程序逻辑结构

程序逻辑结构

- 顺序结构
- 选择结构
- 循环结构
- 综合案例

顺序结构

选择结构

循环结构

顺序结构

- 从上往下依次执行
- 通常为赋值语句、计算语句等

赋值语句

◆ 使用 = 进行赋值（简单回顾）

赋值类型	描述	示例
基本赋值	使用等号 (=) 进行赋值。	x = 10
同一个值给多个变量	可以使用一个值来赋给多个变量。	x = y = z = 10
多重赋值	可以同时给多个变量赋多个值。	x, y, z = 1, 2, 3
使用下划线的赋值	当不关心某个值时，可用下划线 (_) “丢弃” 变量。	x, _ = 1, 2

计算语句

运算符	描述	示例
+	加法	5 + 3 = 8
-	减法	5 – 3 = 2
*	乘法	5 * 3 = 15
/	算术除法	5 / 3 = 1.6666666666666667
//	整数除法求商	5 // 3 = 1
%	求余数	5 % 3 = 2
**	幂运算	5 ** 3 = 125

计算同时赋值

运算符	示例	展开式	运算（假设a=10）	运算结果
+=	a += 1	a = a + 1	a = 10 + 1	a = 11
-=	a -= 1	a = a - 1	a = 10 - 1	a = 9
*=	a *= 2	a = a * 2	a = 10 * 2	a = 20
/=	a /= 3	a = a / 3	a = 10 / 3	a = 3.333333333
//=	a //= 3	a = a // 3	a = 10 // 3	a = 3
%=	a %= 3	a = a % 3	a = 10 % 3	a = 1
**=	a **= 2	a = a ** 2	a = 10 ** 2	a = 100

数据格式

数据类型	描述	示例	特点
int	整型，整数	5　3　-1	可正数、负数或零
float	浮点型，有小数点	3.14　-0.6 1e-3	有精度限制
str	字符串，文本数据	‘Hello, World!’ “Python 编程“ ’’’ 这也是字符串’’’	- 用引号创建 - 可字符串连接和重复 - 可用索引和切片
complex	复数，有实部虚部	2+3j　3-4j	j表示虚部

查看数据类型

- 使用type()可以查看数据类型

```
a = 1
b = 3.14
c = '123'
d = "这也是字符串"
e = '''这也可以是字符串'''

print("a的数据类型是：",type(a))
print("b的数据类型是：",type(b))
print("c的数据类型是：",type(c))
print("d的数据类型是：",type(d))
print("e的数据类型是：",type(e))
```

```
a的数据类型是： <class 'int'>
b的数据类型是： <class 'float'>
c的数据类型是： <class 'str'>
d的数据类型是： <class 'str'>
e的数据类型是： <class 'str'>
```

数据格式的转换

- 转为整数
- 转为小数
- 转为字符串

转为整数

进制	单个取值范围	示例
10进制	0-9	0, 2, -1, 3
2进制	0-1	0(十进制的0) 1(十进制的1) 10(十进制的2) 11(十进制的3)
8进制	0-7	$162 = (1 \times 8^2) + (6 \times 8^1) + (2 \times 8^0) = 114$
16进制	0-9，A-F	$64 = (6 \times 16^1) + (4 \times 16^0) = 100$

转为浮点型

- 使用float()转为小数

```
num_str = input("请输入小数:")
print("num_str = ",num_str," 格式是: ",type(num_str))
num_float = float(num_str)
print("num_float = ",num_str," 格式是: ",type(num_float))
```

```
请输入小数:12
num_str =  12  格式是:  <class 'str'>
num_float =  12  格式是:  <class 'float'>
```

转为字符串

◆ 使用str()转为字符串

```
name = "Alice"
age = 30
print("My name is %s and I'm %d years old."%(name, age))
print("My name is {} and I'm {} years old.".format(name, age))
print(f"My name is {name} and I'm {age} years old.")
```

```
My name is Alice and I'm 30 years old.
My name is Alice and I'm 30 years old.
My name is Alice and I'm 30 years old.
```

转为字符串

◆ 输出时控制精度

```
number = 12.3456
print("%.2f" % number)
print("{:.2f}".format(number))
print(f"{number:.2f}")
```

```
12.35
12.35
12.35
```

选择结构

运算符	名称	描述	示例	结果
==	等于	检查两个操作数是否相等	5 == 3	False
!=	不等于	检查两个操作数是否不相等	5 != 3	True
>	大于	检查左操作数是否大于右操作数	5 > 3	True
<	小于	检查左操作数是否小于右操作数	5 < 3	False
>=	大于等于	检查左操作数是否大于等于右操作数	5 >= 3	True
<=	小于等于	检查左操作数是否小于等于右操作数	5 <= 3	False

选择结构

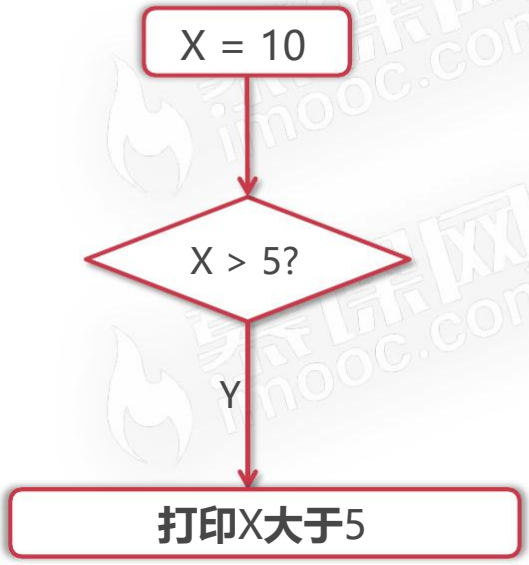
- if 语句

选择结构

◆ if 语句

```
1 x = 10
2 if x > 5:
3     print("x大于5")
```

```
1 x大于5
```

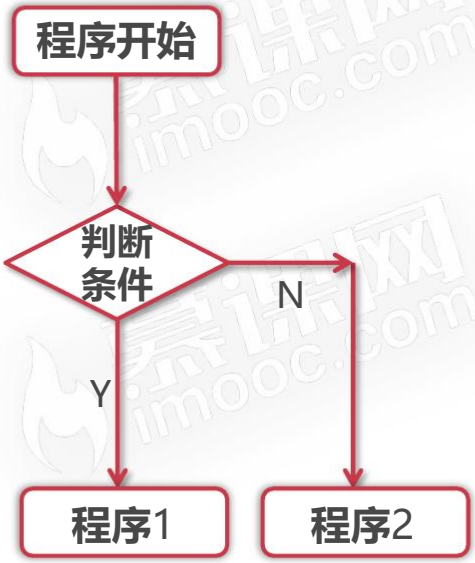


选择结构

◆ if...else 语句

```
1 x = 10
2 if x > 5:
3     print("x>5")
4 else:
5     print("x<=5")
```

```
1 x>5
```

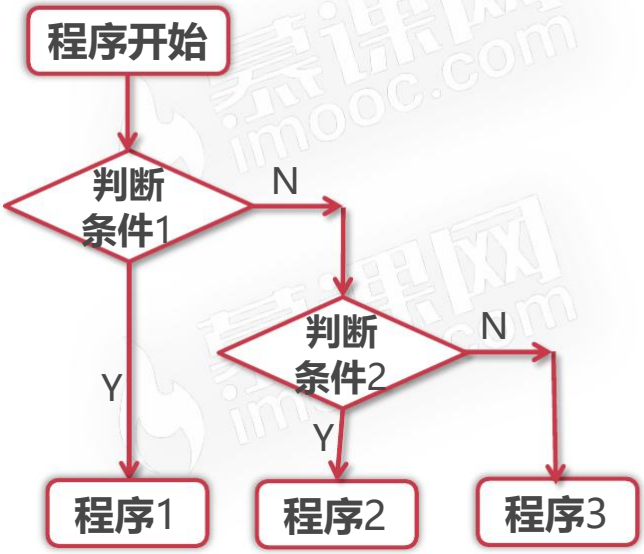


选择结构

◆ if...elif...else 语句

```
x = 5
if x > 10:
    print("x大于10")
elif x == 5:
    print("x是5")
else:
    print("x小于10，但不是 5")
```

```
x是5
```



综合案例

◆ 看懂下面代码

```
a = 10
b = 20
result = a + b
answer = int(input(f"请输入{a}+{b}的结果"))
if result == answer:
    print("回答正确！")
else:
    print("回答错误")
```

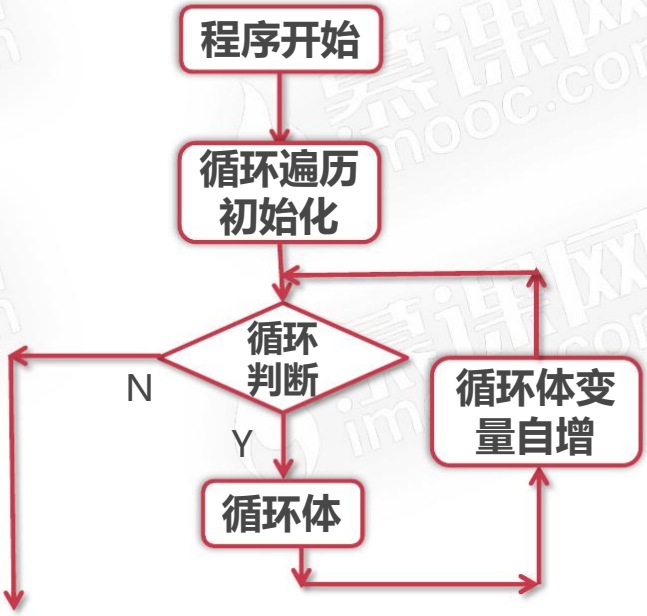
```
请输入10+20的结果：39
回答错误
```

循环结构

- 可指定遍历对象的for循环
- 不确定循环次数的while循环
- 改变循环逻辑的break和continue

循环结构

- 循环的程序流程图



for循环

◆ 可指定循环次数

```python
epoch = 5
for epoch_i in range(epoch):
    print("--------------")
    print(f"正在处理第{epoch_i}个epoch的数据")
    print(f"第{epoch_i}个数据处理完毕")
```

```
--------------
正在处理第0个epoch的数据
第0个数据处理完毕
--------------
正在处理第1个epoch的数据
第1个数据处理完毕
--------------
正在处理第2个epoch的数据
第2个数据处理完毕
--------------
正在处理第3个epoch的数据
第3个数据处理完毕
--------------
正在处理第4个epoch的数据
第4个数据处理完毕
```

for循环

◆ 可指定迭代对象

```python
optimizers = ["SGD", "Adam", "Momentum","Adagrad"]
for optimizer_i in optimizers:
    print("正在使用 ",optimizer_i," 进行优化")
```

```
正在使用  SGD  进行优化
正在使用  Adam  进行优化
正在使用  Momentum  进行优化
正在使用  Adagrad  进行优化
```

for循环

◆ 可对数据进行枚举

```
img_list = ["img_1.png", "img_2.png", "img_3.png"]
for index, img_i in enumerate(img_list):
    print(f"索引 {index} 对应的数据是 {img_i}")
```

```
索引 0 对应的数据是 img_1.png
索引 1 对应的数据是 img_2.png
索引 2 对应的数据是 img_3.png
```

while循环

◆ 当不清楚应该循环多少次时，用while

```
command = ""
while command != "end":
    command = input("请输入命令：")
    print("正在执行命令：",command)
```

```
请输入命令：forward
正在执行命令： forward
请输入命令：backword
正在执行命令： backword
请输入命令：stop
正在执行命令： stop
请输入命令：end
正在执行命令： end

Process finished with exit code 0
```

break打破循环

◆ 使用break可以停止循环，跳出整个循环

```python
# 这是一个数字列表，机器人将在这个列表中搜索数字“5”
numbers = [1, 3, 4, 2, 5, 6, 8, 7, 9]

# 这是一个标志，用来表示机器人是否找到了数字“5”
found = False

# 机器人开始搜索数字“5”
for number in numbers:
    print(f"正在查看数字{number}")
    if number == 5:
        found = True
        print(f"机器人找到了数字{number}！")
        break  # 一旦找到数字“5”，就退出循环

# 检查机器人是否找到了数字“5”
if not found:
    print("机器人没有找到数字5。")
```

```
正在查看数字1
正在查看数字3
正在查看数字4
正在查看数字2
正在查看数字5
机器人找到了数字5！
```

continue跳过当前回合

◆ continue跳过当前回合，仍在循环中

```python
# 这是一个数字列表，机器人将在这个列表中搜索不是“5”的数字
numbers = [1, 3, 4, 2, 5, 6, 8, 7, 9]

# 机器人开始搜索不是“5”的数字
for number in numbers:
    print(f"正在查看数字{number}")
    if number == 5:
        continue  # 如果数字是“5”，跳过当前迭代，继续下一次循环
    print(f"机器人找到了数字{number}！")
```

```
正在查看数字1
机器人找到了数字1！
正在查看数字3
机器人找到了数字3！
正在查看数字4
机器人找到了数字4！
正在查看数字2
机器人找到了数字2！
正在查看数字5
正在查看数字6
机器人找到了数字6！
正在查看数字8
机器人找到了数字8！
正在查看数字7
机器人找到了数字7！
正在查看数字9
机器人找到了数字9！
```

综合案例

- 看懂下面代码

```
a = 10
b = 20
result = a + b
while True:
    answer = int(input(f"请输入{a}+{b}的结果"))
    if result == answer:
        print("回答正确！")
        break
    else:
        print("回答错误")
```