



Dify 节点逻辑 + 数据类型

本教程参考 [🚀 Dify workflows架构全解析 | 节点逻辑 + 数据类型深度剖析！#Dify #Dify节点 #AI自动化 #Dify教程 #Workflow #Chatf 哔哩哔哩 bilibili](#)，总结了**工作流（Workflow）基本架构**以及其中涉及的**各类数据类型**，以便帮助对编程不太熟悉的小白快速理解并上手

工作流

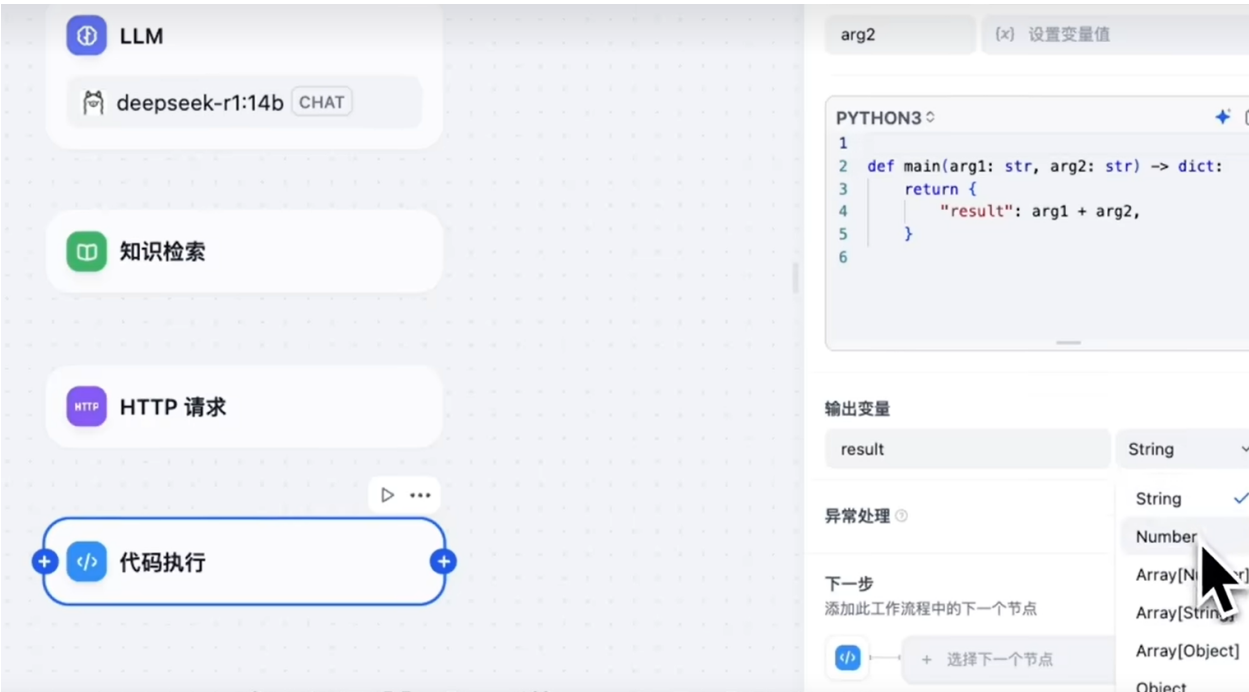
工作流 是由一系列 Node 组成的自动执行流程。每个Node会执行一个特定的功能，比如接收输入、处理数据、调用模型或服务，并输出结果供下一个任务使用。理解这些组成元素和它们之间流动的数据类型，是成功搭建一个工作流的关键。

工作流基本架构

模块	目的	核心任务	典型节点
数据预处理	接受输入并标准化输出	输入接受，数据清洗，格式转换，上下文构建	[开始]节点, [文档提取器]节点, [代码执行]节点

模块	目的	核心任务	典型节点
AI生成模块	执行核心AI逻辑和业务处理	利用AI能力处理信息，这是工作流的“大脑”	[LLM]节点, [知识检索]节点, [Agent]节点
数据输出整形模块	优化输出并呈现最终结果	数据整形，结果格式化，数据聚合	[回复]节点, [模板转换]节点, [代码执行]节点

各个节点中的输入和输出都会设计到不同的数据类型



数据类型

- Number

python

Number (数字) 类型示例

整数示例

age = 25 # 年龄, 整数类型

浮点数示例

weight = 65.5 # 体重, 浮点数类型

负数示例

temperature = -10 # 温度, 整数类型 (负数)

数学运算示例

average_temp = (23 + 28 + 25 + 22 + 20) / 5 # 计算一周平均温度

celsius_to_fahrenheit = temperature * 9/5 + 32 # 摄氏度转华氏度

类型转换示例

temperature_str = str(temperature) # 将数字转换为字符串

- String

python

```
# 字符串是由字符组成的序列
name = "张三"
greeting = 'Hello, World!'
multiline = """这是
多行
字符串"""

# 字符串操作
print(name[0]) # 输出: 张
print(greeting + "!") # 输出: Hello, World!!
print(len(greeting)) # 输出: 13
print(greeting.upper()) # 输出: HELLO, WORLD!
```

- Object

python

```
# 对象 (字典)
person = {
    "name": "张三",
    "age": 30,
    "is_student": False,
    "hobbies": ["读书", "旅游", "编程"],
    "address": {
        "city": "北京",
        "district": "海淀区",
        "street": "中关村大街"
    }
}

# 访问对象属性
print(person["name"]) # 输出: 张三
print(person["hobbies"][1]) # 输出: 旅游
print(person["address"]["city"]) # 输出: 北京
```

- Array[Number] 数字列表

python

```
# 数字数组
scores = [98, 87, 95, 63, 75]
temperatures = [36.5, 37.2, 36.9, 37.5]

# 访问元素
print(scores[0]) # 输出: 98
print(scores[-1]) # 输出: 75 (最后一个元素)

# 数组操作
scores.append(82) # 添加元素: [98, 87, 95, 63, 75, 82]
scores.sort() # 排序: [63, 75, 82, 87, 95, 98]
print(len(scores)) # 输出数组长度: 6
print(sum(scores)) # 计算总和
print(max(scores)) # 找出最大值
```



- Array[String] 字符串列表

```
python Copy

# 字符串数组
names = ["李雷", "韩梅梅", "张伟", "吉姆"]
# print(names) 输出: ["李雷", "韩梅梅", "张伟", "吉姆"]

fruits = ["苹果", "香蕉"]
# 访问和修改
print(names[1]) # 输出: 韩梅梅 (注意这里应该是"韩梅梅"而不是"李雷")

names[2] = "大张伟" # 修改元素
# print(names) 输出: ["李雷", "韩梅梅", "大张伟", "吉姆"]

names[-1] = "朱莉叶" # 修改元素
# print(names) 输出: ["李雷", "韩梅梅", "大张伟", "朱莉叶"]

# 数组操作
names.append("王老五") # 添加元素
# print(names) 输出: ["李雷", "韩梅梅", "大张伟", "朱莉叶", "王老五"]

combined = names + fruits # 合并数组
# print(combined) 输出: ["李雷", "韩梅梅", "大张伟", "朱莉叶", "王老五", "苹果", "香蕉"]
# (注意: 此操作不改变names本身)

names.sort() # 按字母顺序排序
# print(names) 输出: ["大张伟", "李雷", "朱莉叶", "王老五", "韩梅梅"]
# (注意: 在中文环境下按照汉字拼音或编码排序)

print("王老五" in names) # 是否存在, 输出: True
```

- Array[Object] 对象列表

```
python Copy

# 对象数组（字典列表）
students = [
    {"id": 1, "name": "李明", "age": 20, "score": 95},
    {"id": 2, "name": "王芳", "age": 19, "score": 88},
    {"id": 3, "name": "张伟", "age": 21, "score": 92}
]

# 访问对象数组中的元素
print(students[0]["name"]) # 输出：李明
print(students[1]["score"]) # 输出：88

# 操作对象数组
students.append({"id": 4, "name": "刘洋", "age": 20, "score": 85}) # 添加新对象
top_student = max(students, key=lambda x: x["score"]) # 找出分数最高的学生
print(top_student["name"]) # 输出：李明
```

对于刚接触 workflow 搭建的小白，理解这些数据类型能帮助你理解在设计一个流程时，如何正确地设置节点的输入与输出，确保数据能顺利地流转下去并实现目标功能。