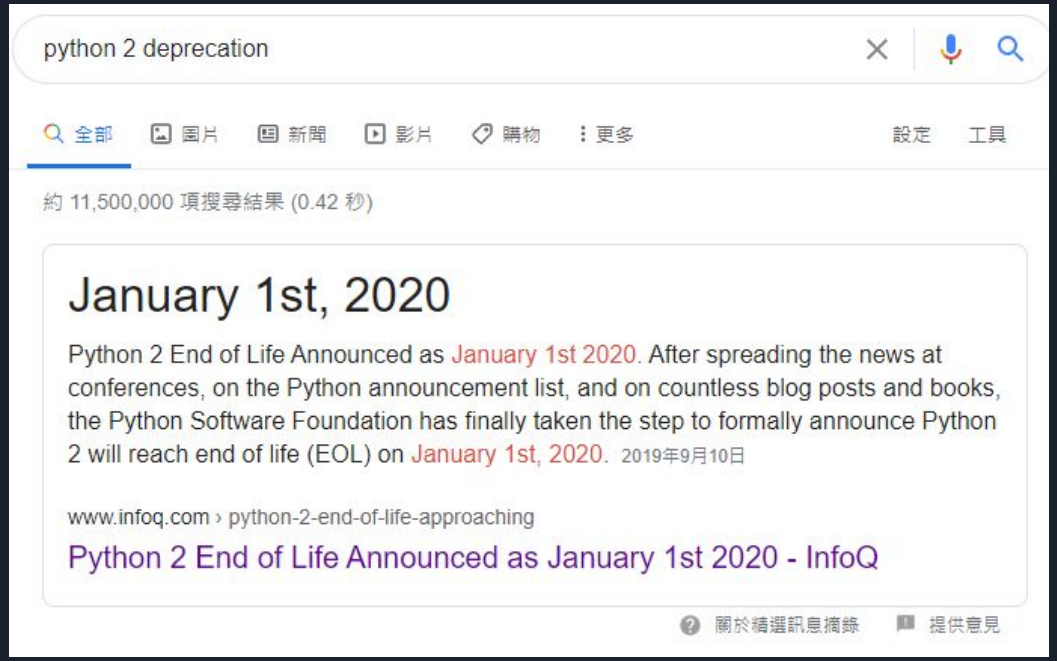
- We want to use a program to help us to the job :)
- We will use python 3.

Why Python?

- Interpreted language: No compilation!
- Lot's of libraries!
  - We will use pwntools to help us.
- Dynamically Typed: No need to define type of variable
- Level by indentation
- Many CTF writeups are written in python

Python 2 or Python 3?

Python 2 is dead. Yay!



Contents

- Variable
- Array (list)
- Logic control flow
- IO
- function
- file IO
- List slicing
- pwntools

Before starting...

To run python3, run `python3`

To execute a python script, do `python3 <file>`

Of course

```
print("Hello world!")
```

Variables

No need to define the type of variable.

```
a = 1                    # integer
a = 1.0                  # floating point
a = '1'                  # String
a = "1"                  # String
a = b"1"                 # bytes
a = [1,2,3,4]            # List
a = {"a": 1, "b":2}      # Dictionary (key-value map)
a = True                 # Boolean
a = False
a = None                 # NoneType
```

Type conversion

```
b0 = b'some strings' # b0 is a 'bytes' object
print(b0)
print(b0.decode())   # convert the bytes to string with default utf-8 encoding

b1 = b'\xa4\xa4\xa4\xe5\xb4\xfa\xb8\xd5'
print(b1)
print(b1.decode('big5')) # convert the bytes to string with big5 encoding

s0 = '中文' # s0 is a string
print(s0)
print(s0.encode()) # convert the strings to bytes with utf-8 encoding

str(1)         # convert 'anything' to string
int("1")       # convert 'anything' to int
int("0010", 2) # convert 1010 to int in base 2
```

Operators

```
+ - * /    arithmetics
//         floor division
%          modulo
**         power
|          bitwise or
&          bitwise and
^          bitwise xor
```

Calculation:

```
a = 3 + 1      # can replace '+' by any one of above
a += 2         # a = a + 2
```

Comment

Comment:

```
# write comment here


'''

write comment here
'''
```

IO

Input:

```
user_input = input("What is your input? ")
```

Output:

```
print()
print(123)
print('abc' + 'def')
print('a'*100)
print('age: ' + str(15))
```

List

```
a = ["a", "b", "c", "d", "e"]
b = [1, 2, 3, 4]
arr = [a[i] for i in b]
c = [1, "a", 1.234, True, None]
```

```
lst = []
lst.append('1')
print(lst[0])
lst.remove('1')

lst = range(10)
print(len(lst))
print(lst.pop())
print(lst.pop(5))
```

range(), Slicing

```
list(range(10))          #output: [0,1,2,3,4,5,6,7,8,9]

list(range(4, 10)) #output: [4,5,6,7,8,9]

list(range(4, 10, 2))    #output: [4,6,8]

list(range(4, -4,-1))    #output: [4,3,2,1,0,-1,-2,-3]


s0 = [0,1,2,3,4,5,6,7,8,9,10]

print(s0[2:10])    #output: [2,3,4,5,6,7,8,9]
print(s0[2:])      #output: [2,3,4,5,6,7,8,9,10]
print(s0[:10])     #output: [0,1,2,3,4,5,6,7,8,9]
print(s0[::-1])    #output: [10,9,8,7,6,5,4,3,2,1,0]


s1 = "Hello World!"

print(s1[2:10])    #output: llo Worl

print(s1[2:])      #output: llo World!

print(s1[:10])     #output: Hello Worl

print(s1[::-1])    #output: !dlroW olleH
```

String Methods

```
print("   <- spaces ->   \n\n\n".strip()) # <-spaces->
                            (the spaces and newlines are trimmed)

print("The answer is: {}".format(42))     # The answer is: 42

print("a b c".split(' '))                 # ['a', 'b', 'c']
```

Logic control flow

- If

```
if 1==1:
    print("Yes!")
```

- for

```
for i in range(10):
    print("i = {}".format(i))
```

- while

```
while 2==2:
    print("Infinite loop")
```

If

```
== , != , < , <= , > , >=
condition_1 and condition_2
condition_1 or condition_2

not condition


if not( number>=10 and number<20 ):
     print("input out of range")
elif number == 15:
     print("error")
else:
     print("else")
```

For

For is different than what you would be used to in other languages.

Similar meaning to “for each”

```
a = [1,3,"a",b"b"]
for x in a:
    print(x)
```

```
1
3
a
b'b'
```

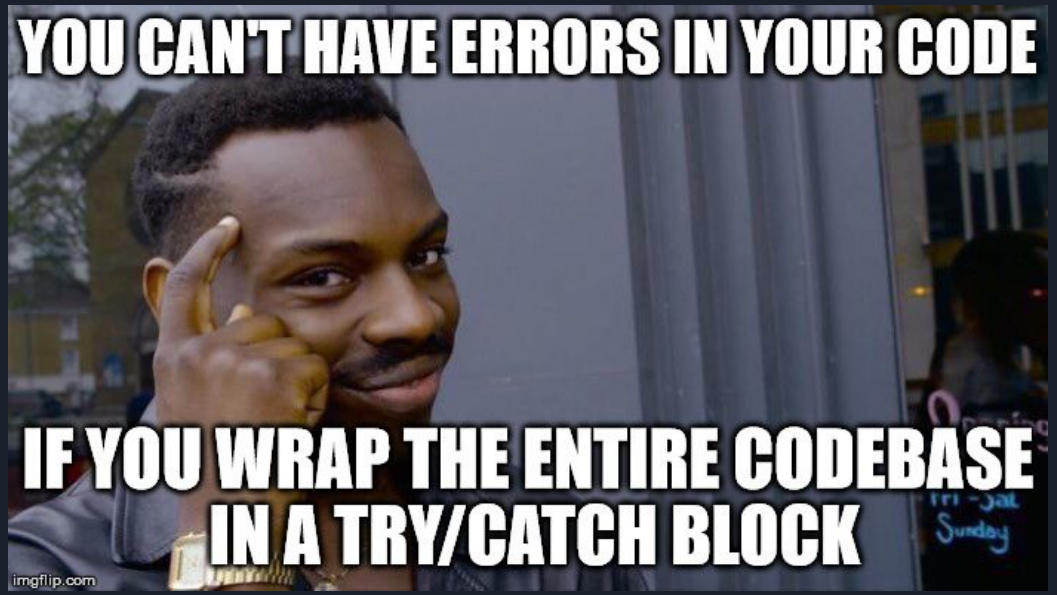
For Loop like the other languages

```
# for(int i=0; i < n; i += step)
for i in range(0,n,step):
    print(i)
```

`break` / `continue` works just like other programming languages

Try harder!

```
try:
    <statements>
except Exception as e:
    <statements>
else:
    <statements>
finally:
    <statements>
```



Wait... else?

else can be used in for, while and try/except blocks

If no break or exception are called, then the else block will execute

```
for element in large_list:
    if element == "haha":
        break
else:
    print("Could not find \"haha\"")
    raise Exception("Why???")
```

Function

```
def sum( a, b ):
    return a + b

def calc( a, b ):
    x = sum( a, b )
    return (x, a - b)

s , diff = calc(1.2, b=2.3)
print(s)
print(diff)
```

Tuple Unpacking (like std::tie in C++)

File IO

```
f = open( filename, mode ) # mode = { r,r+,rb,rb+,w,w+,wb,wb+,a,a+,ab,ab+ }
f.read(size)
f.readline()
f.write(string)
f.close()
```

Example of opening file in 2020

```
>>> with open("some file","r") as fp:
...     print(fp.read())
...
```

Alternatively...

```
with open(<filename>, "r") as f:
    print(f.read())
```

Handles file closing for you. (Useful especially for Windows)

Equivalent to:

```
try:
    f = open(<filename>, "r")
    <statements>
finally:
    f.close()
```

Library

```
python3 -m pip install --user <name>
sudo python3 -m pip install <name>

import <library_name_here>
from <library> import <function> as <alias>
```

- os
- system
- time
- random
- pwn
- itertools
- base64
- urllib , urllib2, md5, socket, requests, image, threading
- pycrypto

from pwn import *

Pwntools

> Pwntools is a CTF framework and exploit development library. Written in Python, it is designed for rapid prototyping and development, and intended to make exploit writing as simple as possible.

Pwntools is a python library allow easier interaction with pwn / nc challenges in CTF

https://github.com/Gallopsled/pwntools

Install with: `sudo python3 -m pip install --upgrade pwntools`

Pwntools

```
from pwn import *

r = remote("ip", port)
r = process("loc")

string = r.recv()
string = r.recvline()
string = r.recvuntil("sth")

r.send(str)
r.sendline(str)

r.interactive()

r.close()
```

http://docs.pwntools.com/en/stable/

Black magic

From python -h

-c cmd : program passed in as string (terminates option list)

- You can print stuff with it just like echo
- Or you can do more (reverse shell)

```
$ python3 -c "print('Hello world!')"
Hello world!
```

Exercise

- Python 1+1
  - I guess there are already enough hints.
  - `nc chal.firebird.sh 35004`

Credit

- Intro slides from Firebird
- Myself

End!

- Feel free to join our discord server for further discussion! We will have other events and invited talks so stay tuned for more CUHKOIL activities.
- We are recruiting CTF players! Join by sharing your write-ups with us. Join the discord server for more details. (CUHK students only)
- Like our facebook page https://www.facebook.com/cuhkoil also for events!

Facebook

Discord
Note: This link is only valid for 50 invites.