HD EDU

# INFO1110
# Introduction to Programming

## 同步辅导课2

Tutor：Hilbert

HD·EDUCATION
全球累计服务用户超十万

HD EDU



HD EDU

## 企业介绍

海道教育（HD EDU）成立于2018年，是一家助力国际化人才成长的教育科技企业，同时也是一个力求突破语言和文化障碍、推动国际教育与文化属性相融合的新锐品牌。

海道教育已实现全球规模化发展，旗下产品已走入英国、美国、澳大利亚、新西兰等主流留学的国家40多所院校，为超30万的中国留学生提供高品质的教学服务，成为留学后市场在线教育的领军企业。

2022年，HD EDU发布首个学术诚信合规准则，HD EDU愿始终与所有用户一起，坚守学术尊严，真正意义上打造既保护用户权益又懂用户需求的教学产品，推动海外华人留学生成长与发展。

## 产品介绍

成立五年来，HD EDU 作为专注于留学后市场服务的后起之秀，秉持着“提升中国留学生核心竞争力”的产品理念，深度洞察留学生教育场景叠加、需求变化，研发多阶段教育产品，为用户提供多元化、高品质的一站式留学教育服务，打造出符合留学生全生命周期的三大教育产品线：HD Tutoring、HD Career、HD Offer。

### HD Tutoring

学习有海道，提分更有效

节点辅导　全学期学业辅导

### HD Career

让年轻人找到喜欢的工作

技能课程　资格证书　实训项目　求职方案

### HD Offer

专注藤校G5级别院校申请

选校定位　文书创作　申请服务　面试辅导


HD EDU
以用户的成长为第一

HD EDU

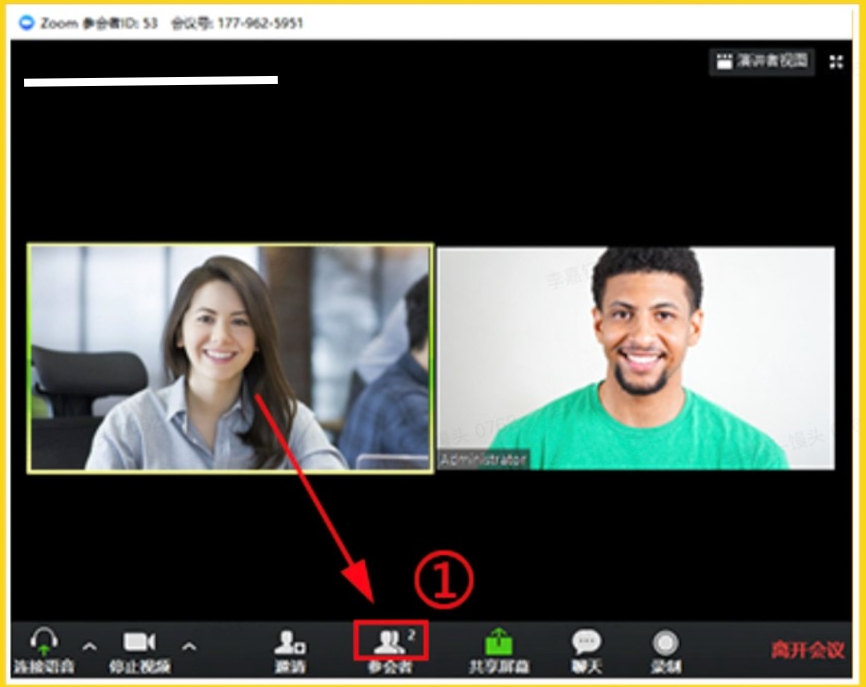


# TUTOR Hilbert

## Self- Introduction

- USYD Bachelor of Advanced Computing Honours First Class毕业 (Major in Software Development)
- 東京工業大学Master of Artificial Intelligence在读
- 曾任USYD INFO1110的Lab Tutor
- INFO1110总分91.0 HD
- 2018年加入HD Edu，主要教授预科及本科的IT入门课程
- HD Edu学科带头人兼面试官，曾经撰写过HD的INFO课程规范指南

# 同学们遇到问题怎么办？










1. 点【参会者】
2. 点【举手】即可与老师实时互动
3. 问题被解答了还可以【手放下】
4. 点【聊天】/【Chat】就可以与老师实时文字互动哦

# 直播平台互动方法

## 直播平台 ： 举手+聊天室提问



1. 点【参会者】再点【举手】，即可与老师实时互动！在此输入你想问的问题
2. 问题被解答了还可以【手放下】

# 本章知识点

1. Flow Chart
2. Condition
3. Truthiness *
4. Short-circuiting *
5. Iteration
6. Desk Check
7. Binary vs Decimal

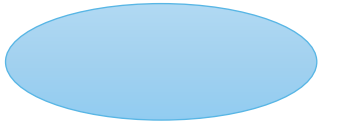


HD EDU




# 知识点讲解

# 考点1. Flow Chart



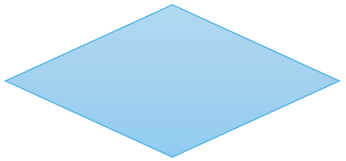
❖ Start/End of a program



❖ Input/Output



❖ Statements



❖ Condition

*Flow chart -> Code*

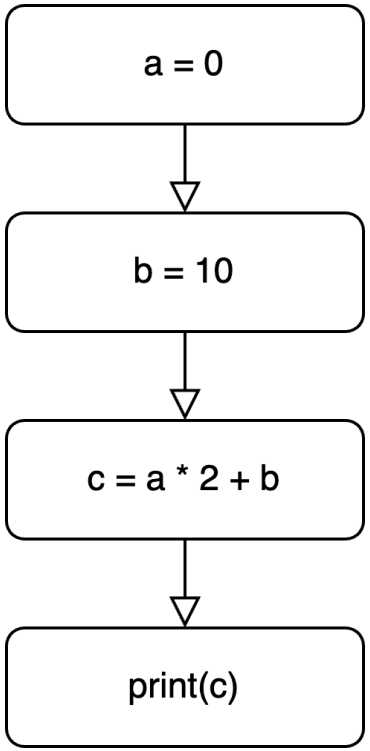
*Code -> Flow chart*

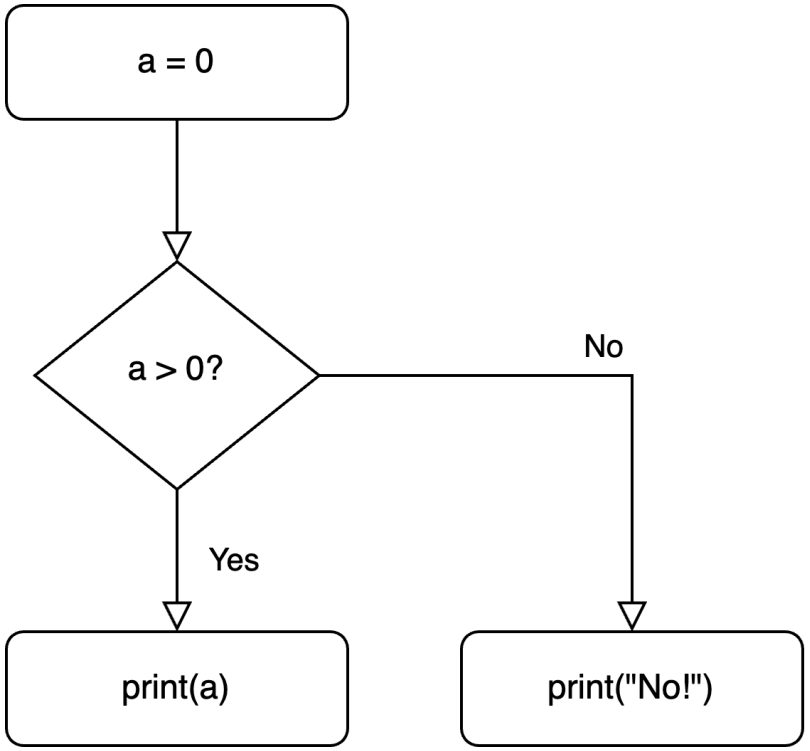
*都要会!*

# 考点2. 条件 Condition

HD EDU

- 在学习if之前，我们的程序都是线性执行的（从上到下）
- 而从这一章开始，我们程序的执行Flow开始有**岔路**了！



# 考点2. if

HD EDU

❖ 标准格式：

```
if <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
<Main Body>
```

❖ Condition：

True/False

↑

a == 12/b >= c

↑

a == 12 and b >= c or c ** 2.5 / 2 % 10 …

❖ 程序执行到if时，当condition evaluate为True时，就会进入Sub Code Block执行里面（缩进indented）的代码。

❖ 当indented部分的代码执行完成后，又会出来到Main Body的部分继续执行。

❖ 如果condition为False，那程序会跳过Sub Code Block的所有部分，直接执行Main Body的部分。

# 考点2. Indentation缩进

HD EDU

- 如何区分Main Body和Sub Block完全看该行代码是否indent，所以indentation在Python中尤为重要！

```
if <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
    print("A")
<Main Body>
```

```
if <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
print("A")
<Main Body>
```

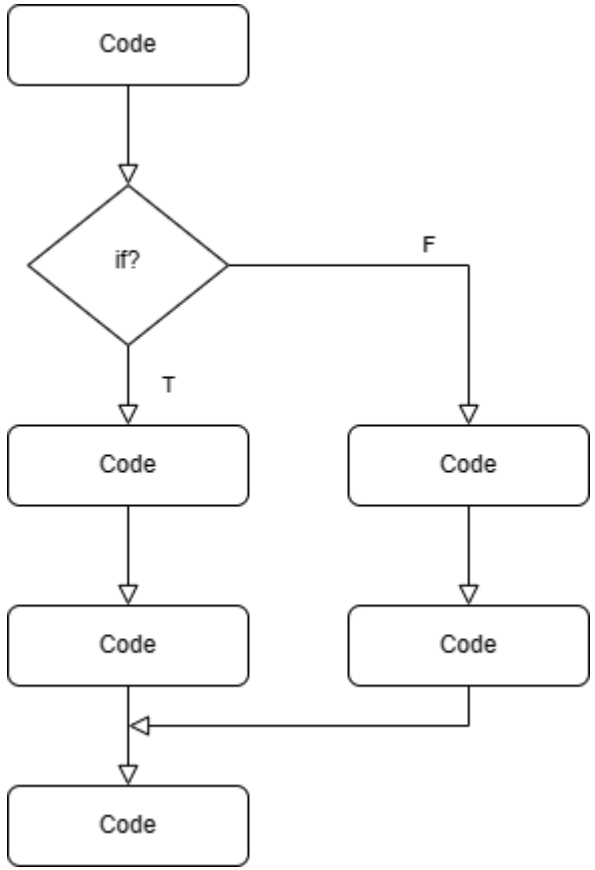
- 按Tab键，输入\t来做indent
  - 按空格键也可以，常见的indent width=2, 4, 8。Python会自动检测，要consistent
- * Soft tab: the tab key insert **spaces**

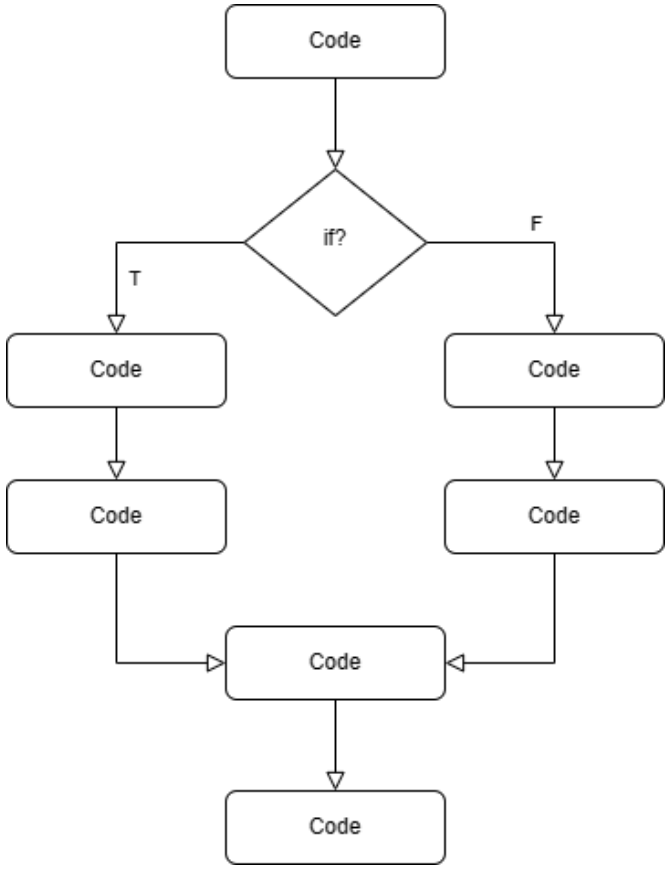
**TabError: inconsistent use of tabs and spaces in indentation**

# 考点2. 条件 - else

- else**必须**跟在一个同级（indent等级）的if statement后面
- 表示，前面条件区间的**补集**的情况下，程序要做的事情

```
if <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
else:
    <Sub-code block>
    <do something...>
    ...
    ...
<Main Body>
```



# 考点2. 条件 - else

HD EDU

- else表示补集可以非常便利的帮我们把前文复杂逻辑的补集给表示出来，而不需要我们把反面也一一列举
- 但是，有的时候这个便利性也会让我们失去对于补集的判断

```
a = <something>

if a == 10:
  print("Yes")
else:
  print("No")
```

- 请问代码中else部分所指代的集合是?
- a不为零的情况?
- 不要忘记a也可以不是number

# 考点2. 条件 - elif

HD EDU

- elif**必须**跟在一个同级（indent等级）的if statement后面
- if接elif的数量没有限制，最后可以选择接else代表以上所有条件合集的补集

```
if <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
elif <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
<Main Body>
```

```
if <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
else:
    if <condition>:
        <Sub-code block>
        <do something...>
        ...
        ...
<Main Body>
```

```
if <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
elif <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
else:
    <Sub-code block>
    <do something...>
    ...
    ...
<Main Body>
```

```
if <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
elif <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
elif <condition>:
    <Sub-code block>
    <do something...>
    ...
    ...
<Main Body>
```

# 考点2. 条件 - elif

HD EDU

- if-else**只是**双叉路，但是因为elif的使用次数是不限的，所以原则上说我们的分叉点现在一个路口可以有**无数**个分支branches
- 但是，无论如何程序执行的flow在一个分叉点，**有且只会**走进一条岔路。

```
a = <something>

if a < 10:
  print("A")
elif a < 9:
  print("B")
elif a < 8:
  print("C")
```

```
a = <something>

if a < 10:
  print("A")
elif a < 11:
  print("B")
elif a < 12:
  print("C")
```

# 考点2. 废话大全

HD EDU

```
a = <something>

if a < 10:
  print("A")
elif a < 11:
  print("B")
elif a < 12:
  print("C")
```

```
if True:
    <Sub-code block>
    <do something...>
    ...
    ...
```

```
if False:
    <Sub-code block>
    <do something...>
    ...
    ...
```

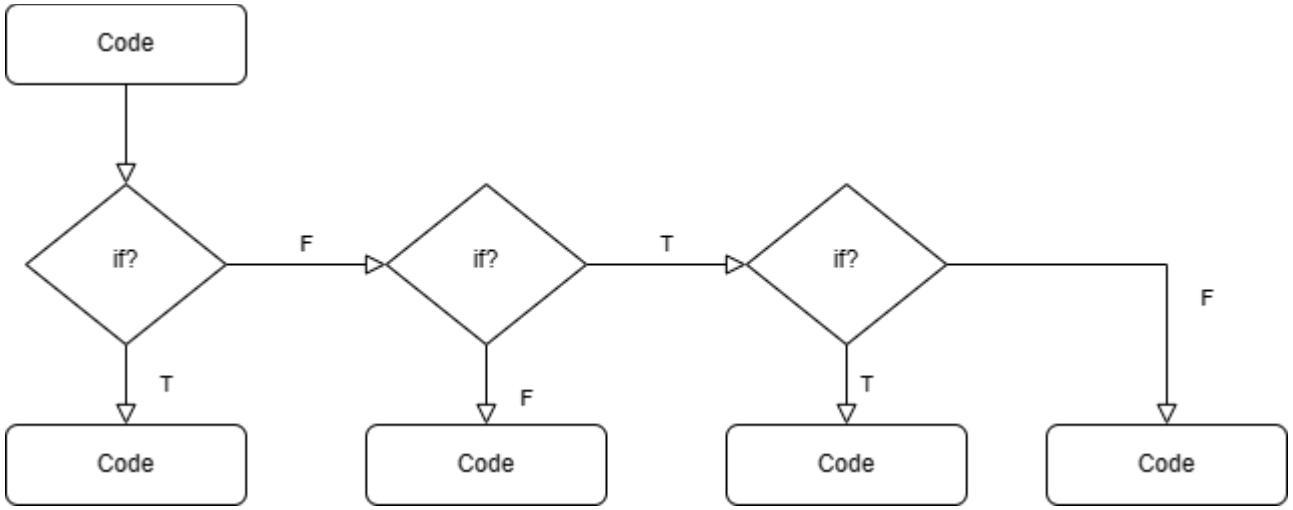
- 能正确执行的程序不代表它的每一行代码都是有意义的，有质量的。十万行代码的程序，未必就比一万行代码的程序质量高。
- 读高质量的代码！
- Keeping things simple, minimal and usable. Suckless

# 考点2. 条件 - Nested

HD EDU

❖ 由内到外，同一indentation level内从上到下的执行顺序

```
if <condition>:
    <do something...>
    if <condition>:
        <do something...>
        ...
    elif <condition>:
        if <condtion>:
            <do something...>
        <do something...>
        ...
    else:
        <do something...>
        ...
    <do something...>
else:
    if <condition>:
        <Sub-code block>
        <do something...>
        ...
        ...
<Main Body>
```



# 考点3. Truthiness

HD EDU

- type( ) function检测出来的每个datatype name都是一个built-in function用于casting
  - int( )
  - str( )
  - float( )
  - * bool( )

```
>>> type(1)
<class 'int'>
>>> int("3")
3
>>> type(True)
<class 'bool'>
>>> bool(3)
True
>>> bool("")
False
>>> not "1"
False
>>> if "hello world":
...     print("Yes")
...
Yes
```

```
>>> bool("")
False
>>> bool("0")
True
>>> bool(0)
False
>>> bool(0.0)
False
>>> bool(None)
False
```

# 考点4. Short-Circuiting

HD EDU

- 根据and和or的truth tables
  - and：二者中有一个False就为False
  - or：二者中有一个True就为True
- 因此，我们可以节约计算时间…

```
if True or input():
  print("A")
```

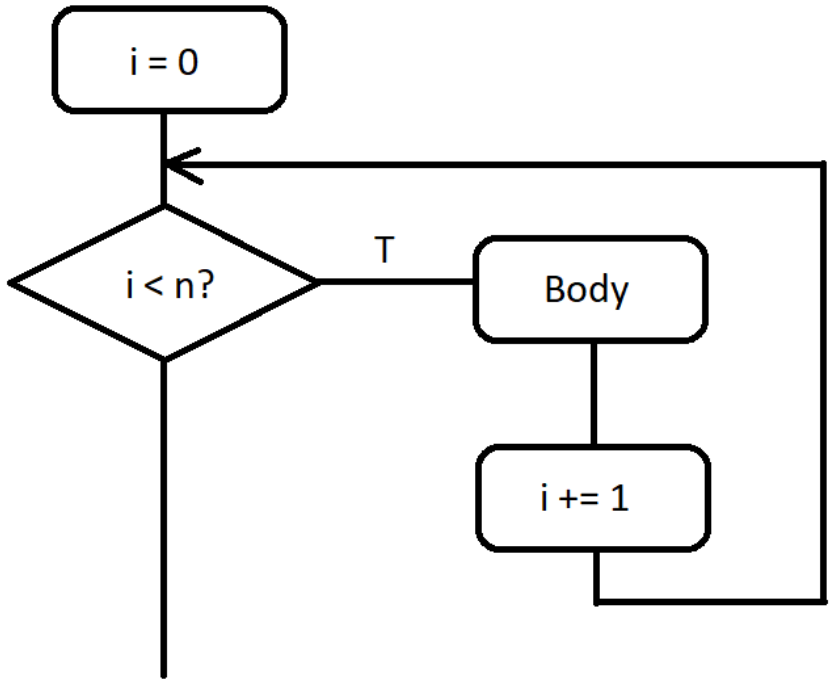
```
>>> if True or int("a"):
...     print("A")
...
A
```

- Python也会偷懒
- 所以，被省去计算的部分不会执行
  - 该报错不会报
  - 该做的事没做

# 考点5. Iteration迭代

HD EDU

- 在学习Iteration之前，我们写的代码最多只会执行一遍
- 而从这一章开始，我们程序的执行Flow开始有**循环**了！




- 学完循环以后，我们Control Flow部分的所有实现操作就全部学完了！
- 循环也正是最体现计算机相比人类的优势。Computers能**高速的完成**repetitive tasks

# 考点5. while and for

HD EDU

- 在一般的编程语言中，while和for两个keyword都可以用来实现循环。而且二者可以**互相替代**（能用while写出来的循环也一定能用for写出）
- Python的for稍微有些特殊，所以INFO1110并没有同时教
  - 以while为重点
  - for教了也未必让用
  - while更能练习使用循环的基本功
  - 只用while也能解决该门课所有需要用到循环的场景

# 考点5. while loop

HD EDU

❖ 标准格式：

```
while <condition>:
    <do something...>
    ...
    ...
<Main Body>
```

❖ Condition：

True/False

↑

a == 12/b >= c

↑

a == 12 and b >= c or c ** 2.5 / 2 % 10 …

❖ 程序执行到while时，当condition evaluate为True时，就会进入Sub Code Block执行里面（缩进indented）的代码。

❖ 当indented部分的代码执行完成后，又会回到while condition的部分重新判断。

❖ 直到condition为False，程序会跳过Sub Code Block的部分，出到Main Body的部分。

# 考点5. while loop

HD EDU

- **Infinite loop**: Stop condition is **always** evaluated to True.

```
while True:
  do ...
```




当你在Mark的时候出现，等了半天最后提示Program runs for too long，那么**大概率**你的程序里出现了infinite loop (Ctrl + C)

- **Counter-controlled loops** —— repeat a specified number of times (when you know upfront the number of iterations)

```
i = 0
while i < 5:
  print(i)
  i += 1
```

```
stop_iteration = False
while stop_iteration:
  if <some_condition>:
    stop_iteration = True
```

- **Event-controlled loops** —— some condition within the loop body changes and this causes the repeating to stop and the loop to be exited (when you don't know the exact number of iterations)

# 考点5. Counter-controlled loop 范式idiom

HD EDU

```
i = 0
while i < n:
  do ...

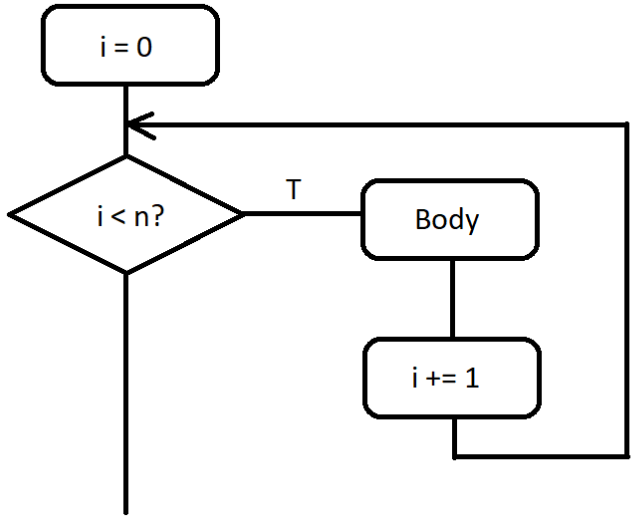
  i += 1
```

Counter Variable

Stop Condition

Body

Increment / Decrement

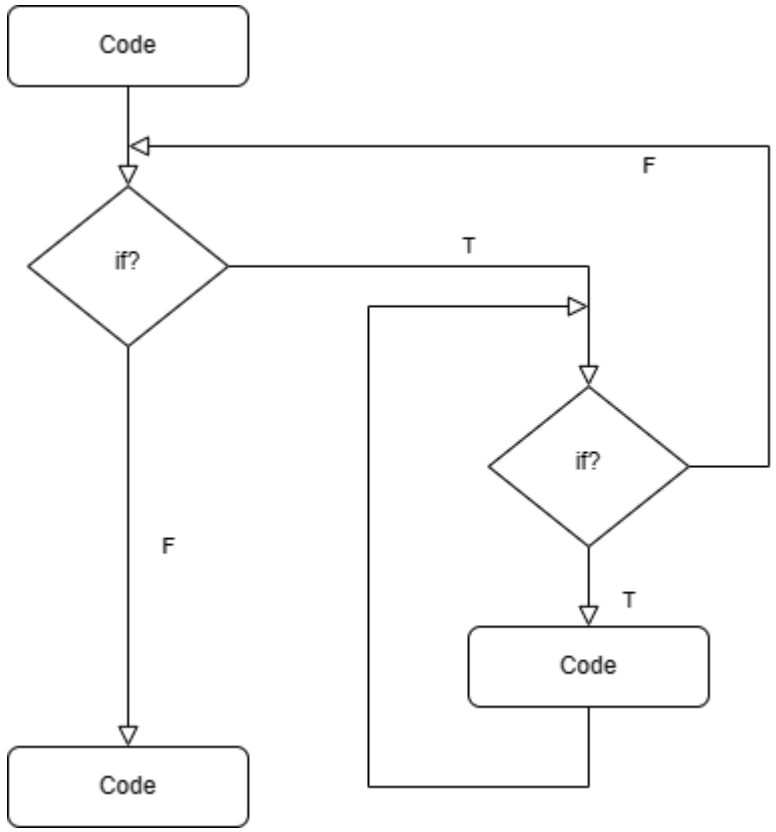


# 考点5. 循环 - Nested

HD EDU

❖ 由内到外，同一indentation level内从上到下的执行顺序

```
while <condition>:
    <do something...>
    while <condition>:
        <do something...>
        ...
    <do something...>
<Main Body>
```




[1, 2, 3, 4, 5],
[1, 2, 3, 4, 5],
[1, 2, 3, 4, 5],
[1, 2, 3, 4, 5]

matrix

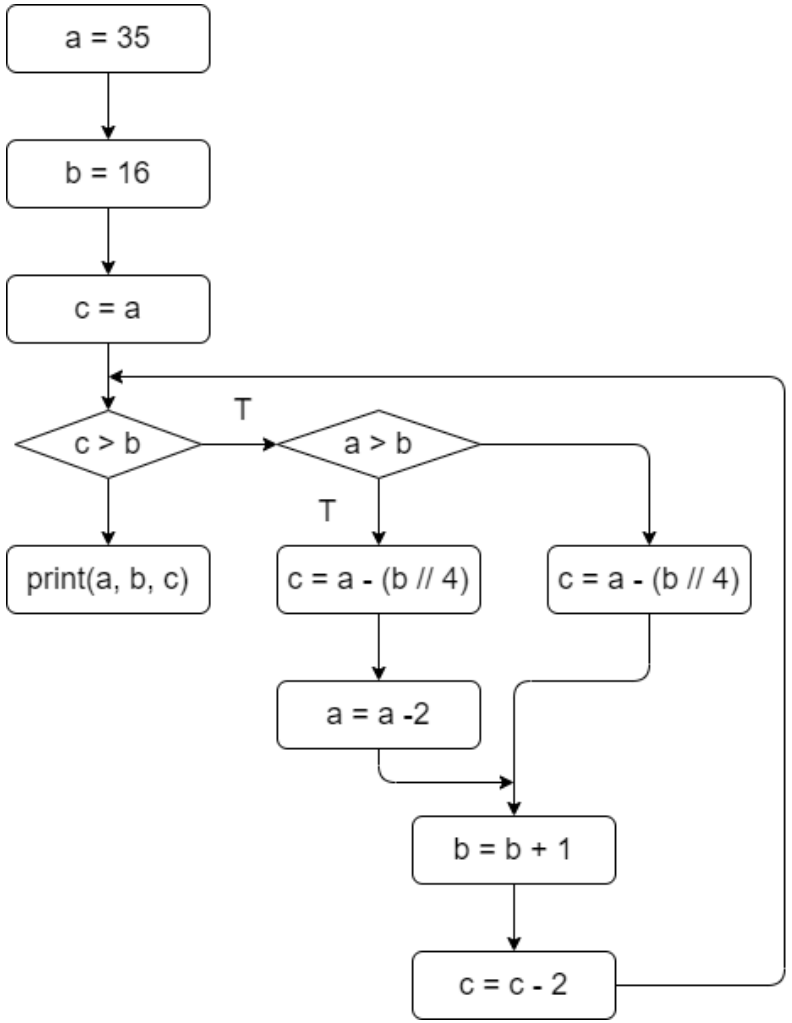
❖ 运算呈**指数**型增长

# 考点6. Desk Check and Variable Tracing

HD EDU

```
a = 35
b = 16
c = a
while c > b:
    if a > b:
        c = a - (b // 4)
        a = a - 2
    else:
        c = a - (b // 4)
    b = b + 1
    c = c - 2
print(a, b, c)
```

| a | b | c |
|---|---|---|
| 35 | | |
| 35 | 16 | |
| 35 | 16 | 35 |
| 35 | 16 | 35 |
| 35 | 16 | 31 |
| 33 | 16 | 31 |
| 33 | 17 | 31 |
| 33 | 17 | 29 |
| 33 | 17 | 29 |
| 31 | 17 | 29 |
| 31 | 18 | 29 |
| 31 | 18 | 27 |
| 31 | 18 | 27 |
| 29 | 18 | 27 |
| 29 | 19 | 27 |
| 29 | 19 | 25 |
| 29 | 19 | 25 |
| 27 | 19 | 25 |
| 27 | 20 | 25 |
| 27 | 20 | 23 |
| 27 | 20 | 22 |
| 25 | 20 | 22 |
| 25 | 21 | 22 |
| 25 | 21 | 20 |
| 25 | 21 | 20 |




- A desk check, also known as dry run or trace table, is a manual way of walking through the code to understand how it executes without actually running it on a computer. It involves tracking the values of variables step by step.

# 考点7. Binary vs Decimal

HD EDU

- N进制就是逢n个数字向下一位进1
  - 十进制：0~9后变成10
  - 二进制：0,1,10
  - 十六进制：0~9，（没满十六怎么办，用字母来凑），A~F

```
>>> hex(16)
'0x10'
>>> hex(15)
'0xf'
>>> bin(16)
'0b10000'
>>> bin(8)
'0b1000'
>>> hex(25)
'0x19'
```

| Binary 0b | Decimal | Hexadecimal 0x |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 10 | 2 | 2 |
| 11 | 3 | 3 |
| 100 | 4 | 4 |
| 101 | 5 | 5 |
| 110 | 6 | 6 |
| 111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | 10 | A |
| 1011 | 11 | B |
| 1100 | 12 | C |
| 1101 | 13 | D |
| 1110 | 14 | E |
| 1111 | 15 | F |
| 10000 | 16 | 10 |

# 考点7. Binary ⇔ Decimal

HD EDU

- 二进制转十进制
  - 将二进制数的每一位乘以2的对应次幂，并求和。

| 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 |

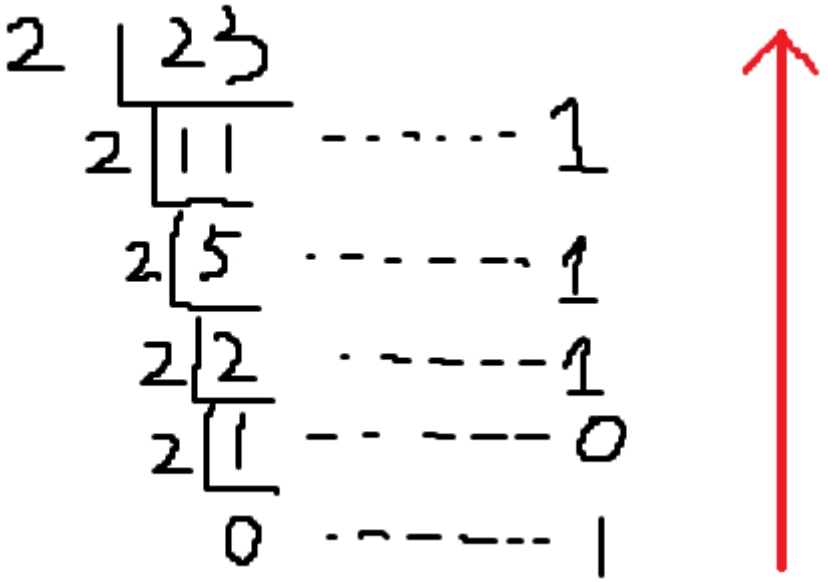
$$1\times2^4+0\times2^3+1\times2^2+1\times2^1+1\times2^0=23$$

- 该方法适用于任何进制转十进制，只需把底数改为进制数即可

# 考点7. Binary ⇔ Decimal

HD EDU

- 十进制转二进制
  - 将十进制数除以2，记录余数（0或1），直到商为0为止。
  - 从下往上记录余数，就得到了对应的二进制数。




```
n = input("Enter the decimal number: ")
n = int(n)

result = ""
while n != 0:
    remainder = n % 2
    n //= 2

    if remainder == 1:
        result = "1" + result
    else:
        result = "0" + result

print(result)
```




# 经典题型放送

# Code - Sum

(自编题)

Write a program `str_sum.py` that receives a string of digits, then calculates and prints the sum of every single digit from the string.

```
$ python str_sum.py
#12345
The sum is 15
$ python str_sum.py
#968810
The sum is 32
```

# Code – My Terminal

HD EDU

(自编题)

It's time to design your own terminal. It has 3 commands:

- **EXIT**: This command will shut down your terminal.
- **HELP**: This command will print a detail explanation of all the commands available.
- **ECHO** <*message*>: Your terminal will print the <message> to the screen.
- If the user gives any other input, display an error message.

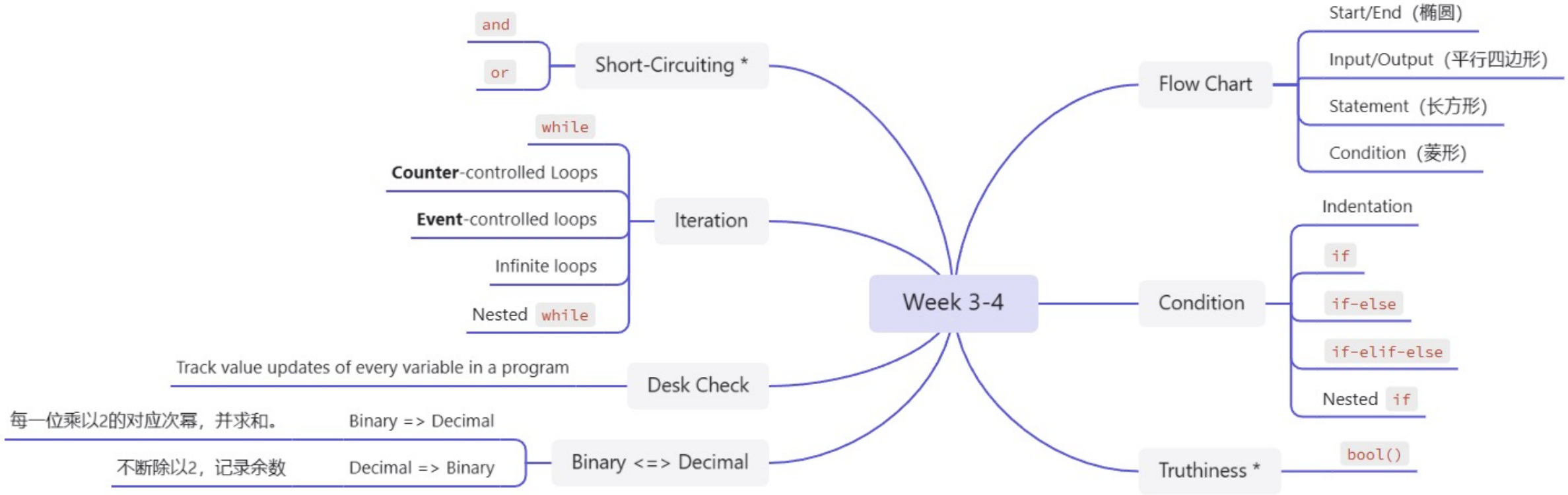
```
$ python term.py
>>>HELLO
Command HELLO not found.
>>>HELP
Command List
EXIT --- Shutdown the program
HELP --- Get the list of all available commands
ECHO --- Print the given message to the screen
>>>ECHO Hilbert is happy!
Hilbert is happy!
>>>echo Yay
Command echo Yay not found.
>>>exit
Command exit not found.
>>>EXIT
```

Command List
EXIT --- Shutdown the program
HELP --- Get the list of all available commands
ECHO --- Print the given message to the screen




# 重难点总结

# 重难点总结




❖ 下周的Test好好加油喔！




# 下节课预告

# 下节课预告

HD EDU

- Functions
  - Variable Scope
  - `Print vs Return`
- Collection Datatypes
  - `List`
  - `Arrays`
- Command-line Arguments
- `Break/Continue`



# 您每一份诚挚的问卷都会拯救一根濒临掉落的头发。

HD EDU



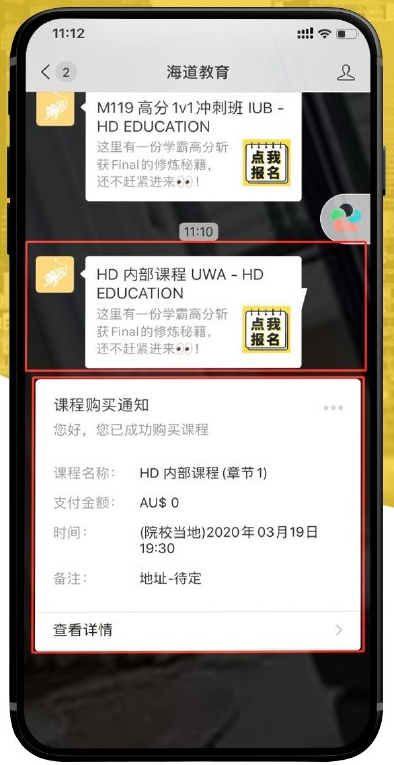
帮我开个张吧，求求你了！



课程结束后，如果您对课程或者服务的任何建议和意见

**请给予我们提高和改进的机会，感谢您对 HD•EDUCATION 课程和服务的信任！**

# · 填写问卷操作流程 ·




**第一步**

关注【海道教育】服务号后
点击【购买通知】或【上课提醒】




**第二步**

点击【前往学习】




**第三步**

点击【去评价】就可以为课程进行评价




# Q & A