HD EDU

INFO1110
Introduction to Programming

同步辅导课3

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

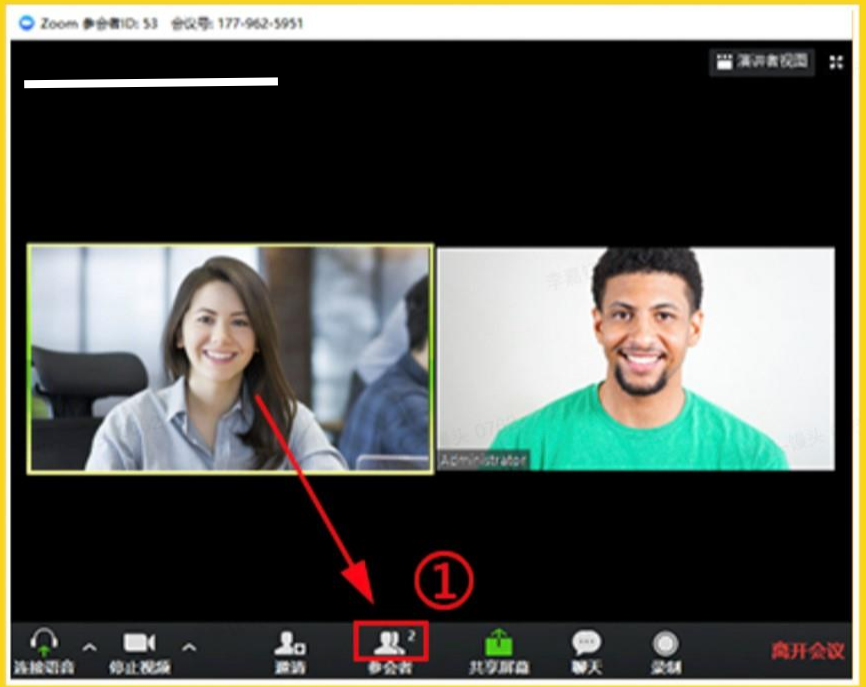


TUTOR Hilbert

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 of Execution and Desk Check
2. Function
3. Variable Scope
4. Collection Datatype
5. Command Line Arguments
6. Primitive vs. Reference Data Types
7. pdb



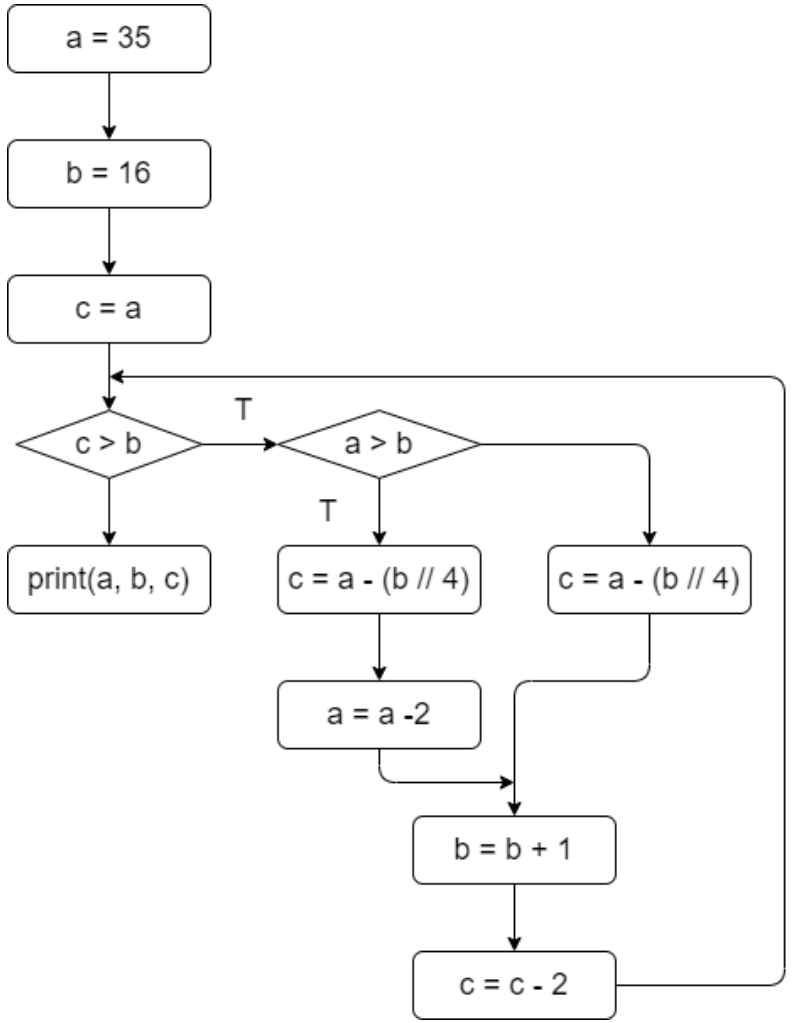


知识点讲解

考点1. Flow – Variable Tracing and Desk check

```
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1. 目前已学的Operations

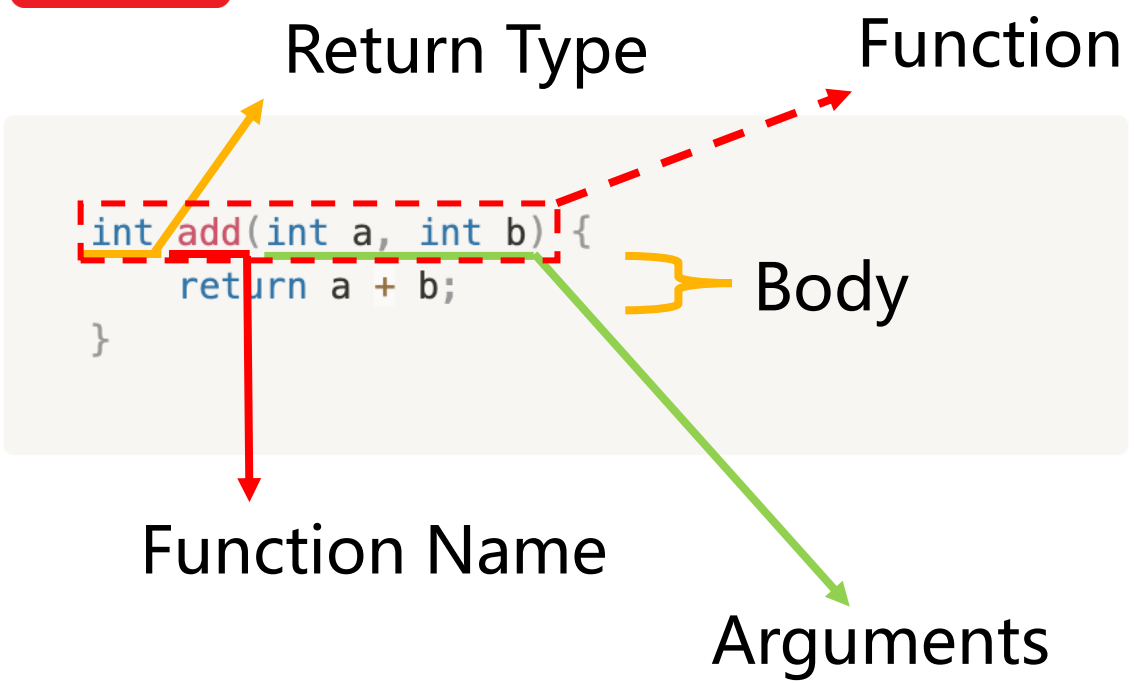
HD EDU

- Built-in Functions/Keywords
  - `import`
  - `print(), input(), …`
- Variables的赋值和更新
- （Arithmetic, Relational, Logical）运算
- Control Flow (Condition and Iteration)
- String Formatting
- Type Conversion

考点2. Function Basics



❖ 这部分不是在讲Python，而是Programming Basics



❖ Function Name

❖ Arguments: Inputs

❖ Return: Outputs

❖ Function Body

我们用call作动词，意指使用一个已经defined function: `add(1, 2);`

考点2. Python Function Basics

HD EDU

```python
def add(a, b):
    return a + b
```

- Function Name
  - 取名规则和Variable Name相同
- Arguments
  - Python对于arguments type没有严格要求
- Return
  - Python对于return type没有严格要求
- Function Body

```python
def add(a, b):
    return a + b

print(add(1, 2))
print(add("1", "2"))
print(add("1", 2))
```

```
$ python3 temp.py
3
12
Traceback (most recent call last):
  File "/Users/soda/Library/CloudStorage/OneDrive-Personal/HD-Edu/INFO1110/2024 Semester 1/同步辅导课/同步辅导课3/temp.py", line 6, in <module>
    print(add("1", 2))
          ^^^^^^^^^^^
  File "/Users/soda/Library/CloudStorage/OneDrive-Personal/HD-Edu/INFO1110/2024 Semester 1/同步辅导课/同步辅导课3/temp.py", line 2, in add
    return a + b
           ~~^~~
TypeError: can only concatenate str (not "int") to str
```

考点2. Python Function Type Hints *

HD EDU

```python
def add(a: int, b: int) -> int:
    return a + b

print(add(1, 2))
print(add("1", "2"))
print(add("1", 2))
```

```
$ python3 temp.py
3
12
Traceback (most recent call last):
  File "/Users/soda/Library/CloudStorage/OneDrive-Personal/HD-Edu/INFO1110/2024 Semester 1/同步辅导课/同步辅导课3/temp.py", line 6, in <module>
    print(add("1", 2))
          ^^^^^^^^^^
  File "/Users/soda/Library/CloudStorage/OneDrive-Personal/HD-Edu/INFO1110/2024 Semester 1/同步辅导课/同步辅导课3/temp.py", line 2, in add
    return a + b
           ~~^~~
TypeError: can only concatenate str (not "int") to str
```

- 新版Python有提供Type **Hints**的Syntax来补足缺失的Type信息
- 但是Python仍然不会enforce
- 所以代码运行结果仍然不变

考点2. Type Check *

HD EDU

- Python是一个对type要求相对宽松的语言，尽管我们可以通过Function Signature来enforce，但有的时候我们仍然希望验证一个variable所存的data type

```
    1 def f(a: int) -> int:
    2     return a + 2
    3
E   4 print(f("1"))      ■ Argument of type "Literal['1']" cannot be assigned to
 Diagnostics:
 1. Argument of type "Literal['1']" cannot be assigned to parameter "a" of type
  "int" in function "f"
      "Literal['1']" is incompatible with "int" [reportGeneralTypeIssues]
```

```
isinstance(<variable>, <type>)

if type(<variable>) == <type>:
  do ...
else:
  do ...
```

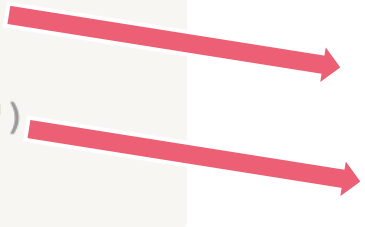
```
>>> a = 1
>>> isinstance(a, int)
True
>>> isinstance(a, float)
False
>>> if type(a) == int:
...     print("Yea")
... else:
...     print("No")
...
Yea
```

Pay attention that no “ ” are here!

考点2. Parameters vs Arguments

HD EDU

```
def myfunction(fname, lname)
  print(fname + " " + lnmae)
myfunction("Emil", "Refsnes")
```



Parameters

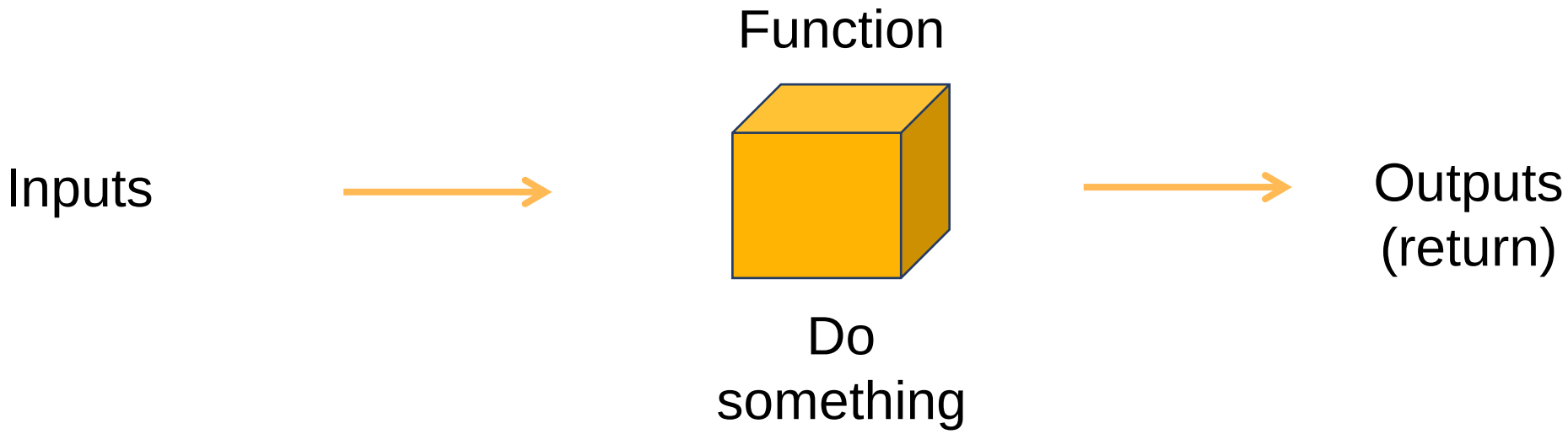
Arguments

From a function's perspective:

- A parameter is the variable **listed inside the parentheses** in the function definition.
- An argument is the **value that are sent** to the function when it is called.

考点2. print( ) vs return (Reflect back to Wk1)

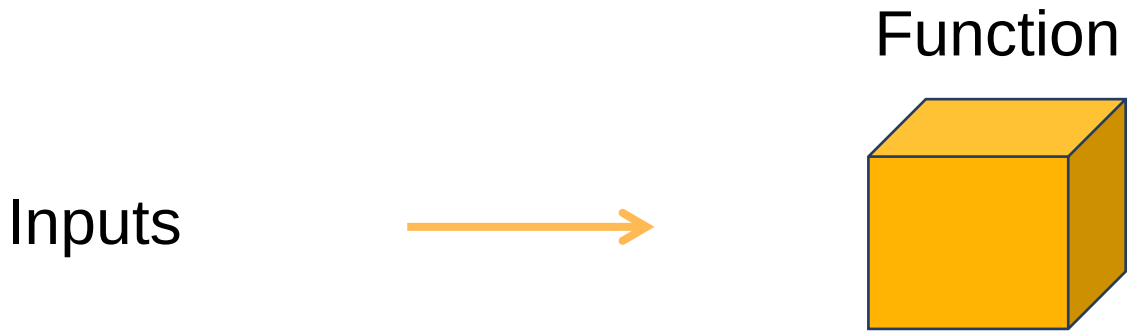
HD EDU




```
print("Hello")
print("Hello", "World", 123321)
print(a)
print(12)
print(-12.5)
print(True)
```

```
>>> b = print("Hello world!")
Hello world!
>>> print(b)
None
>>> type(b)
<class 'NoneType'>
```

考点2. print( ) vs return (Reflect back to Wk1)



- print( ) 把Message给人看
- return 把function运行后的最终结果返回给电脑
- return 可以返回任何object，甚至是Function calls

```
def f():
    return input()
```

```
$ python3 temp.py
1
1
1
None None None
```

```
def f1():
  print("1")

def f2():
  print("1")
  return

def f3():
  print("1")
  return None

a = f1()
b = f2()
c = f3()
print(a, b, c)
```

考点2. import *

HD EDU

❖ import keyword让我们可以从别的文件中导入代码

import <file path>

```python
# a.py
def add(a, b):
  return a + b

print("Hello World")
```

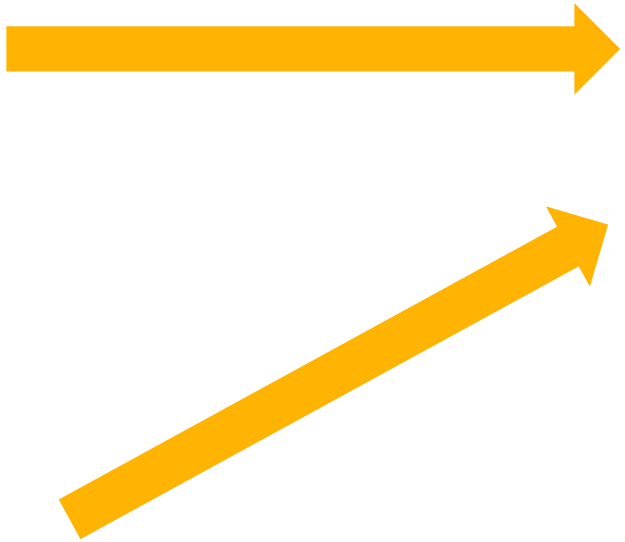
```python
# b.py
import a

print(a.add(1, 2))
```

```python
# import a
def add(a, b):
    return a + b

print("Hello World")

print(add(1, 2))
```



```
$ python3 b.py
Hello World
3
```

考点2. import *

❖ 我们一般只想import功能性的代码，而不是文件里的正文。

```python
# a.py
def add(a, b):
    return a + b

print("Hello World")
```

这一块我们不想要

```python
# b.py
import a

print(a.add(1, 2))
```

❖ 所以我们可以把正文写在这一个if语句里：

```python
# a.py
def add(a, b):
    return a + b

if __name__ == "__main__":
    print("Hello World")
```

❖ 对于if里的条件，__name__是Python的一个Built-in Variable。里面存的值只有当python a.py运行时，才会等于”__main__”

```
$ python3 b.py
3
```

Ref. https://docs.python.org/3/library/__main__.html

考点2. Main Function

HD EDU

```java
public class Test {
  public static void main(String[] args) {
    System.out.println("Hello!");
  }
}
```

```c
#include <stdio.h>

int main() {
    // Print "Hello, World!" to the console
    printf("Hello, World!\n");
    return 0;
}
```

- 在常用的编程语言像C(COMP2017), Java(INFO1113)中，程序正文一般都会写在一个叫**main**的function definition内作为切入点。
- 但是Python不需要。

我们可以模拟这种感觉：

```python
def main(args):
  <do something>
  return

if __name__ == "__main__":
    main()
```

考点2. Functions - Why using functions?

HD EDU

- ❖ Separation
- ❖ Easy to reuse
- ❖ Reduce chance of errors

```
print("a")
print("a")
print("a")
print("a")
print("a")
print("a")
print("a")
print("a")
print("a")
print("a")
```

```
def print5a():
  print("a")
  print("a")
  print("a")
  print("a")
  print("a")
print5a()
print5a()
```

```
def myprint(string, times):
  i = 0
  while i < times:
    print(string)
    i += 1
  return
myprint("a", 10)
```



Specific *Generic*

考点3. Variable Scope

HD EDU

❖ Accessibility of defined variables

```
def func():
  a = 1
  return

print(a)
```

Error

```
def func():
  a = 1
  return
func()
print(a)
```

Error

```
def func():
  a = 1
  print(a)
  return
a = 2
func()
```

1

```
def func():
  print(a)
  return
a = 1
func()
```

1

```
def func():
  a = 1
  return
a = 2
func()
print(a)
```

2

```
def func():
  global a
  a = 2
  return
func()
print(a)
```

2

```
def func():
  global a
  a = 2
  return
a = 1
func()
print(a)
```

2

考点3. global

HD EDU

- global这个keyword能够 **overwrite** scope默认的行为
- Functions that modify out-of-scope variables are called functions with side effects.
- Pure functions are functions without side effects.

```
# Declare a global variable
x = 10

def my_function():
    # Trying to modify x without the global keywor
d will create a new local variable
    x = 20
    print(x)  # Output: 20

def another_function():
    global x  # Declare that we want to use the gl
obal variable x
    x = 30
    print(x)  # Output: 30

my_function()  # Output: 20
print(x)  # Output: 10 (the global variable is unc
hanged)

another_function()  # Output: 30
print(x)  # Output: 30 (the global variable is now
changed)
```

考点4. Collection (Container) Datatypes

HD EDU

- A collection datatype is a type of objects that can store a group of objects.
- Array



  - Array is the most primitive collection datatype in most of programming languages. (**Not in Python by default**)
  - You need to define the data type of the objects to be stored in an array.
  - You need to specify the number of objects to be stored in an array.

```
int myArray[5];
```

C Array Initialization

考点4. Collection (Container) Datatypes

HD EDU

- 共有特点：
  - Length：指代collection的size
    - 都能通过 `len()` function获得一个integer的长度
  - Index：都可以通过index来定位到特定位置的一个object
    - Index starts from 0
    - Negative index *
  - Slice operation: 都可以使用slicing operation分割出一个sub-collection
  - Concatenation: collection1 + collection2 = collection12
  - Dictionary除外

考点4. String, Tuple, List

HD EDU

❖ String

```
>>> a = "123456789"
>>> len(a)
9
>>> a[5]
'6'
>>> a[1:7]
'234567'
>>> '123' + '576'
'123576'
>>> '123' * 5
'123123123123123'
>>> str(1)
'1'
```

❖ Tuple

```
>>> b = (-10.5, True, "a")
>>> len(b)
3
>>> b[1]
True
>>> (1, 2) + ('a', 'b')
(1, 2, 'a', 'b')
>>> (1, 2) * 4
(1, 2, 1, 2, 1, 2, 1, 2)
>>> tuple("abc")
('a', 'b', 'c')
```

❖ List

```
>>> c = []
>>> c.append(True)
>>> c.append((1, 2))
>>> c
[True, (1, 2)]
>>> c.insert(1, -1.5)
>>> c
[True, -1.5, (1, 2)]
>>> c.remove(True)
>>> c
[-1.5, (1, 2)]
>>> c += [1, 2, 3, 4, 5]
>>> c
[-1.5, (1, 2), 1, 2, 3, 4, 5]
>>> del c[1]
>>> c
[-1.5, 1, 2, 3, 4, 5]
>>> len(c)
6
>>> [1] * 4
[1, 1, 1, 1]
>>> list((1, 2))
[1, 2]
```

Official Doc

考点4. Dictionary, 2D Collection

HD EDU

- Dictionary本质上相当于一个增强版的list，一般的list只有value是有意义的，但是以index来进行索引。Dictionary给予了程序员给索引的方式也创造意义的自由。

- 2D的Collection：即Collection之间产生嵌套。这种情况下，我们可能需要连续使用若干个index operator才能进到最深层的collection中去

```
>>> twod = [[1, 2, 3, (5, 6, 7)], ['a', 'b', 'c']]
>>> twod[0][3][1]
6
```

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

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

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

[1, 2, 3, 4, 5]

matrix

```
>>> dict = {'1': 100}
>>> dict[True] = 50
>>> dict
{'1': 100, True: 50}
>>> dict['abc'] = False
>>> dict
{'1': 100, True: 50, 'abc': False}
>>> dict.keys()
dict_keys(['1', True, 'abc'])
>>> dict.values()
dict_values([100, 50, False])
>>> dict.items()
dict_items([('1', 100), (True, 50),
('abc', False)])
>>> list(dict.keys())
['1', True, 'abc']
>>> list(dict.items())
[('1', 100), (True, 50), ('abc', Fals
e)]
>>> list(dict.values())
[100, 50, False]
>>> len(dict)
3
>>> dict.get('1')
100
>>> dict.pop(True)
50
>>> dict
{'1': 100, 'abc': False}
```

Ref. https://docs.python.org/3.11/library/stdtypes.html#mapping-types-dict

考点4. Slicing

HD EDU

- When we want a single element from a collection, we can just use index operator:

```
>>> a = "abcde"
>>> a[1]
'b'
```

- When we want a subset from a collection, we use slice operator:

```
var[start:end:step]
```

- 当start=0、end=len(var)、step=1时，可以直接省略
- Python里的range永远是**包含起点不包含终点**。
- Slice operation是在原collection的基础上截取，生成一个新的collection。
- *当start、end、step为负值是会发生什么?

考点4. Slicing

HD EDU

```
>>> var
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> var[:6]
[0, 1, 2, 3, 4, 5]
>>> var[2:]
[2, 3, 4, 5, 6, 7, 8, 9]
>>> var[::2]
[0, 2, 4, 6, 8]
>>> var[1:8:2]
[1, 3, 5, 7]
>>> var[::-1]
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

- Slicing常用于String和List的processing。

考点4. Mutability - Mutable vs. Immutable

HD EDU

```
# Mutable types
a = [1, 2]
b = {'a': 1, 'b': 2}
# Mutable types could
# be modified
a[0] = 1
b['a'] = 2
print(a, b)
# Immutable types
c = "string"
d = (1, 2, 3)
c[1] = '1'
d[0] = 2
print(c, d)
```



```
[1, 2] {'a': 2, 'b': 2}
```



```
Traceback (most recent call last):
  File "/Users/apple/Desktop/教研/practice.py", line 16, in <module>
    c[1] = '1'
TypeError: 'str' object does not support item assignment
```

Whether an object's **internal state** can be modified or not after creation.

考点4. 从头到尾过一个Collection的范式

HD EDU

Traversal

```
i = 0
while i < len(collection):
        current_object = collection[i]
        <do something>
        i += 1
```

考点5. Command Line Arguments

HD EDU

```
$ python3 temp.py abc 12 666 True
```

```
$ cat temp.py
import sys

print(sys.argv)

# soda @ Mac in ~/OneDrive/HD-Edu/INFO111
$ python3 temp.py abc 12 666 True
['temp.py', 'abc', '12', '666', 'True']
```

- 跟在文件名后面输入的内容，归根结底就是给程序输入数据的一个方式
- 那我们应该如何在自己的程序里面调用这些arguments呢?
  - 使用sys module里的argv这个**list**
- **Caveat**



  - sys.argv这个list的第一个元素一定存在，且是这个程序的文件名
  - 无论我们input的是数字还是文本，list里的内容都是str

考点6. Primitive vs. Reference Data Types

但是，有的时候我们确实希望在内存里面复制一个新的list给variable b该怎么办？

```
a = 1
b = a
b = 2
print(a)
```

a = 1

```
a = [1]
b = a
b[0] = 2
print(a)
```

a: [2]

For primitive data types, data are stored as **values**.

For reference data types, data are stored as **references**.

考点6. Primitive vs. Reference Data Types

HD EDU

- def list_copy(li):

```
    new_list = []
    for each element in li:
        new_list.append(element)
    return new_list
```

- 纯手动，或者有更好的办法? 🤔

```
new_list = li[:]
new_list = list(li)
new_list = li.copy() <- This is a built-in method!
```

考点7. Python Debugger

HD EDU

```python
import random
import string

# Generate a random password
length = 10
characters = string.ascii_letters + string.digits + string.punctuation
password = ''.join(random.choice(characters) for _ in range(length))
print("Your random password is:", password)

# Shuffle a list
my_list = [1, 2, 3, 4, 5]
random.shuffle(my_list)
print("Shuffled list:", my_list)

# Choose a random item from a list
# choices = ['apple', 'banana', 'orange', 'pear']
# random_choice = random.choice(choices)
# print("Random choice:", random_choice)

# Generate random numbers
# print("Random integer between 1 and 10:", random.randint(1, 10))
# print("Random float between 0 and 1:", random.random())
```

- 我们先前教给大家的方法，是通过 comment out 无关的代码以及`print()`，来测试我们的程序是否达到我们期望的结果
- 而pdb本质上其实就是做一样的事情，只是把操作简化，不在需要我们去手动一行一行的 comment 代码了
- 用VSCode或者IDE的同学可以善用Breakpoint来帮助自己debug。（Ed好像也支持Breakpoint）



经典题型放送

Code – str.split

(自编题)

Write a function `split(s, sep)` that mimics the behaviors of Python built-in str.split() method.

- Assume len(sep) = 1

```
>>> '1 2 3'.split()
['1', '2', '3']
>>> '   1   2   3   '.split()
['1', '2', '3']
>>> '1,2,3'.split(',')
['1', '2', '3']
>>> '1,2,,3,'.split(',')
['1', '2', '', '3', '']
```

Boundary Cases

HD EDU

- An edge case is an issue that occurs **at an extreme** (maximum or minimum) operating parameter.
  一个维度的边界

- A corner case is when **multiple parameters** are **simultaneously** at extreme levels.
  多个维度同时到达边界







Reference: https://www.educative.io/answers/corner-case-vs-edge-case

Code – Sort

(自编题)

Write a function `sort(li)` that receives a list of integers and return a sorted list in ascending order.

- Assume the list does not contain duplicates.

```
sort([4, 5, 2, 8, 1, 0, 3, 7] -> [0, 1, 2, 3, 4, 5, 7, 8]
```

Intuitive Sort with Min

HD EDU

```
Function my_sort(collection):
        new_collection = []
        i = 0
        while i < len(collection):
                temp = find 第i小的element in collection
                new_collection.append(temp)
        return new_collection
```

Sort – in-place

HD EDU

```
Function bubble_sort(collection):

        i = 0

        while i < length of the collection then

                j = 0

                while j < length of the collection - i - 1 then

                        if collection[j] > collection[j+1] then

                                swap values between collection[j] and collection [j+1]

                        j += 1

                i += 1

        return
```

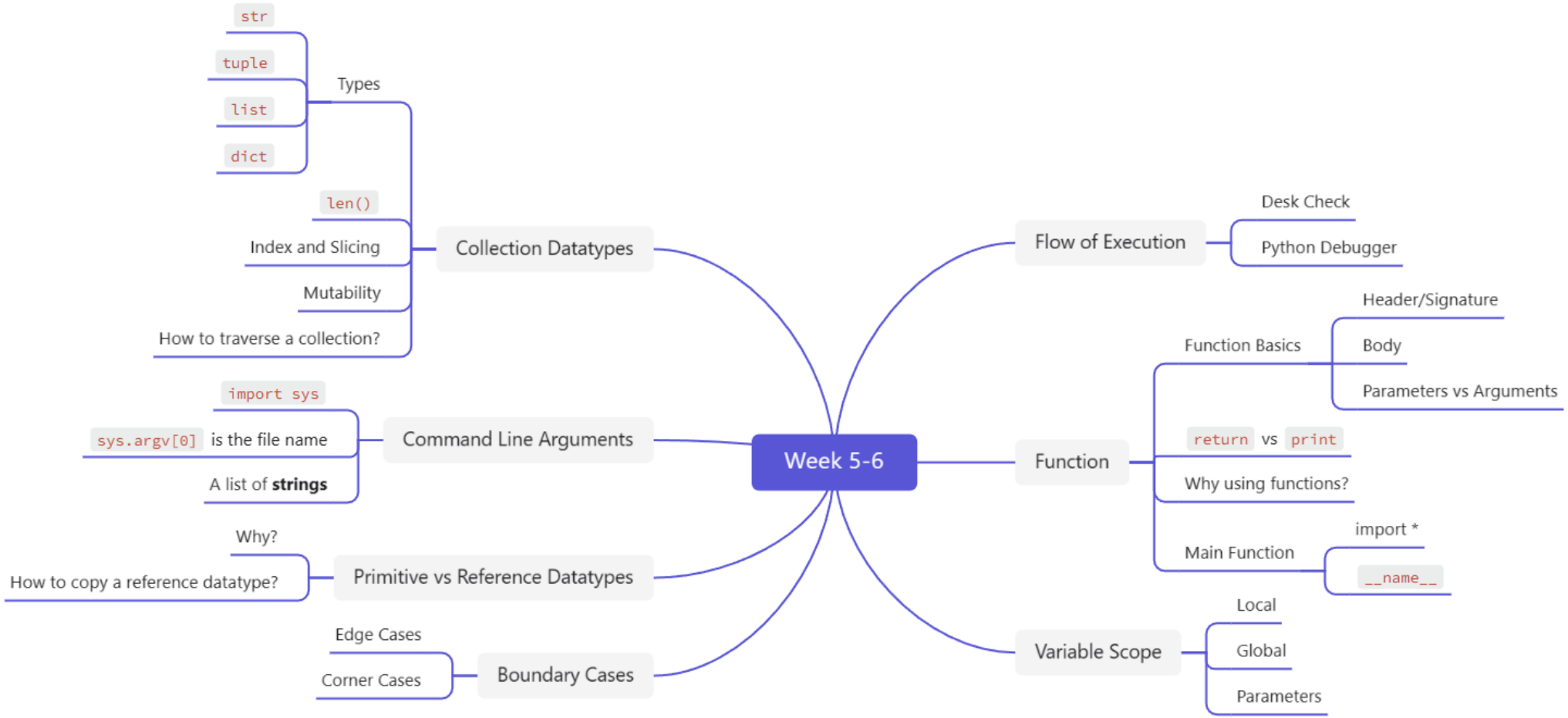
Descending order?





重难点总结

重难点总结



❖ 到目前为止，我们已经学完写程序功能性这一块的全部基本操作



下节课预告

下节课预告

HD EDU

- Classes and Objects
  - Constructors
  - Attributes
  - Methods
- File I/O
  - How to read a file?
  - How to write to a file?



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