Lesson 1

Python Basics

Haoxiang Sun @ RUC CS

写在前面

- 在 Python 语言基础的学习中，请尽量杜绝 copilot 和 chatGPT 的使用
- 在调用各种包时 numpy，pytorch，pandas，**笔者认为**可以借助 GPT
- 仔细阅读各种包的文档，或者咨询 GPT 来获取用法（保持怀疑的态度）
- 练习选做
- Demo部分尽量同时在本地敲一遍

配置 Anaconda / Miniconda 环境

- “Miniconda is a free, miniature installation of Anaconda Distribution that includes only conda, python, …”
- Miniconda 更适合使用命令行环境的用户
- 从 https://docs.anaconda.com/miniconda/ 下载 miniconda 安装包
- (demo)

配置 VSCode 与 Jupyter Notebook

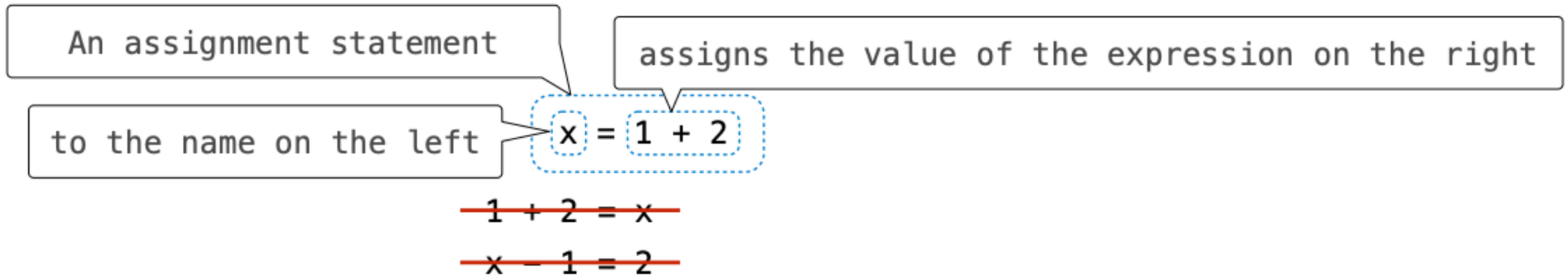
- 可以选择安装 code runner 来运行（不建议）
- 建议使用 Miniconda + VSCode + Python 插件 + Jupyter Notebook 插件
- (demo)

Get started !

- 在命令行输入 python / python3 来开始尝试 Python 解释器和交互模式
- C/C++ 是编译性编程语言，Python 是解释性编程语言
- 如何安装需要使用的包？（PyTorch, vLLM, …）
- `pip install torch` `pip uninstall torch`，如果遇到网络问题，请使用清华等pip源

- Python 是一种严格缩进的语言，建议统一使用四空格缩进（Tab）
- (demo)

Assignment Statements 赋值语句



The expression (right) is evaluated, and its value is assigned to the name (left).

Operators 基本的运算符

demo

```
a = -1234
b = 12.34
c = "hello, w"
d = "orld"

a + b
a - b
a * b
a / b
13 / 3 - 13 // 3
13 / (3 - 13) // 3
13 ** 2
13 ** 0.5
c + d
c * 3
print("hello world!")
```

- +
- -
- *
- /
- //
- %
- **

Operators 基本的运算符

demo

```
p = (1 > 3) # p: False
q = ("apples" < "banana") # q: True

p and q # False
p or q # True
(not p) and q # True
```

- >
- <
- >=
- <=
- !=
- ==
- And
- Or
- Not

Assignment Statements 赋值语句

demo

```
a = b = 0

def f(x, y):
    x *= 2
    y += 3
    return x, y

c, d = f(20, 24)
# what is c & d ?
```

杂项

- 变量需要先声明再使用
- 命名规则：可以使用字母、数字、下划线，可以使用中文
- 建议使用含义明确的下划线连接的英文单词　num_test_sample
- 基本的类型：
- 使用 # 来开始注释
- 可以借助 ‘” 来实现多行注释
- 请写清晰的注释
- None: lack of value

demo

```
a = 3 # int
b = 3.3 # float
c = True # bool
d = "hello!" # string
e = (5, 4, 3) # tuple
f = {'Alex': 3, 'Bob': 10, 'John': 9} # dictionary
g = {4, 3, 2} # set
h = [4, 3, 2, 7] # list
```

字符串基础

- Python 中单引号 `'` 和双引号 `"` 使用完全相同。
- 使用三引号(`'''` 或 `"""`)可以指定一个多行字符串。
- 转义符 `\`。
- 反斜杠可以用来转义，使用 `r` 可以让反斜杠不发生转义。 如 **r"this is a line with \n"** 则 `\n` 会显示，并不是换行。

F-string & format

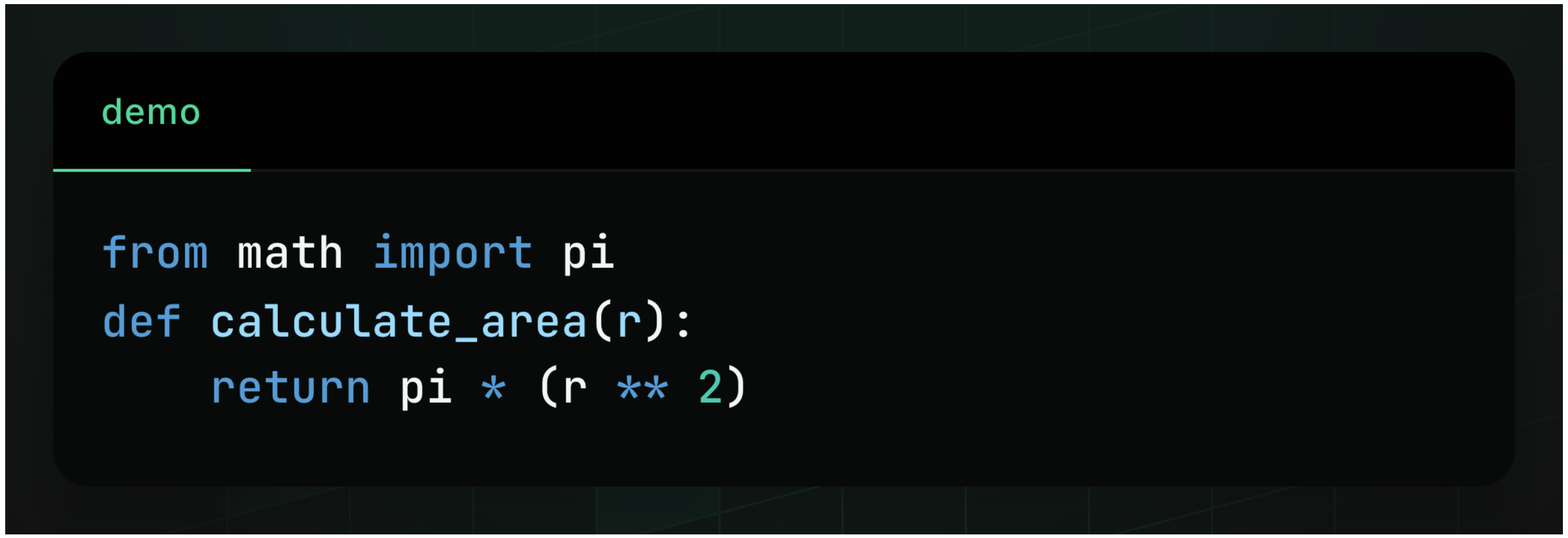
- 使用 f-string 来进行格式化输出
- 方式可以非常灵活
- (demo)

input() 和 print()

- print() 默认换行
- 无法使用 input() 来直接读入数字，请使用 int()
- （关注每个object的类型）
- (demo)

函数

- 如何复用代码?

- 

- (Demo)

函数

demo

```
def convert_html_to_txt(website):
    # load the website and convert it into txt
    return content

def find_infos(content):
    # extract the content and find all job positions
    return jobs

def analysis(jobs):
    # analysis the job positions
    return results

website = input("请输入你想分析的网站")
content = convert_html_to_txt(website)
job_list = find_infos(content)
print(analysis(job_list))
```

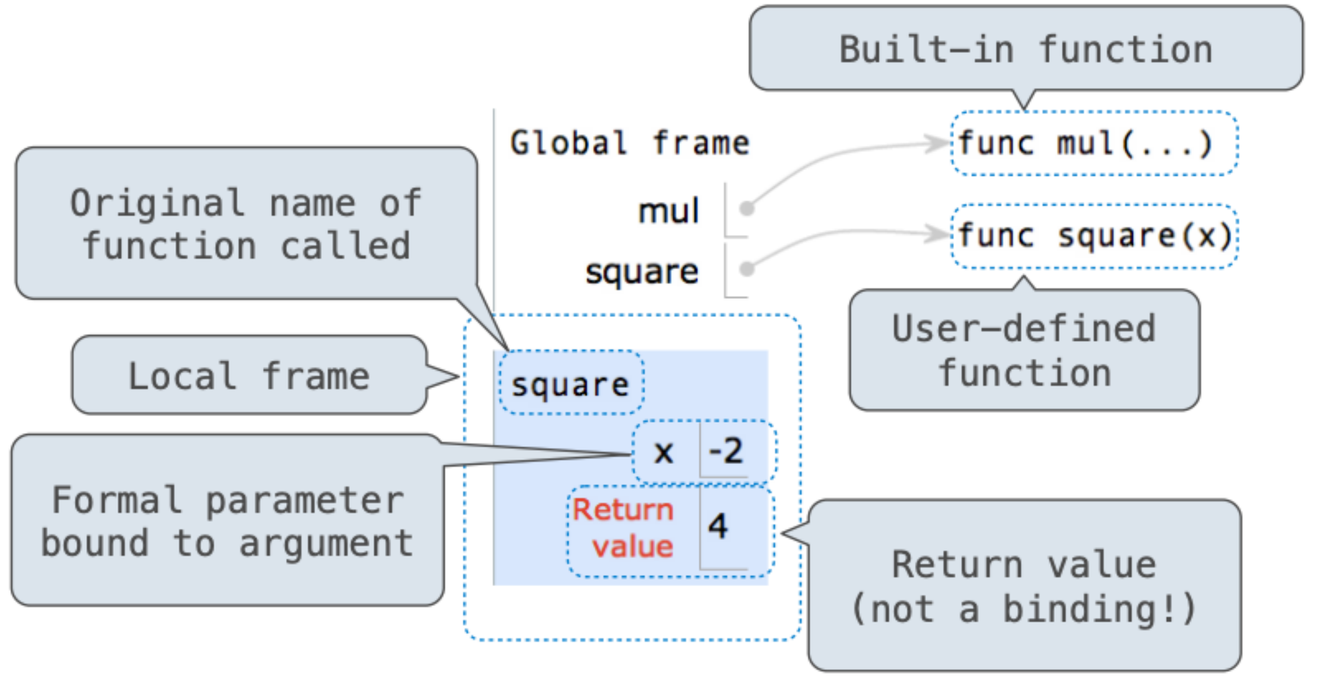
函数

Calling User-Defined Functions

Procedure for calling/applying user-defined functions (version 1):

1. Add a local frame, forming a new environment
2. Bind the function's formal parameters to its arguments in that frame
3. Execute the body of the function in that new environment

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```



函数

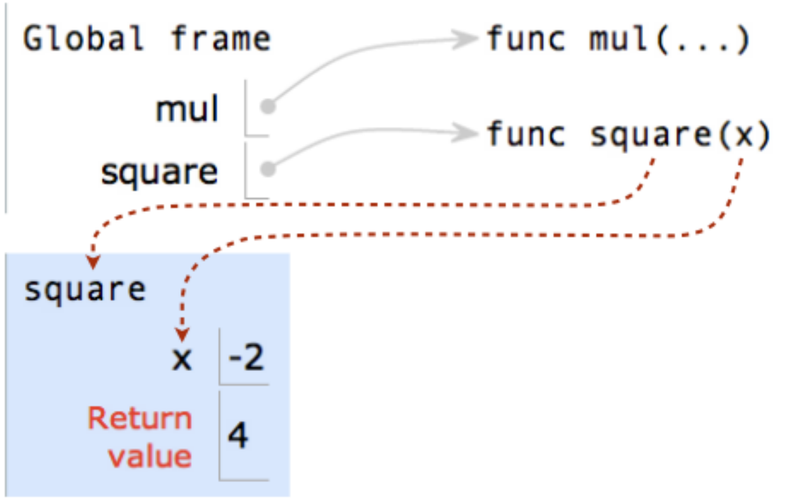
Calling User-Defined Functions

Procedure for calling/applying user-defined functions (version 1):

1. Add a local frame, forming a new environment
2. Bind the function's formal parameters to its arguments in that frame
3. Execute the body of the function in that new environment

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```

A function's signature has all the information needed to create a local frame



http://pythontutor.com/composingprograms.html#code=from%20operator%20import%20mul%0Adef%20square%28x%29%3A%0A%20%20%20%20return%20mul%28x,%20x%29%0Asquare%28-2%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

函数

Looking Up Names In Environments

Every expression is evaluated in the context of an environment.

So far, the current environment is either:

- The global frame alone, or
- A local frame, followed by the global frame.

Most important two things I'll say all day:

An environment is a sequence of frames.

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

E.g., to look up some name in the body of the square function:

- Look for that name in the local frame.
- If not found, look for it in the global frame.
  (Built-in names like "max" are in the global frame too,
  but we don't draw them in environment diagrams.)

Control: if / else / elif

demo

```
a = int(input())

if a > 100:
    print(f"100 is too small because I'm {a}")
elif a % 2 == 0:
    print(f"{a} says: I'm small, but I'm even!")
else:
    print("So I'm an odd number and I'm smaller than 100")
```

Control: while / for

demo

```
i, total = 0, 0

while i < 3:
    i = i + 1
    total = tatal + i
```

demo

```
total = 0

for i in range(0, 4):
    total += i

for i in [4, 3, 2, 5, 9]:
    print(i)
```

Example: 数的唯一分解

- 给你一个数，将其做唯一分解 $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$

- 例如：
- 120 = 2^3 * 3 * 5
- 输出：
- 2 3
- 3 1
- 5 1

Higher-Order Function

demo

```
def make_adder(k):
    def adder(n):
        return n + k
    return adder

func = make_adder(3)
func(4)
```

Lambda Function

demo

```
def make_adder(k):
    return lambda n: n + k

func = make_adder(3)
func(4)
```