MF20006: Introduction to Computer Science

Lecture 4: Programming Languages

Hui Xu

xuh@fudan.edu.cn

Outline

1. Overview of Programming Languages

Why (High-level) Programming Languages?

- **Assembly code lacks high-level constructs.**
- **Writing assembly code directly is painful.**
  - Low efficiency: far more lines of code, leading to slower development.
  - Hard to scale: readability and maintainability collapse as projects grow.
  - Error-prone: hard to maintain correctness at the instruction level.
  - Poor portability: code tied to a specific architecture/CPU.

Comparison between Source and Assembly Code

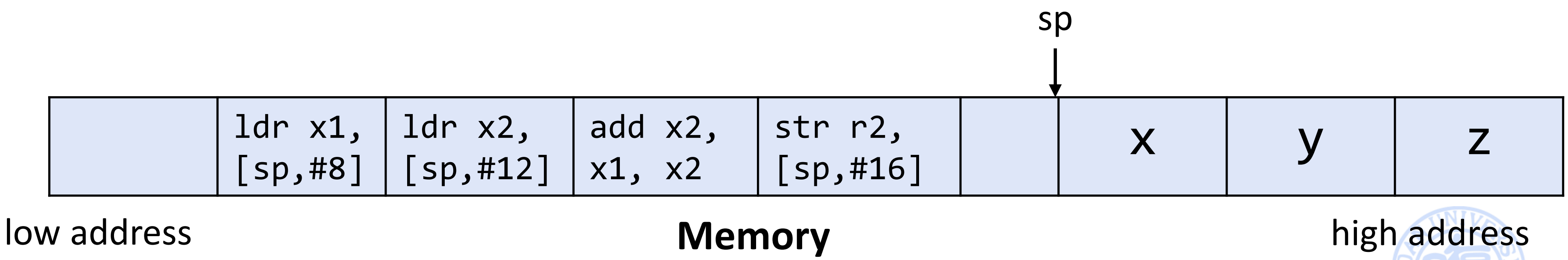
- In computer programs, variables are placed on memory.
- Assembly code has no variable names, but only memory addresses.
- Access variables via a stack pointer (sp) + offset of the variable.

```
z = x + y
```

compile

```
ldr x1, [sp]
ldr x2, [sp, #4]
add x2, x1, x2
str x2, [sp, #8]
```

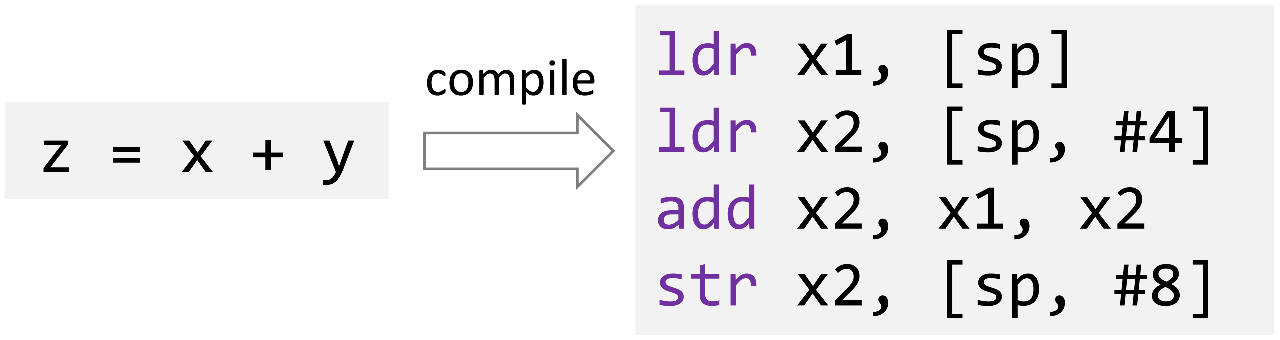
Source Code **Assembly Code**



Memory

Features of Compilers

- ❑Understanding: Parse and interpret the source code.
- ❑Translation: Generate equivalent assembly code.
- ❑Optimization: Improve the speed and size of the code.



```
z = x + y
```

```
ldr x1, [sp]
ldr x2, [sp, #4]
add x2, x1, x2
str x2, [sp, #8]
```

Source Code **Assembly Code**

Programming with Natural Language?

- ❑How to interpret programs specified in natural languages?
- ❑Large language models make natural-language programming increasingly promising.

Prompt: Write a python program to calculate the value of y based on the input x. The rule is: if x equals 1 or 2, y equals x; otherwise, y equals x times the result of y for x - 2.

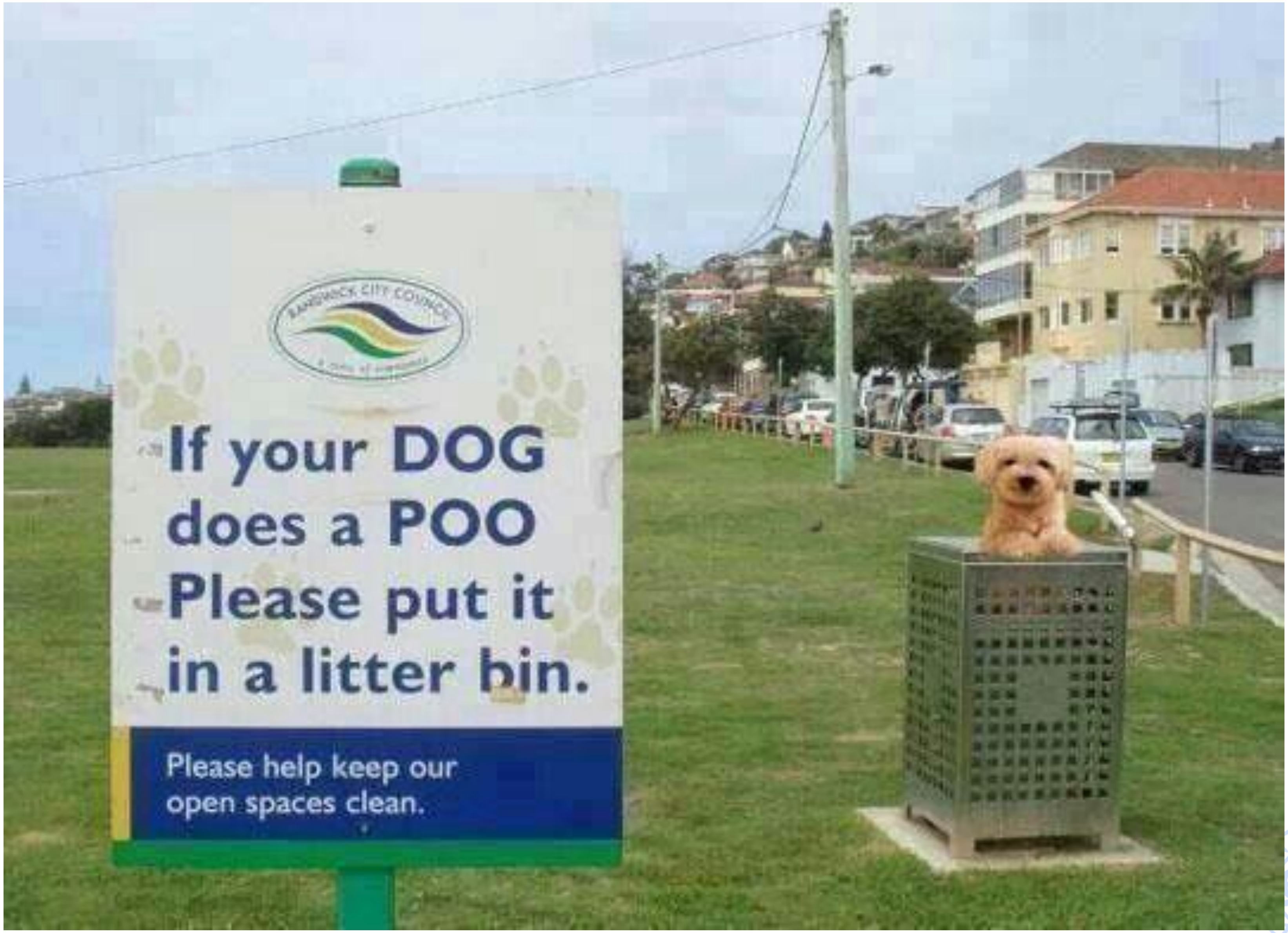
Code generated by LLM:

```python
def foo(x):
    if x == 1 or x == 2:
        return x
    else:
        return x * foo(x - 2)
```

Programming with Natural Language

- **Challenge: Natural language is often vague or underspecified.**
  - Vague or Ambiguous: The same phrase can mean different things.
  - Underspecified: The description isn’t self-contained; key details are missing.
    - Example 1: “Write a Fibonacci function.” What is the definition of such a function?
    - Example 2: “Write a Crawler that can XXX.” What content? What data format?
- **Pseudocode is more accurate than natural language.**

Challenge: Ambiguity of High-level Languages


RANDWICK CITY COUNCIL
If your DOG does a POO Please put it in a litter bin.
Please help keep our open spaces clean.

Challenge: Ambiguity of High-level Languages

A programmer's wife asks him to go to the grocery. She says "Get a gallon of milk. If they have eggs, get 12."

The programmer returns with 12 gallons of milk.

Ambiguity of Code: Example

- Not only natural language but also code written in programming languages can be ambiguous.
- In the following example, what is the result when a = 1, b = 0?

```
result = 0
if (a>0)
    if (b>0)
        result = 1;
    else
        result = -1;
```

C/C++ Code

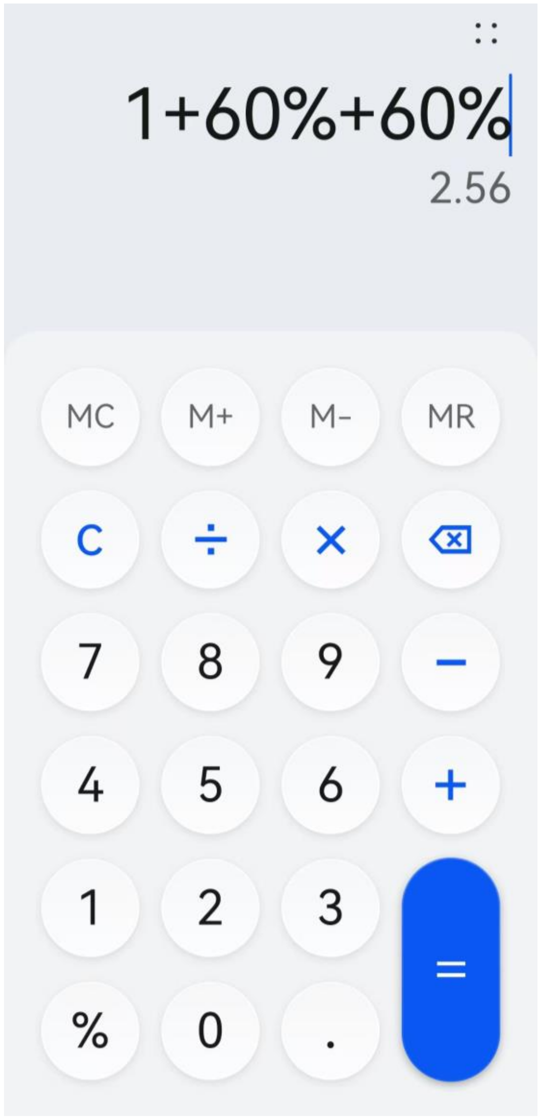
```
result = 0
if (a>0) {
    if (b>0) {
        result = 1;
    } else {
        result = -1;
    }
}
```

Interpretation 1

```
result = 0
if (a>0) {
    if (b>0) {
        result = 1;
    }
} else {
    result = -1;
}
```

Interpretation 2

Underspecified Expressions

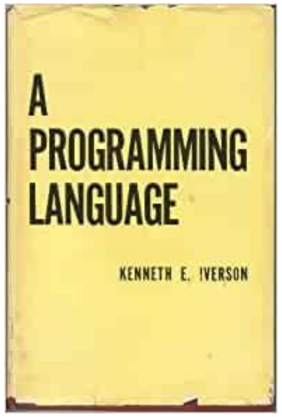

1+60%+60%

2.56

Concept of Programming Language

"Applied mathematics is largely concerned with the design and analysis of explicit procedures for calculating the exact or approximate values of various functions. Such explicit procedures are called algorithms or programs. Because an effective notation for the description of programs exhibits considerable syntactic structure, it is called a programming language."

- Kenneth Iverson



From Mathematics to Program Languages

$$f(x) = \begin{cases} 1, & x = 1 \\ x * (f(x) - 1), & x > 1 \end{cases}$$

Math Formula

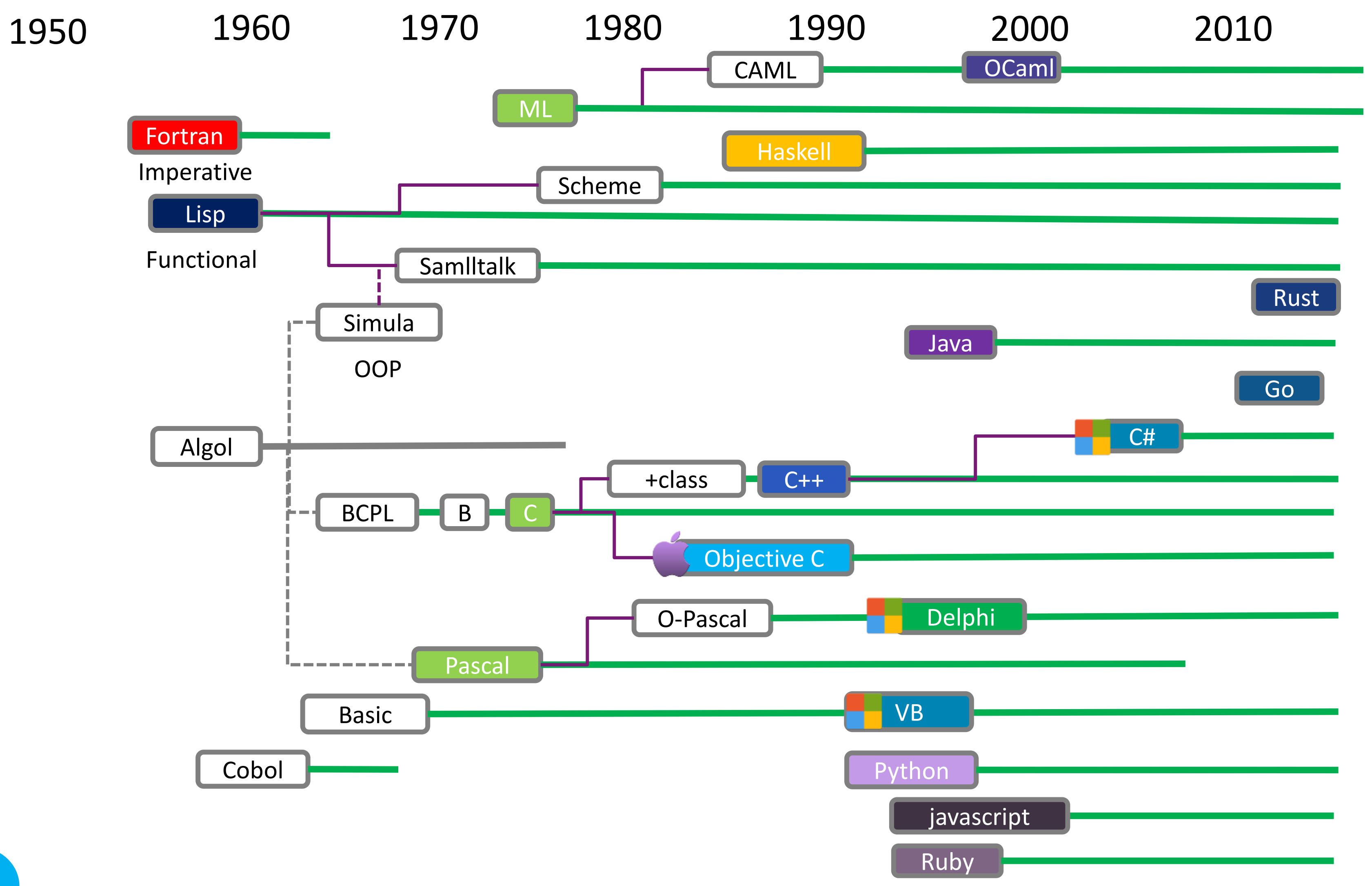
```lisp
(defun fac (n)
    (if (= n 0)
        1
        (* n (fac (- n 1)))))
```

Lisp Code

```python
def fac(n):
    if n == 0:
        return 1
    else:
        return n * fac(n - 1)
```

Python Code

Genealogy of Programming Languages



Do We Need a New Programming Language?

- **Reinventing a language can be useful for new scenarios.**
  - Java: designed for portable execution across platforms.
  - Python: excels at data analysis and machine learning.
  - Rust: focuses on system software security and safe concurrency.

Consider this image before saying "don't reinventing the wheel?"

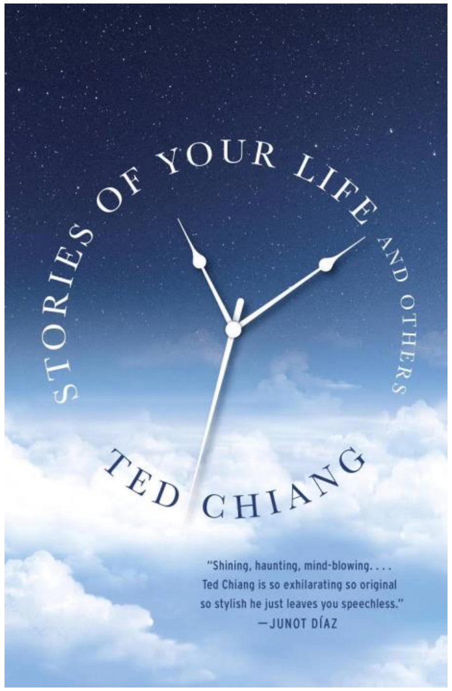
TIOBE Index

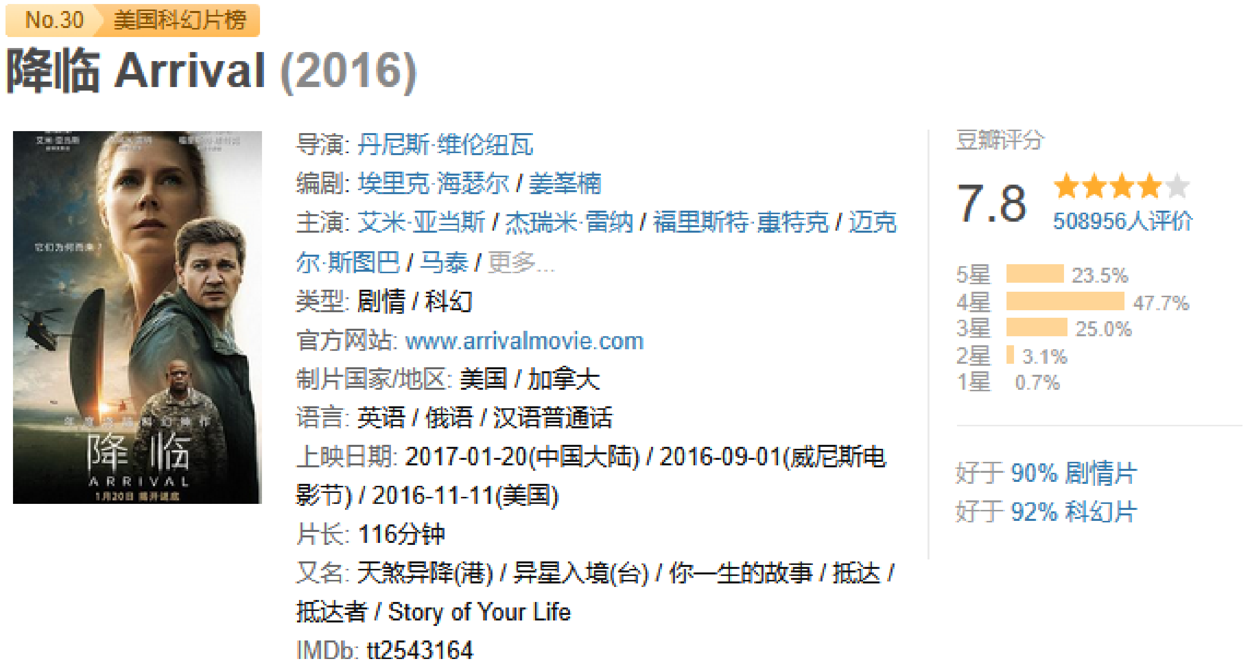
Aug 2024	Aug 2023	Programming Language	Ratings	Change
1	**1**	**Python**	**18.04%**	**+4.71%**
2	3	C++	10.04%	-0.59%
3	2	C	9.17%	-2.24%
4	4	Java	9.16%	-1.16%
5	5	C#	6.39%	-0.65%
6	6	JavaScript	3.91%	+0.62%
7	8	SQL	2.21%	+0.68%
8	7	Visual Basic	2.18%	-0.45%
9	12	Go	2.03%	+0.87%
10	14	Fortran	1.79%	+0.75%
11	13	MATLAB	1.72%	+0.67%
12	23	Delphi/Object Pascal	1.63%	+0.83%
13	10	PHP	1.46%	+0.19%
14	**19**	**Rust**	**1.28%**	**+0.39%**
15	17	Ruby	1.28%	+0.37%
16	18	Swift	1.28%	+0.37%
17	**9**	**Assembly language**	**1.21%**	**-0.13%**
18	27	Kotlin	1.13%	+0.44%
19	16	R	1.11%	+0.19%
20	11	Scratch	1.09%	-0.13%

https://www.tiobe.com/tiobe-index/

Sapir-Whorf Hypothesis: Language Influences Thought

❑The structure and vocabulary of the language you speak can affect how you perceive, categorize, and reason about the world.


STORIES OF YOUR LIFE AND OTHERS
TED CHIANG
"Shining, haunting, mind-blowing. . . . Ted Chiang is so exhilarating so original so stylish he just leaves you speechless."
—JUNOT DÍAZ


No.30 美国科幻片榜

降临 Arrival (2016)

导演: 丹尼斯·维伦纽瓦
编剧: 埃里克·海瑟尔 / 姜峯楠
主演: 艾米·亚当斯 / 杰瑞米·雷纳 / 福里斯特·惠特克 / 迈克尔·斯图巴 / 马泰 / 更多...
类型: 剧情 / 科幻
官方网站: www.arrivalmovie.com
制片国家/地区: 美国 / 加拿大
语言: 英语 / 俄语 / 汉语普通话
上映日期: 2017-01-20(中国大陆) / 2016-09-01(威尼斯电影节) / 2016-11-11(美国)
片长: 116分钟
又名: 天煞异降(港) / 异星入境(台) / 你一生的故事 / 抵达 / 抵达者 / Story of Your Life
IMDb: tt2543164

豆瓣评分

7.8 ★★★★☆ 508956人评价

5星	23.5%
4星	47.7%
3星	25.0%
2星	3.1%
1星	0.7%

好于 90% 剧情片
好于 92% 科幻片

2. Python

Python

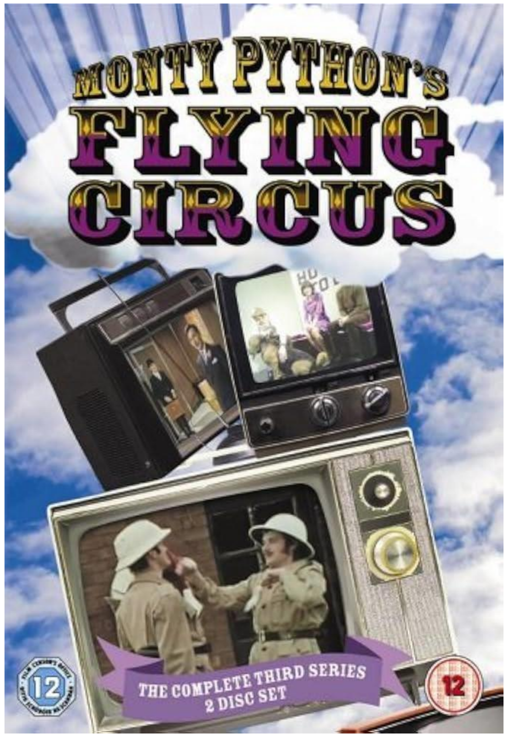
❑Invented by a Dutch programmer, Guido van Rossum, in 1991.

❑A high-level programming language.

❑Used in AI, data science, automation, *etc*.

❑Famous for its clean, simple syntax.

Origin of the name:
A British comedy



Variables

- **A variable is like a label you stick on a box so you can store something and find it later.**
- **Rules of variable naming**
  - Variable names must start with a letter or underscore, letters, numbers.
  - Avoid Python keywords like 'def', 'for', 'if’.
  - Example of valid names: abc123, _abc, ifabc.
  - Example of invalid names: 123abc.
- **Case-sensitive: *e.g.,* 'abc123' and ‘Abc123' are different.**

Naming Conventions: Snake Case vs Camel Case

- **Snake case: All words are lowercase and separated with underscores.**
  - For example: fudan_university, total_score
  - Used in python: variable names, function names
- **Camel Case: Each word starts with a capital letter.**
  - For example: FudanUniversity, TotalScore
  - Used in python: class names.

Types of Variables

- **Each variable is typed, such as integer, float, string, *etc*.**
- **By default, the type is implicit and automatically inferred.**
  - Since Python 3.5, you can annotate your functions and variables with types.

```
age = 25                  # integer
height = 1.75             # float
name: str = "Alice"        # string
age = "25"                # string
is_student = True         # boolean
```

Type Checking

- You cannot apply an operation to an incompatible type.
- Because operations do not make sense, Python Raises TypeError.
- Type coercion (implicit conversion): When two different numeric types are combined in an expression, Python automatically converts the narrower type to the wider type.
  - The operation can be done safely.

```
x = 5 + 3           # integer 8
x = "5" + "3"       # integer "53"
x = 5 + "3"         # TypeError
x = 0.5 + 3         # Type coercion: float 3.5
x = 5 + True        # Type coercion: integer 6
```

Define a Variable

❑Variables must be defined (assigned) before used.

Define a variable :

```
score = 10
score = a           # illegal if a is undefined
```

Update the value of a variable

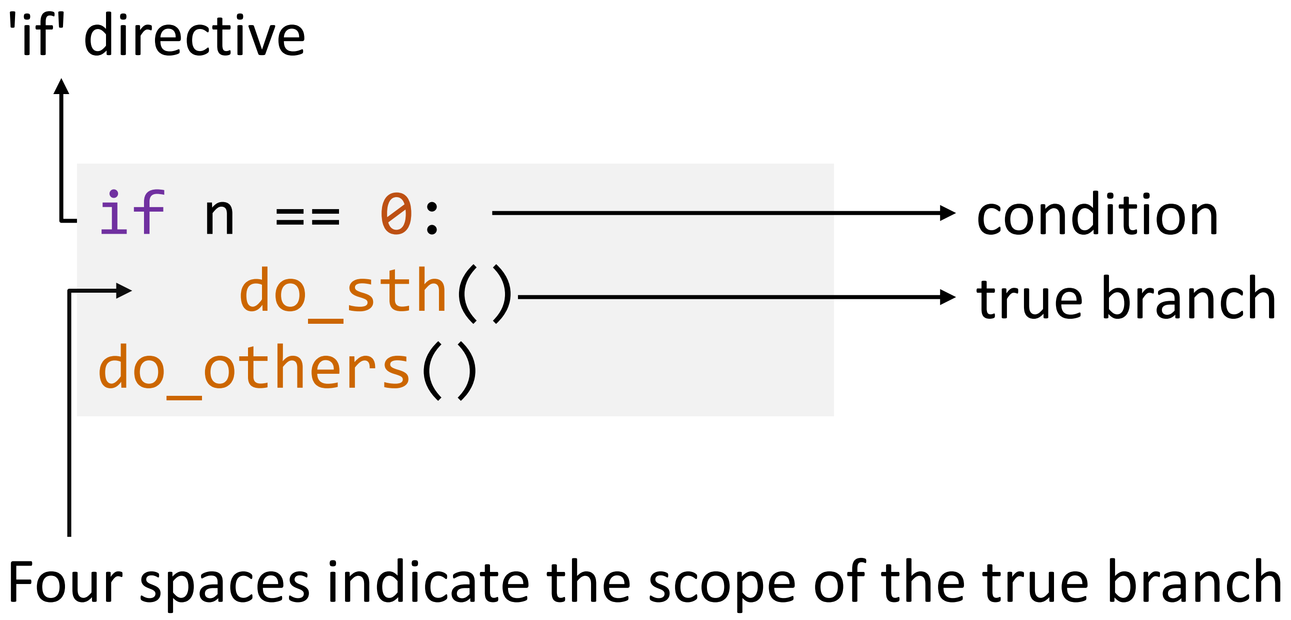
```
score = score + 5   # score must be defined before
score += 5          # score = score + 5
```

Define multiple variables in one statement:

```
x = y = z = 0        # x=0; y=0; z=0
a, b, c = 1, 2, 3    # a=1; b=2; c=3;
```

if Statement

- ❑Condition: A logical expression or Boolean expression
- ❑Code block: Executed if the condition evaluates to true.



```
if n == 0:
    do_sth()
do_others()
```

if-else Statement

- ❑ **An if-else statement consists of:**
  - ➢ An if condition and a code block to be executed if the condition is true.
  - ➢ An else block to be executed if the condition is false.

```
if n == 0:
    do_sth()
else:
    do_else()
```

Question: Is the Following Code Valid?

```
# program start
if random() % 2 == 0:
    b = 1;
b = b + 1;
```

- **Invalid in other programming languages like C/C++, Rust.**
  - Static type checking finds b might be undefined before used.
- **Could be valid/executable in python if the condition is true.**
  - Dynamic checking only checks the current execution path.

Question: Is the Following Code Valid?

```
# program start
if random() % 2 == 0:
    b = 1;
else:
    b = 0;
b = b + 1;
```

- Still invalid in C/C++, Rust.
- Always valid in python because b is defined before use in both paths.

if-elif-else Statement

- An if-elif-else statement consists of:
  - An if condition and the block of code to be executed if the condition is true.
  - One or several elif condition(s) to check and corresponding code blocks.
  - An else block (optional) to be executed if all previous conditions are false.
- Not recommended, use math-case instead.

```
if n == 0:
    do_sth()
elif n == 1:
    do_elif()
else:
    do_else()
```

match-case (Introduced in Python 3.10)

❑It matches a value or multiple values against different patterns.

❑Makes code more readable for multiple discrete cases.

```
match a:
    case 0:
        do_0()
    case 1:
        do_1()
    case _:
        do_else()
```

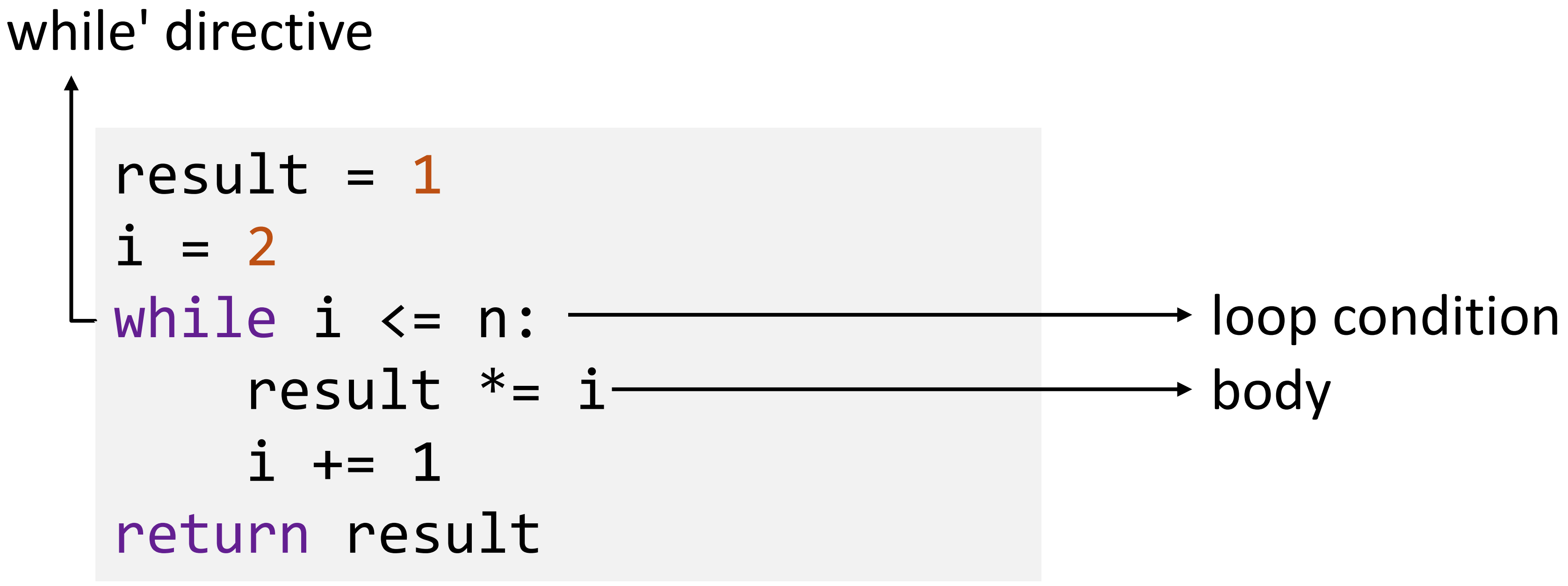
```
match (a, b):
    case (0, 0):
        do_00()
    case (1, _):
        do_1()
    case (_, _):
        do_else()
```

Advanced Conditions of match-case

```
def gcd(a, b):
    match (a, b):
        case (_, 0):
            return a
        case (0, _):
            return b
        case (a, b) if a == b:  →  a = b
            return a
        case (a, b) if a > b:   →  a > b
            return gcd(a - b, b)
        case (a, b):            →  a < b
            return gcd(a, b - a)
```

while Statement

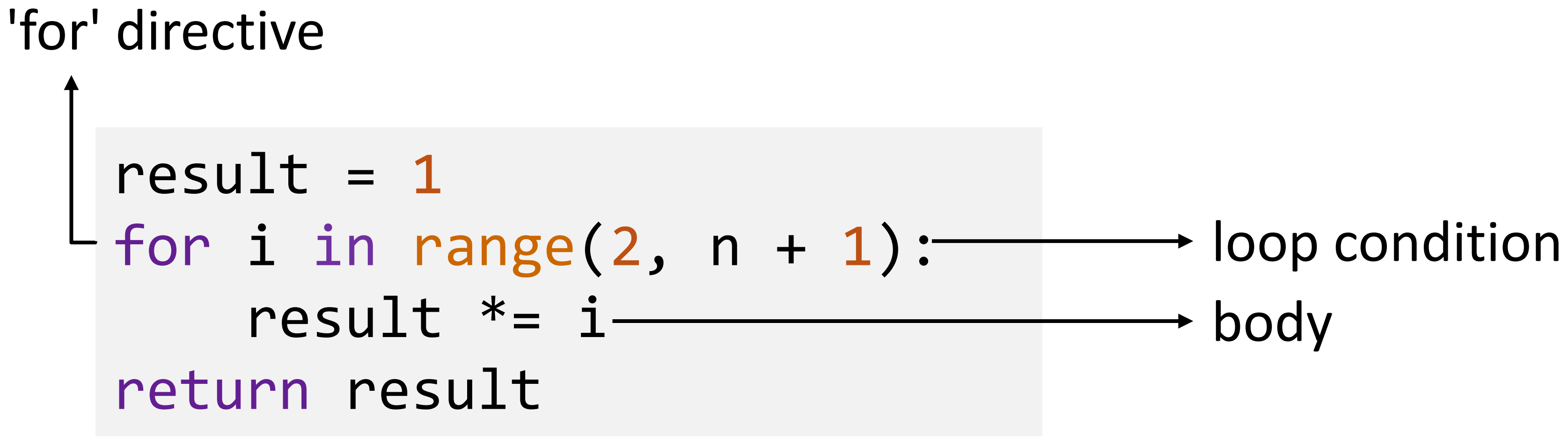
- ❑Repeat a block of code as long as a given condition is true.
- ❑It is often used when the number of iterations is not known.



```
result = 1
i = 2
while i <= n:
    result *= i
    i += 1
return result
```

for Statement

- ❑Repeat a block of code for each item in a sequence.
- ❑You don’t manually manage loop counters.
- ❑Python iterates automatically.



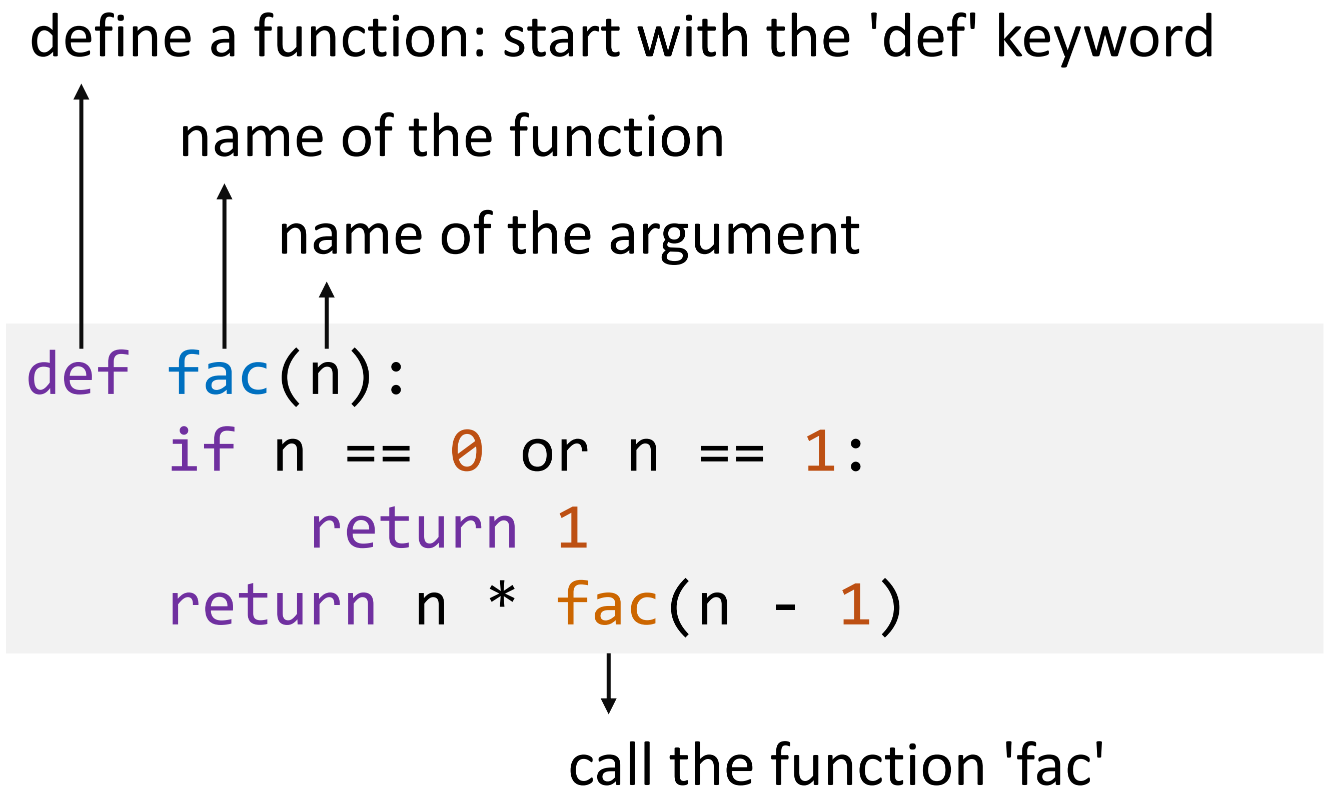
```python
result = 1
for i in range(2, n + 1):
    result *= i
return result
```

Function

- An abstraction that maps a set of input values to a set of output.
- A named block of code that performs a specific task.
    - Take inputs (parameters).
    - Optionally produce an output (return value).
- Function acts as a reusable unit of work.
    - Define once, call multiple times.

Function: Syntax

- ❑A function generally has a name and a code body.
- ❑Arguments and return values are optional.



```python
def fac(n):
    if n == 0 or n == 1:
        return 1
    return n * fac(n - 1)
```

Iteration vs Recursion

- **Iteration:** Repeating a block of code using loops.
    - Doing something repeatedly until a condition is met.
- **Recursion:** A function calls itself to solve a problem.
    - Solving a problem by reducing it to a smaller version of itself.

```python
def fac(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
```

```python
def fac(n):
    if n == 0 or n == 1:
        return 1
    return n * fac(n - 1)
```

More Types in Python

- **Scalar type: A scalar type stores a single value.**
  - Boolean, integer, float, string (immutable in Python)
- **Compound type:**
  - Tuple: An ordered, immutable collection of items
  - List: A built-in mutable sequence to store an ordered collection of items
  - Class

Tuple

- ❑Ordered: The order of elements is preserved.
- ❑Immutable: once a tuple is created, you cannot modify, add, or remove its elements.
- ❑Tuples can contain heterogeneous data (different types), including numbers, strings, lists, or even other tuples.

```
t1 = (1, 2, 3)                # Tuple with elements
t2 = (1, "hello", 3.14)       # Tuple with mixed types
t3 = (1, (2, 3), [4, 5])      # Nested tuple
print(t1[0])                  # 1, access the first element of t1
print(t1[0:2])                # 1,2, access the slice of t1
for item in t1:               # access all elements iteratively
    print(item)
```

List

- ❑Ordered: The order of elements is preserved.
- ❑Mutable: You can add, remove, or change elements.
- ❑Allows duplicates: Same value can appear multiple times.
- ❑Holds different types: Numbers, strings, objects, etc.

```
a = []                      # Empty list
b = [1, 2, 3]               # List with elements
c = [1, "hello", 3.14]      # List with mixed types
print(b[0])                 # access the first element of b
b.append(4)                 # Add at the end
b.insert(2, 100)            # Insert 100 at index 1
b.remove(2)                 # Remove first occurrence of 2
for n in nums:              # Simple for loop
    print(n)
```

Dictionary

- ❑A built-in data structure that stores data as key-value pairs.
- ❑Dictionary is very efficient for lookups.
- ❑You can use a key to quickly retrieve or modify the value.

```python
students = {
    "Alice": 20,
    "Bob": 21,
    "Charlie": 19
}
print(students["Bob"])  # Output: 21
```

Class

- **A programming abstraction that defines a new type of object.**
  - specifying the attributes (data)
  - and methods (behavior) that its instances will have
- **Similar to function that acts as a reusable unit of work:**
  - define it once, create multiple instances that share the same structure and behavior but can hold different data

Define a Class

define a class: start with the 'class' keyword

name of the class in Camel Case

constructor of the class

```python
class Ellipse:
    def __init__(self, f1, f2, radius_sum):
        self.f1 = f1
        self.f2 = f2
        self.radius_sum = radius_sum
        dist_foci = math.dist(f1, f2)
        if radius_sum <= dist_foci:
            raise ValueError("Not an ellipse.")
    def contains_point(self, x, y):
        ...
```

Use the Class

Creating an Ellipse via the construction: __init__

```
e = Ellipse((0, 0), (4, 0), 10)
```

Using the contains_point method

```
print(e.contains_point(3, 0))
```

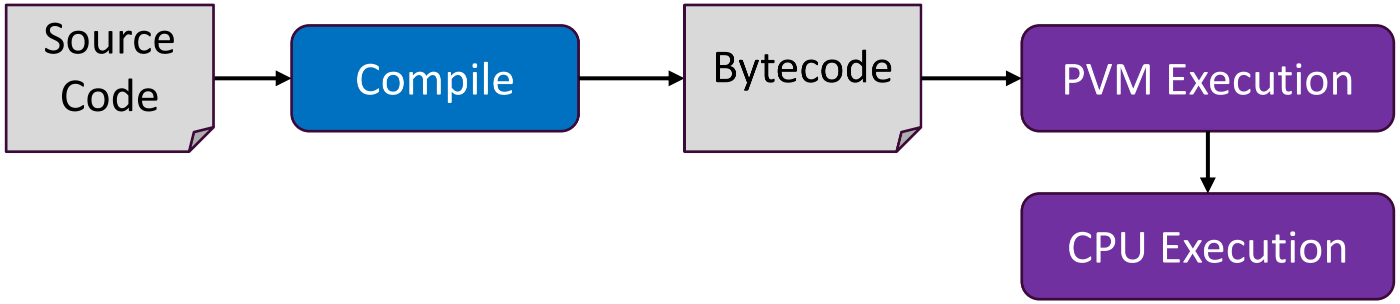
Indentation

- ❑Python uses indentation to define code blocks and scope.
  - ➢Four spaces or Tab?
- ❑Compared to {}, which is commonly used in other languages.

HELLO.
HOWDY.
TABS OR SPACES?
I'M MORE LIKE A SPACE GUY, HEHE.
IT WAS NICE TO MEET YOU, BYE BYE!
CRYSTALLIZE.COM

How Python Code is Executed

- ❑Source code: .py files
- ❑A compiler translates source code into bytecode
  - ➢.pyc files in the __pycache__ folder
- ❑The bytecode is then executed by the Python Virtual Machine



3. Practice: Vibe Coding with Python

Vibe Coding

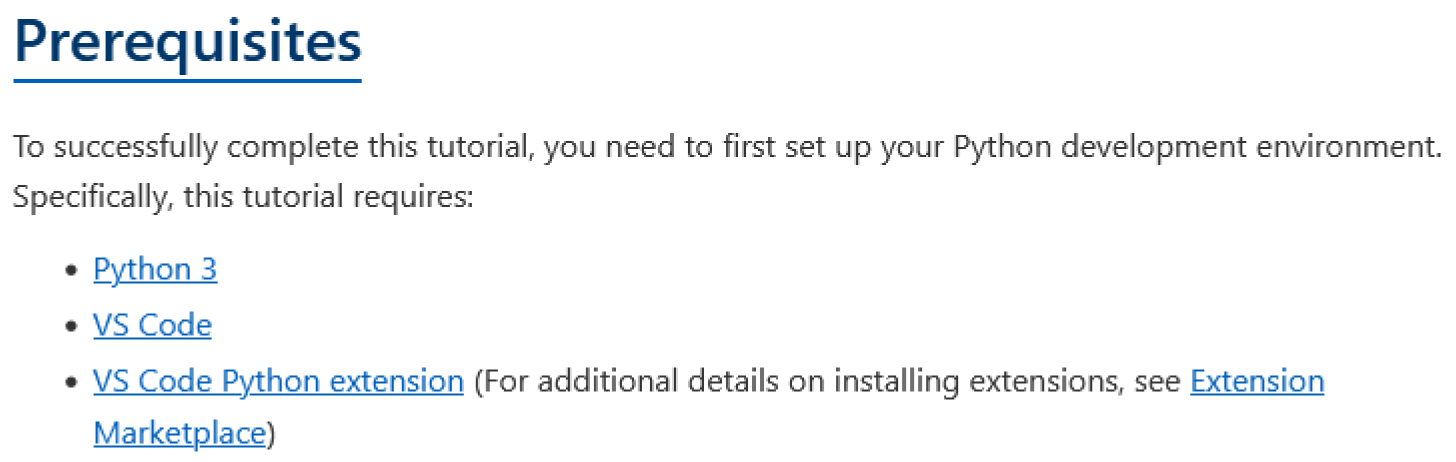
1) The developer describes a project or task to a large language model (LLM).
2) LLM generates code based on the prompt.
3) The developer does not review or edit the code, but solely uses tools and execution results to evaluate it and asks the LLM for improvements.

“There's a new kind of coding I call "vibe coding," where you fully give in to the vibes.”
-Andrej Karapathy

Download and Install Python

❑Tutorial of installing Python with VS Code as the IDE

➢https://code.visualstudio.com/docs/python/python-tutorial



Prerequisites

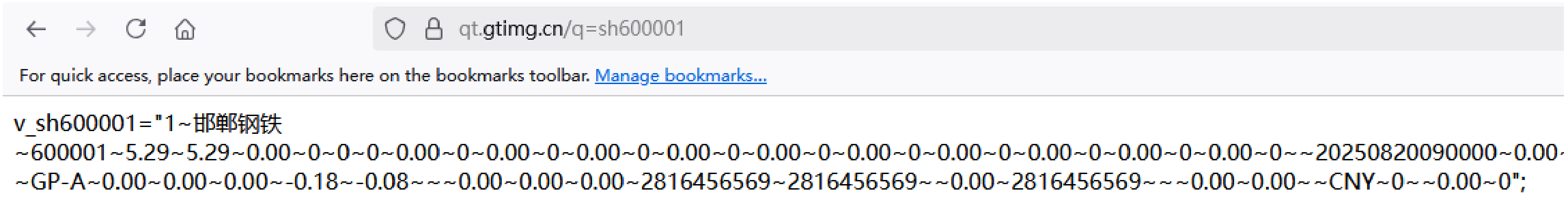
To successfully complete this tutorial, you need to first set up your Python development environment. Specifically, this tutorial requires:

- Python 3
- VS Code
- VS Code Python extension (For additional details on installing extensions, see Extension Marketplace)

Sample Task: Develop a Crawler

- ❑For example, crawling stock information via the following URL.
- ❑Be careful about the legal risks of crawling.

```
http://qt.gtimg.cn/q=sh600001
```

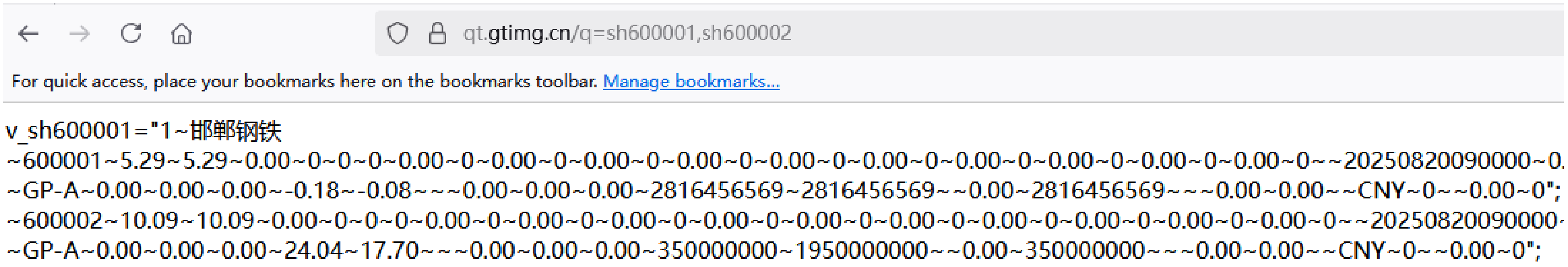


qt.gtimg.cn/q=sh600001

For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...

```
v_sh600001="1~邯郸钢铁
~600001~5.29~5.29~0.00~0~0~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~~20250820090000~0.00
~GP-A~0.00~0.00~0.00~-0.18~-0.08~~~0.00~0.00~0.00~2816456569~2816456569~~0.00~2816456569~~~0.00~0.00~~CNY~0~~0.00~0";
```

```
https://qt.gtimg.cn/q=sh600001,sh600002
```



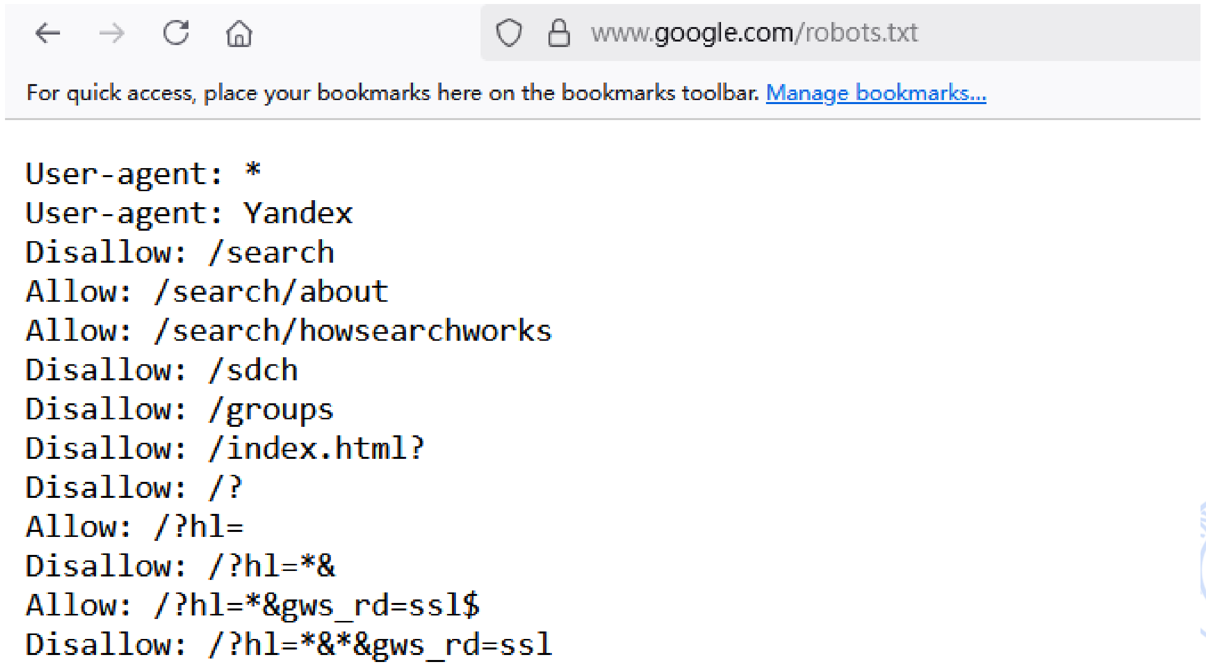
qt.gtimg.cn/q=sh600001,sh600002

For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...

```
v_sh600001="1~邯郸钢铁
~600001~5.29~5.29~0.00~0~0~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~~20250820090000~0.
~GP-A~0.00~0.00~0.00~-0.18~-0.08~~~0.00~0.00~0.00~2816456569~2816456569~~0.00~2816456569~~~0.00~0.00~~CNY~0~~0.00~0";
~600002~10.09~10.09~0.00~0~0~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~0.00~0~~20250820090000
~GP-A~0.00~0.00~0.00~24.04~17.70~~~0.00~0.00~0.00~350000000~1950000000~~0.00~350000000~~~0.00~0.00~~CNY~0~~0.00~0";
```

Robots Exclusion Protocol

- robots.txt is a text file placed at the root of a website.
- It tells web crawlers which parts of the site they are allowed or not allowed to crawl.
- A robots.txt file is made of rules. Each rule has two parts:
  - User-agent: specifies which bot(s) the rule applies to.
  - Allow / Disallow: specifies which paths are allowed or forbidden.


www.google.com/robots.txt
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...

```
User-agent: *
User-agent: Yandex
Disallow: /search
Allow: /search/about
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
Allow: /?hl=
Disallow: /?hl=*&
Allow: /?hl=*&gws_rd=ssl$
Disallow: /?hl=*&*&gws_rd=ssl
```