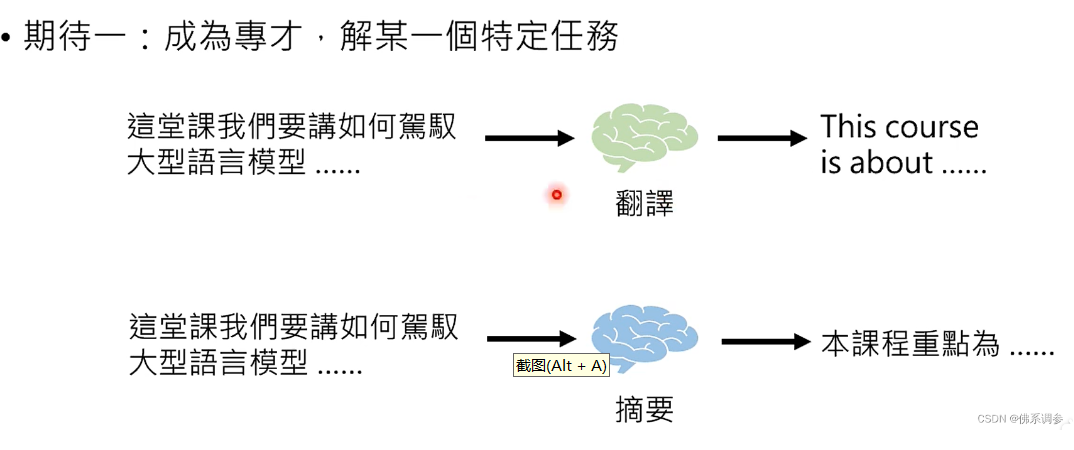
# 大语言模型的两种使用方式（发展的两种方向）Finetune（微调） vs. Prompt（指令）

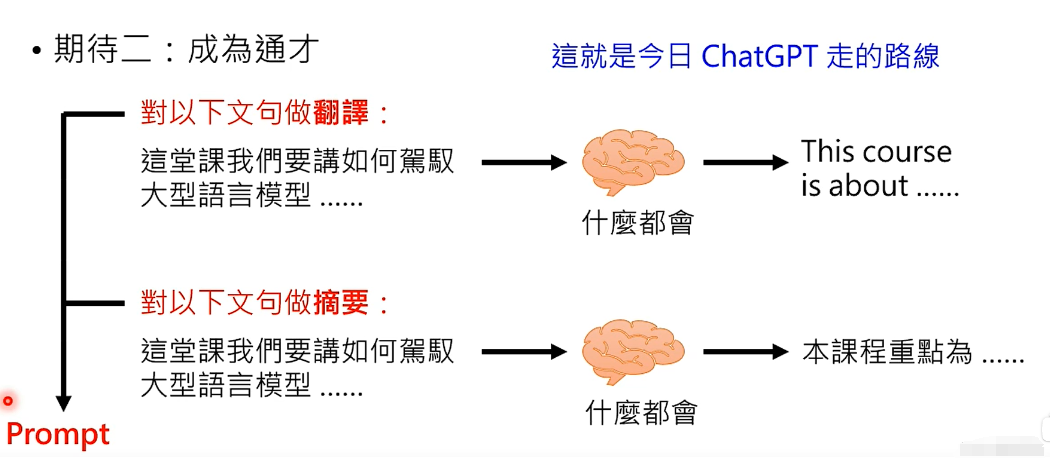
# 一．两种使用方式方简单介绍

### ****成为专才，能够解决某一个特定任务。例如翻译，生成摘要****

****成为专才的好处: 专才在某一个任务上有机会赢过通才****，****例如目前chatGPT的文本翻译性能不如专门的谷歌或者腾讯的专门翻译模型。****



### ****成为通才，解决多种任务****



****成为通才的好处：****

****只需要重新设计prompt就可以快速开发新功能，不用重新写程序****

# 实现这两种功能的方式

## 1.成为专才的做法

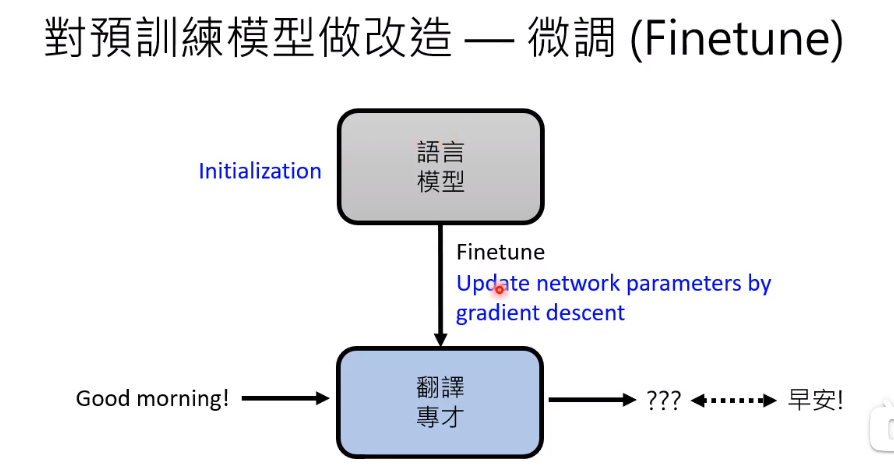
**1.在模型基础上增加新的输出层(head)**

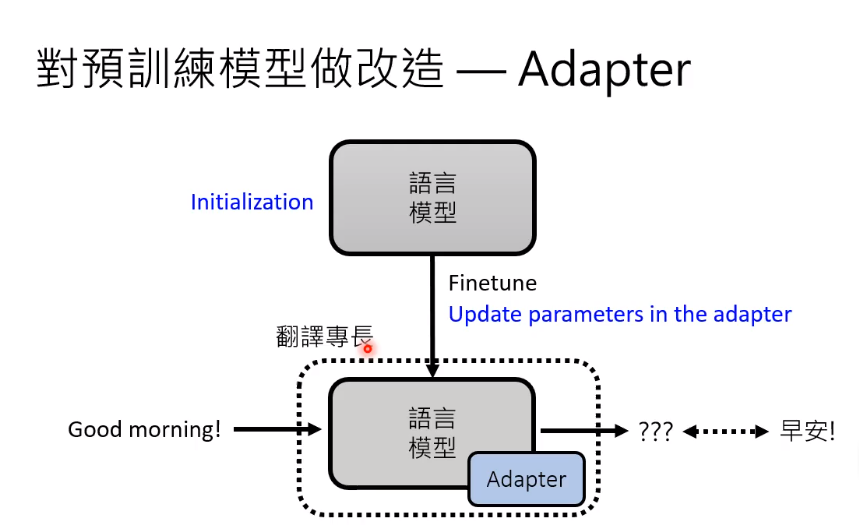
**2.然后采用参数微调Finetune方式进行训练, 需要标注好的数据**

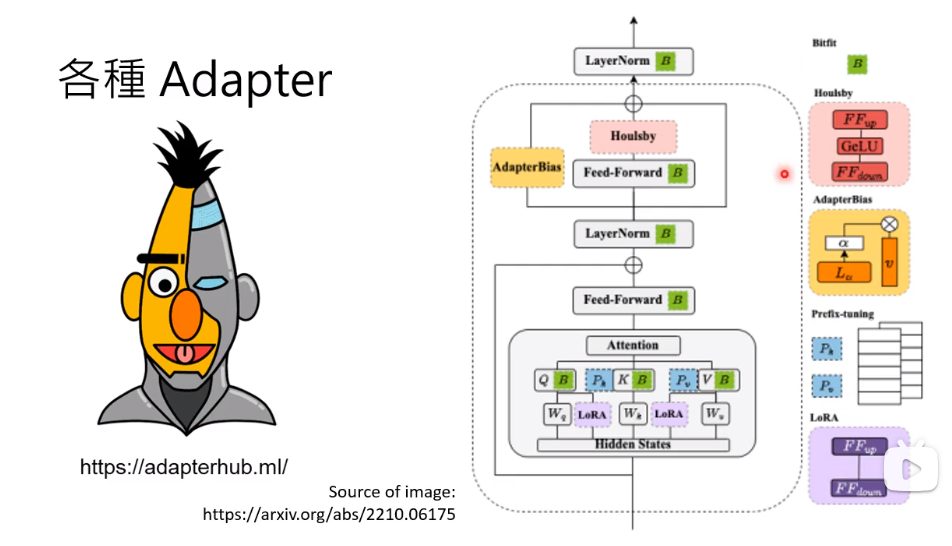
**（1）一种方式是需要对大模型的参数进行梯度更新**

**（2）另一种方式，采用新增的Adapter，可以插在网络的各种位置，只需要对Adapter参数进行微调，大模型本身其他部分的参数是固定不动的。**

**各种Adapter 在https:/adapterhub.ml/中可以查看。下图中为对Transformer中可以添加的几种Adapter示例**







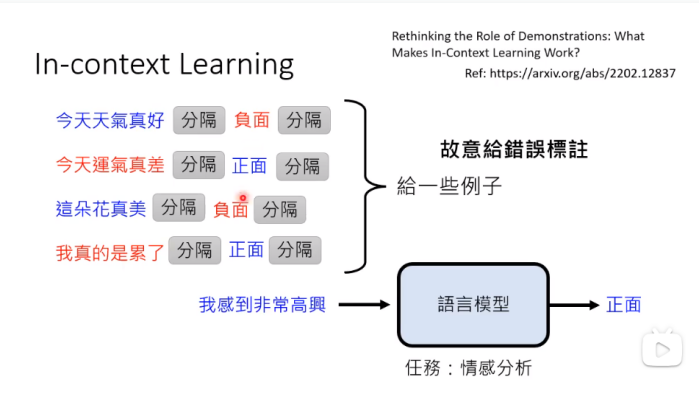
## 2.成为通才的做法

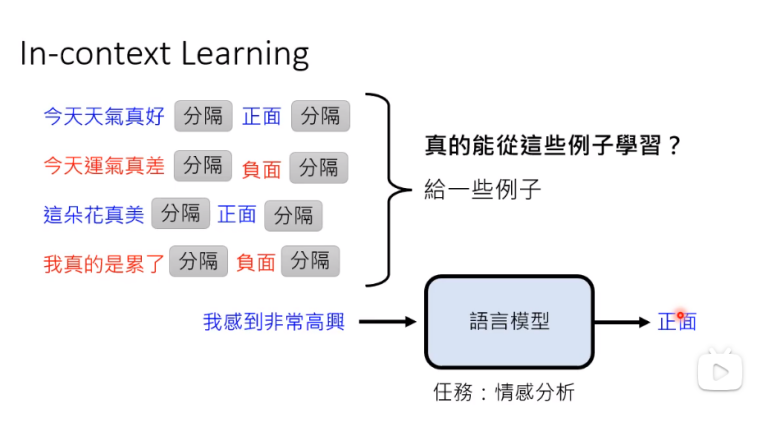
### **（1）In-context learning（也就是Prompting）**

**就是在推理时给模型一些示例作为输入，让大模型知道要完成的任务是什么样的**

**简单来说，就是**模型在不更新自身参数的情况下**，通过在模型输入中带入新任务的描述与少量的样本，就能让模型”学习”到新任务的特征，并且对新任务中的样本产生不错的预测效果。**

**注意：并不是在训练时候给出这样的示例，因为并没有根据这些示例对大模型参数进行更新（另一个角度理解没有对大模型参数进行更新：**大模型预训练好之后参数不动了，但是要完成的任务很多，且任务是各式各样**）**



**图一是给正确的逻辑，图二故意标错，有一篇论文对in-context learning进行了研究，对输入示例语句其中的label（正面或者负面）进行了随机，表明输入时即使给的示例的label是错误的，模型的预测性能也没有降低太多。表明Label其实不重要？（但是另外一篇论文研究表明，示例label错误的比例越高，模型性能越低，并且模型参数量越大的话性能降低的越多，这样前面工作的结论可能是因为采用的模型比较小）**

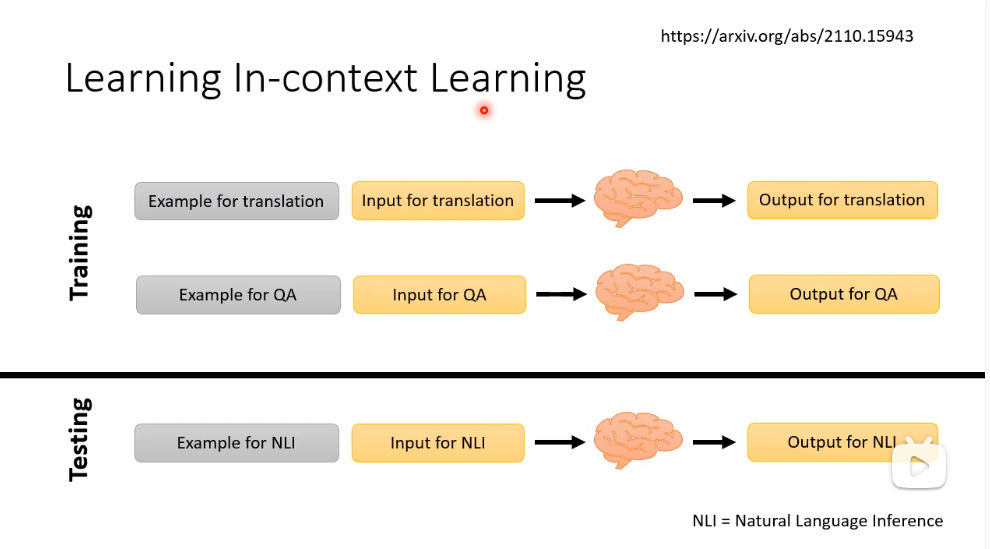
**将输入示例语句其中第一个分隔符之前的语句替换为不同场景/任务的句子，模型性能降低很多，表明示例任务的相关性很重要**

**给出的示例的数量也不是越多越好，越多的话性能反而可能会降低。一般4个或者8个就够了。**

****推论：大模型本来就有做某种任务的能力（比如情感分析），只是需要被指出需要做的任务的种类（比如情感分析），而输入的示例语句就是起到这样的作用（唤起大模型的能力，即prompt）****

### **Learning In-context learning**

**在训练过程中采用In-context learning，做到真正的learning（模型参数进行了更新）**

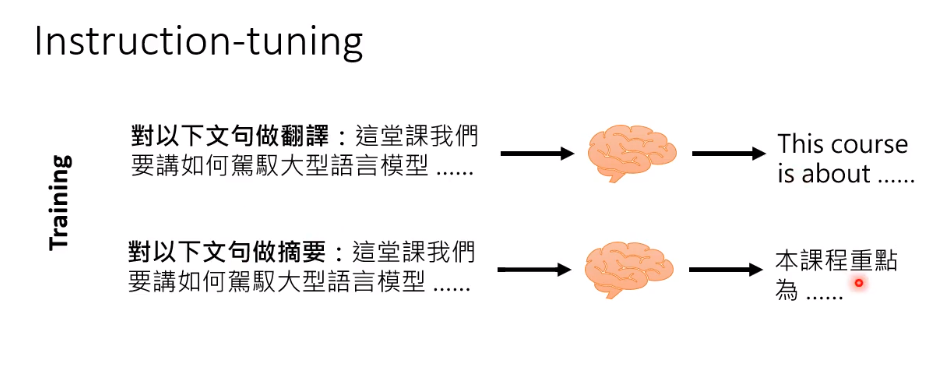


### **Instruction-tuning（指令微调）**

**做法：给出问题的描述**

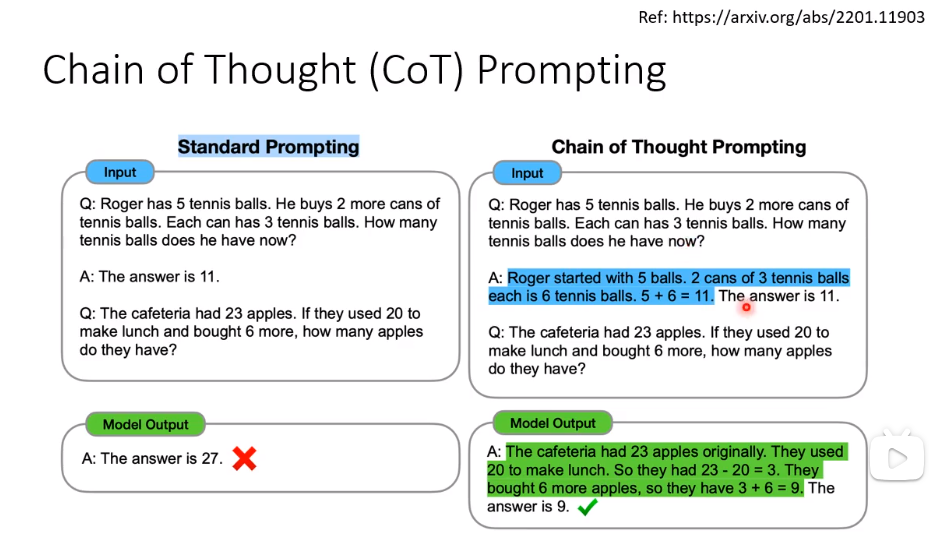
**它的核心思想是尽可能收集不同类型的自然语言处理任务（包括理解和生成），并使用自然语言设计对应的任务指令，让模型试图理解不同任务的指令与特性，最终通过语言模型生成的方式完成不同任务的训练**

**注意:在训练和推理过程中都会用到指令微调，因此需要构建指令微调的训练集**

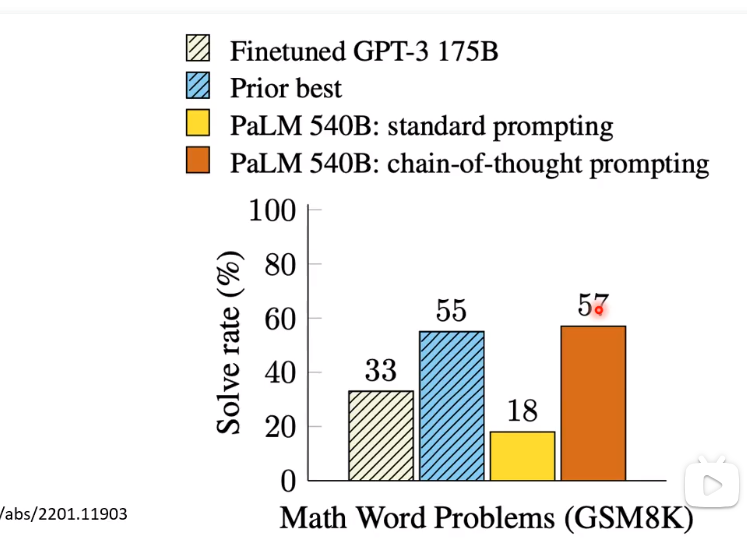


### **chain of Thought（CoT）Prompting**

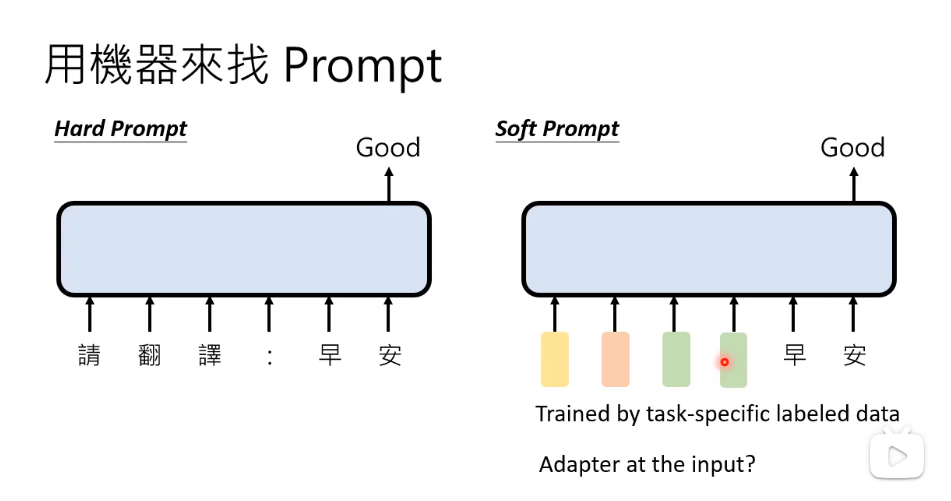
**做法：在给模型示例的时候，顺便给出推论的中间过程**



**这是对比的结果如下：**



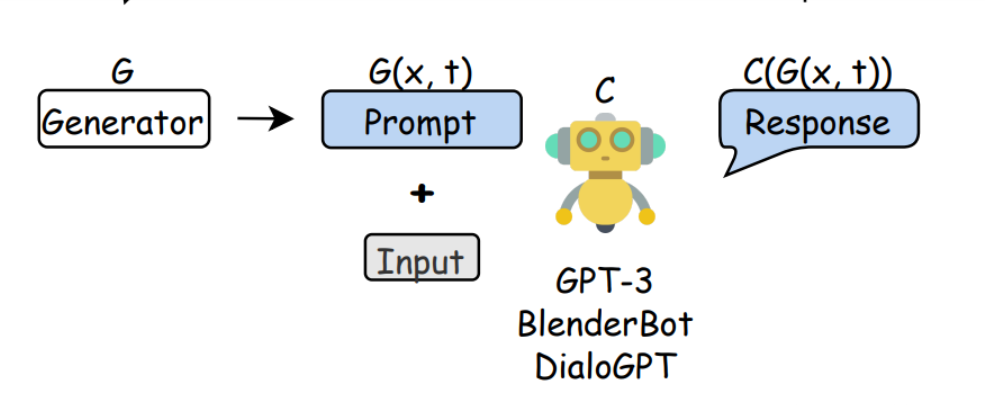
## **用机器来找prompt的方式**



**什么叫 Hard Prompt 呢？就是我们输入给机器的东西是文字。那还有另外一种做法是给机器 Soft Prompt，就是你给机器的额外输入，你给机器的指令，并不是用人类看得懂的语言来描述的，而是一堆连续的向量，而这些向量是可以 train 的，你可以把这些向量想成模型参数的一部分，它们可以跟着 label data 进行调整。**

**如何让机器自己找prompt？**

**可以用 reinforcement learning 的方法来硬造 prompt**



**你现在有一个大型的 Language Model，给一个 Input 输出一个 Response，你直接用另外一个 Language Model 生成一个句子，这个句子就当作 GPT-3 的 prompt，那期待这个句子可以影响 GPT-3，让它 output 的结果是我们想要的。**

**那这个 Generator 它怎么知道要怎么产生合适的 prompt 呢？你就要做 reinforcement learning，你要有一个方法来判断说现在大型 Language Model 的 output 是不是我们要的，如果是就得到高的 reward，如果不是就得到低的 reward，把 reward 再 丢给这个 Generator 让它进行学习，期待它产生出来的 prompt 可以影响 GPT-3，让它出来的结果是我们想要的，所以你确实可以用 reinforcement learning 的方法来硬学一个 Generator 产生一个 prompt 来达成我们的目标。**

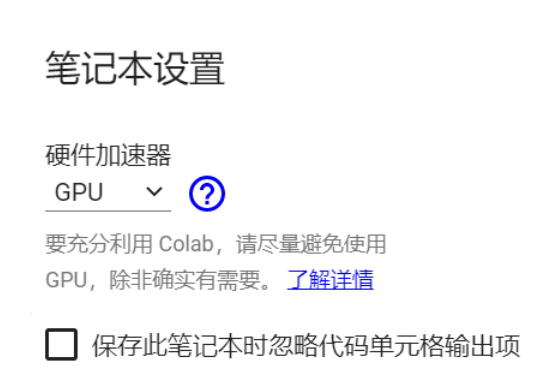
# 总结

****我们主要学习了对 LLM 的两种不同的期待：Finetune 和 Prompt 即专才和通才，像 BERT 这种通过加外挂和微调参数可以成为专才，而 ChatGPT 这种则是一个通才，我们讲了有关通才 prompt 的三种方式，一种是 In-context Learning 通过范例来进行 prompt，一种是 Instruction-tuning 通过题目叙述来进行 prompt，还有一种是 Chain of Thought，不仅给题目叙述和范例还给推论过程，最后我们还简单聊了下如何让机器自己来找 prompt。****

# Colab Tutorial

在运行代码之前就要选择用CPU还是GPU，如果在运行过程中切换会丢掉之前所有运行结果。更换位置：





另需注意，免费GPU是有使用限制的

在代码块中用 ! 可以运行shell语句。

用 ! 开启一个新的shell，运行后就杀死进程，但是如果用 % 就会影响整个notebook进程，因此如果要运行cd（更换根目录）就要用 % 。

notebook的根目录是临时目录，notebook运行结束就没了。这些临时文件可以下载到本地，也可以挂载谷歌云端硬盘（mounting google drive）到根目录下，挂载后对存放路径下MyDrive目录的数据修改就会实时同步到云端。



挂载代码见colab文件。也可以直接点击这个图标自动生成代码：

Colab中常用的Linux命令行：

ls : List all files in the current directory

ls -l : List all files in the current directory with more detail

pwd : Output the working directory

mkdir : Create a directory named

cd : Move to directory named

gdown : Download files from google drive

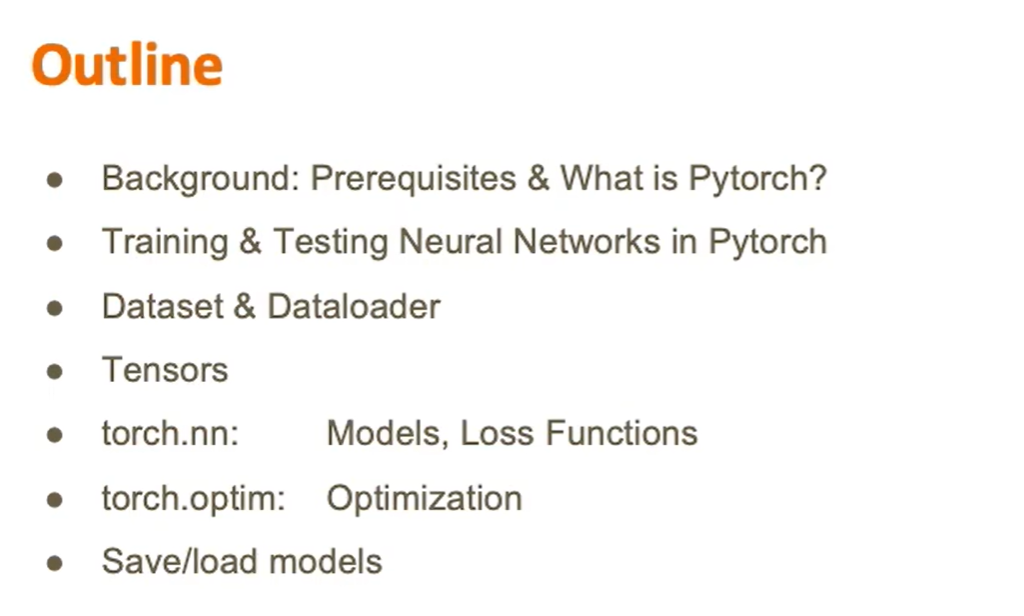
wget : Download files from the internet

python <python\_file> : Executes a python file

参考学习资料：开始使用Google Colaboratory（Coding TensorFlow）2

# Pytorch Tutorial

**大纲**



按照此大纲的各个部分，例如数据处理以及会使用到的函数、操作和代码请参考以下链接：

[李宏毅2021春季机器学习课程视频笔记1：Introduction, Colab & PyTorch Tutorials, HW1\_李宏毅hw1作业2021-CSDN博客](https://blog.csdn.net/PolarisRisingWar/article/details/117229594?ops_request_misc=%7B%22request%5Fid%22%3A%22172195745916800207098583%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=172195745916800207098583&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-117229594-null-null.142^v100^pc_search_result_base7&utm_term=%E6%9D%8E%E5%AE%8F%E6%AF%85%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0 colab tutorial&spm=1018.2226.3001.4187)