HD EDU

INFO1110
Introduction to Programming

同步辅导课4

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

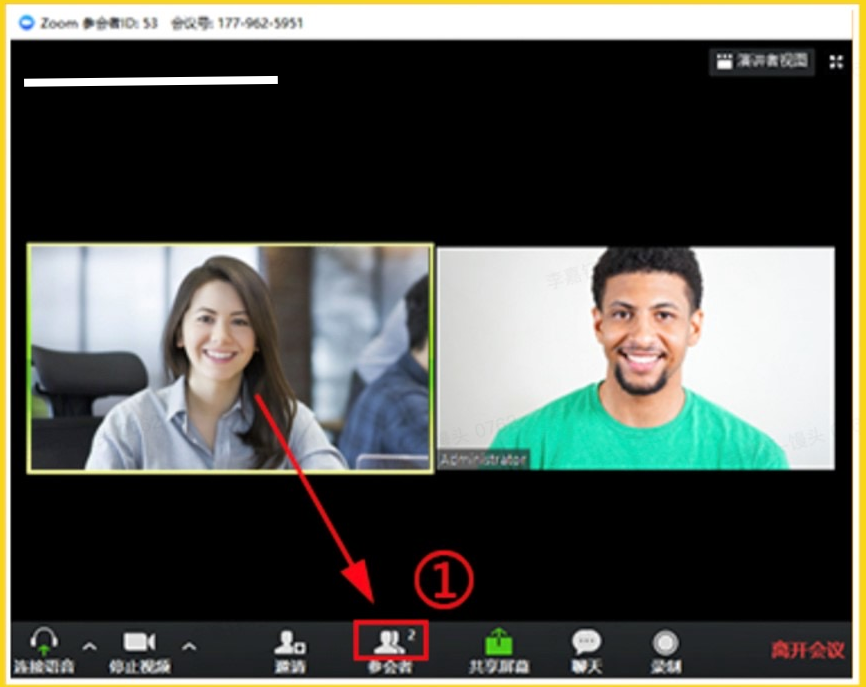


TUTOR Hilbert

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. OOP and Classes
2. Iteration Revisit
3. Error Handling
4. Random
5. `enumerate()` and `zip()` *
6. Unpack Operation *





知识点讲解

考点1. Class

HD EDU

- A class (类) is the **blueprint** of a category of objects (instances).

- 如何判断两个Objects是否同类：
  - 有相同的**属性**(Attributes)
  - 有相同的**行为**(Methods)

- 我们之前一直使用的都是Python pre-defined的classes，而现在使用keyword `class`，我们可以DIY自己的Datatype!

考点1. Class Syntax

HD EDU

```
class MyDataType: # Class Name
  # Constructor Method
  def __init__(self, arg1, arg2):
    # Object Attributes Definition
    self.attr1 = arg1
    self.attr2 = arg2
    self.attr3 = <some default values>

  # Instance Method
  def method(self, arg1):
    # Access Object Attributes
    temp = self.attr1 + self.attr2

    # Modify Object Attributes
    self.attr3 = temp * 2
    return temp

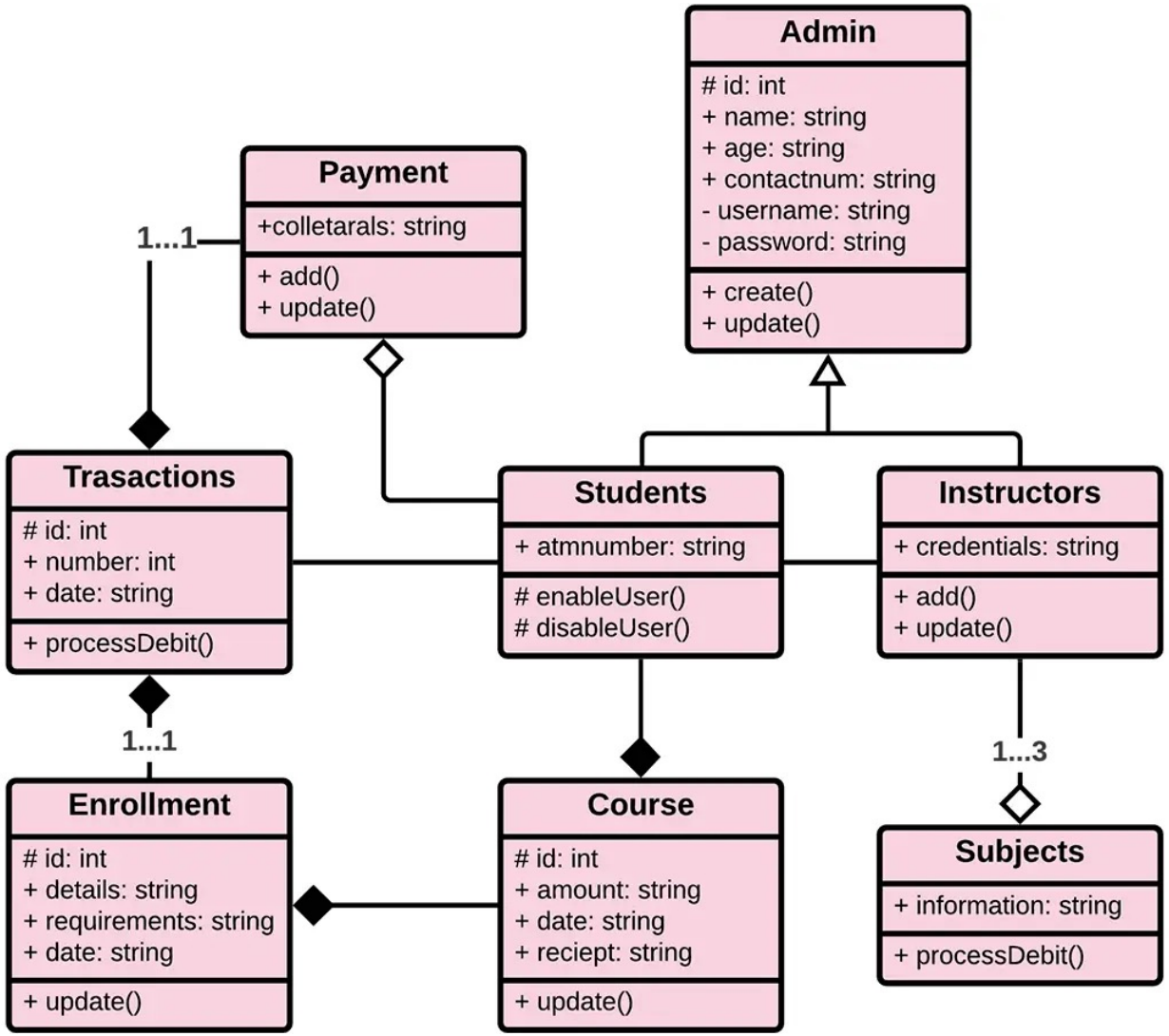
  # Instance Method
  def method2(self):
    # Call Other Methods
    self.method()

  # Static Method
  def static_method(arg1, arg2):
    <some action>
    return <something>
```

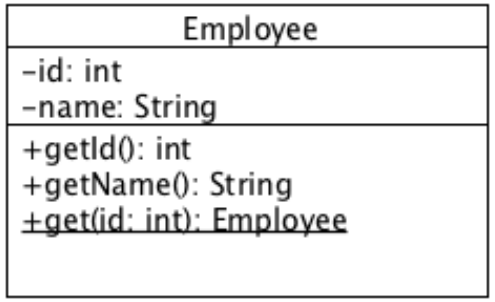
- Constructor describes how the object will be instantiated.
  - This method is usually the first method defined in a class spec.
- Instance methods
- Also known as class method (**Methods belongs to the class not instance**)
- Can be called without creating an instance of the class.

```
a = MyDataType(value1, value2)
```

考点1. Class Diagram

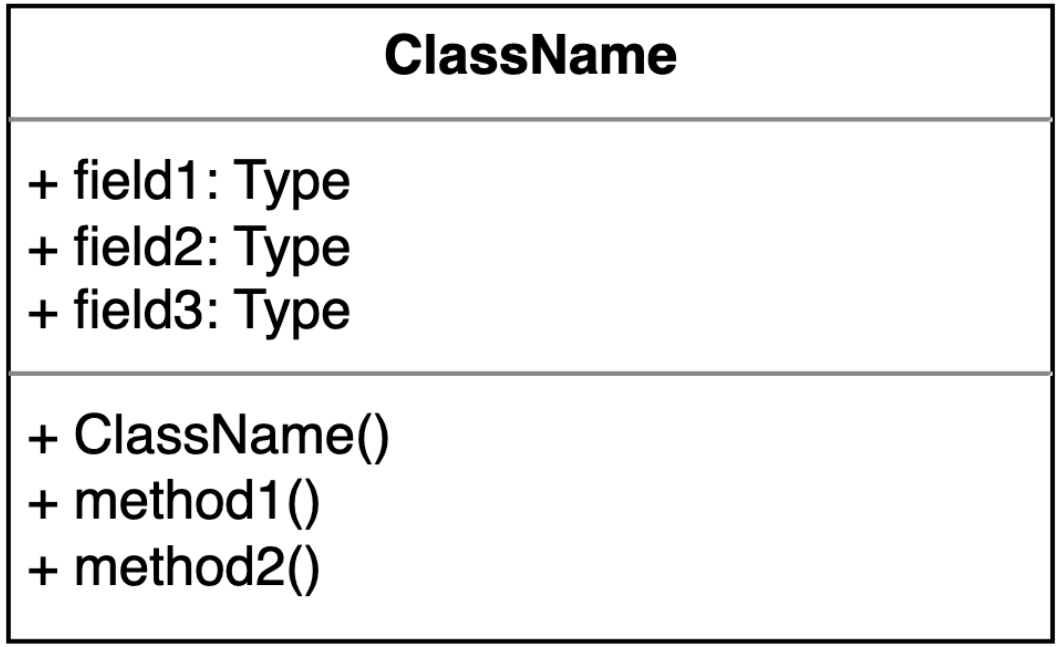


- 我们目前所学的仅仅是一个Class的diagram
- It is a part of **Unified Modeling Language** (UML)，用来描述class与class之间作用关系的diagram
- Static content is indicated using underline



考点1. Class Diagram

HD EDU





Just like Flow Charts: You should be able to convert from Class Diagram to Code and vice versa

- Diagram head部分写Class Name
- 每一个attribute和method前先加一个符号代表该项的Accessability
  - +: Public（对1110而言，永远是+）
  - -: Private
  - #: Protected
- 第一个Method一定是Constructor!

考点1. Getters and Setters

HD EDU

```python
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def get_name(self):
    return self.name

  def set_name(self, name):
    self.name = name

  def get_age(self):
    return self.age

  def set_age(self, age):
    if age >= 0:
      self.age = age
    else:
      print("Bad Age!")
```

- Getter are methods to **retrieve** attribute values.
- Setters are methods that **update** corresponding attribute values.
- It is generally a bad practice to access and modify attributes DIRECTLY:

```python
a = Person("Hilbert", 24)
a.name = "Gilbert"
a.get_name() = "Gilbert" # ERROR
a.age = -100
a.set_age(-100) # Bad Age! and do nothing
```

- 直接访问和修改很方便，但是也会有**安全性**的问题
- 对于一些attribute我们可能会有更复杂的value update逻辑，直接assign的话，无法emforce。

考点1. Static (Class) Variables and Methods

HD EDU

- Static variables belong directly to the class:

```
class Circle:
    pi = 3.1415926

    def __init__(self, r):
        self.r = r

    def area(self):
        return self.r ** 2 * Circle.pi
```

- And can be accessed via the class name

```
>>> print(Circle.pi)
>>> 3.1415926
```

- Static methods also belong directly to the class:

```
class Car:
    def __init__(self, name):
        self.name = name
        self.consumption = 20 # kilo per litre

    def calculate_distance(self, litres):
        return litres * self.consumption

    def kilo_to_mile(kilo):
        return kilo * 0.621371

a = Car("Honda")
kilo = a.calculate_distance(20)
miles = Car.kilo_to_mile(kilo)
```

- One rule-of-thumb: ask yourself "Does it make sense to call this method, even if no object has been constructed yet?" If so, it should definitely be static.

Ref. https://stackoverflow.com/questions/2671496/when-to-use-static-methods

考点1. Some Comparisons

HD EDU

- Class vs. Instance
  - A class is a blueprint for creating objects (instances). It defines the attributes and behaviors that the objects of the class will have. Think of a class as a template or a set of instructions for creating objects.
  - An instance is a specific realization of a class. It is an actual object created (instantiated) from the blueprint defined by the class. Each instance has its own unique set of data and can behave independently of other instances, although they all follow the structure defined by the class.

考点1. Some Comparisons

HD EDU

- Procedural Programming vs. Object-oriented Programming
  - Procedural Programming (基本单位是**Function**)
    - 解决问题的**逻辑**和**流程**
    - Functionalise and reuse some just general program-solving patterns
    - Control flow
  - Object-oriented Programming (OOP) （基本单位是**Class**）
    - Model a complex system’s interaction between different modules
    - Modularity and extensibility
    - Inheritance, polymorphism, and encapsulation for code organization and reuse (*)

考点1. Some Comparisons

HD EDU

❖ Functions vs. Methods

Functions are standalone blocks of code that perform a specific task, while methods are functions associated with objects in object-oriented programming and operate on the data within those objects. The choice between using functions and methods depends on the programming paradigm being used and the context in which the code is being developed.

考点1. Equality

HD EDU

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def get_name(self):
        return self.name

    def set_name(self, name):
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, age):
        if age >= 0:
            self.age = age
        else:
            print("Bad Age!")
```

- By default, the objects defined are compared based on **memory addresses**.

```
>>> a = Person("A", 10)
>>> b = Person("A", 10)
>>> print(a == b)
False
>>> print(a)
<temp.Person object at 0x104ec07a0>
>>> print(b)
<temp.Person object at 0x104ec23f0>
>>> c = a
>>> print(a == c)
True
```

考点1. Equality

HD EDU

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def get_name(self):
        return self.name

    def set_name(self, name):
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, age):
        if age >= 0:
            self.age = age
        else:
            print("Bad Age!")
    def __eq__(self, person):
        if self.get_name() == person.get_name():
            if self.get_age() == person.get_age():
                return True

        return False
```

- By defining __eq__( ) method, we can freely design the equality for our own class:

```
>>> a = Person("A", 10)
>>> b = Person("A", 10)
>>> print(a == b)
True
>>> b.set_age(11)
>>> print(a == b)
False>>> a = Person("A", 10)
>>> b = Person("A", 10)
>>> print(a == b)
True
>>> b.set_age(11)
>>> print(a == b)
False
```

考点1. String and Representation

HD EDU

```
>>> a = Person("A", 10)
>>> b = Person("A", 10)
>>> print(a)
<temp.Person object at 0x104ec07a0>
>>> print(b)
<temp.Person object at 0x104ec23f0>
```

- When we try to print our own defined objects, the message follows Python's default template: <"class" object at "memory">
- But again, we can define __str__( ) or __repr__( ) to overwrite the default behaviour.

```
>>> a = Person("Hilbert", 24)
>>> print(a)
Person(Hilbert, 24)
>>> print(str(a))
Person(Hilbert, 24)
```

考点1. str() and repr() *

HD EDU

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def get_name(self):
        return self.name

    def set_name(self, name):
        self.name = name

    def get_age(self):
        return self.age

    def set_age(self, age):
        if age >= 0:
            self.age = age
        else:
            print("Bad Age!")

    def __str__(self):
        return "Person({}, {})".format(self.name, self.age)

    def __repr__(self):
        return "Person"
```

- Usually define either `__str__()` or `__repr__()` will do the job.
- But if we have them both, `__str__()` is prioritized.
- Deep down inside, they just define the bahaviours called by `str()` and `repr()` function.
- `print()` function implicitly involves those functions to get the printing content.

```
>>> a = Person("Hilbert", 24)
>>> print(a)
Person(Hilbert, 24)
>>> print(str(a))
Person(Hilbert, 24)
>>> print(repr(a))
Person
```

考点1. str() and repr() *

HD EDU

- `__str__()`: This method is called by the `str()` built-in function and by the print() function to obtain the "informal" or user-friendly string representation of an object. It is meant to return a human-readable string that describes the object. This method is often used for logging or displaying objects to end-users.

- `__repr__()`: This method is called by the `repr()` built-in function to obtain the "official" or unambiguous string representation of an object. It should return a string that, when passed to the eval() function, would create a new object with the same value. This method is typically used for debugging and development purposes.

考点2. Iteration Revisit

HD EDU

- `break`: **terminate** the iteration
- `continue`: **skip** the current iteration

```
i = 0

while i < 10:
  print(i)
  if i == 5:
    break
  i += 1
```

```
i = 0

while i < 10:
  print(i)
  if i == 5:
    continue
  i += 1
```

考点2. break and continue

HD EDU

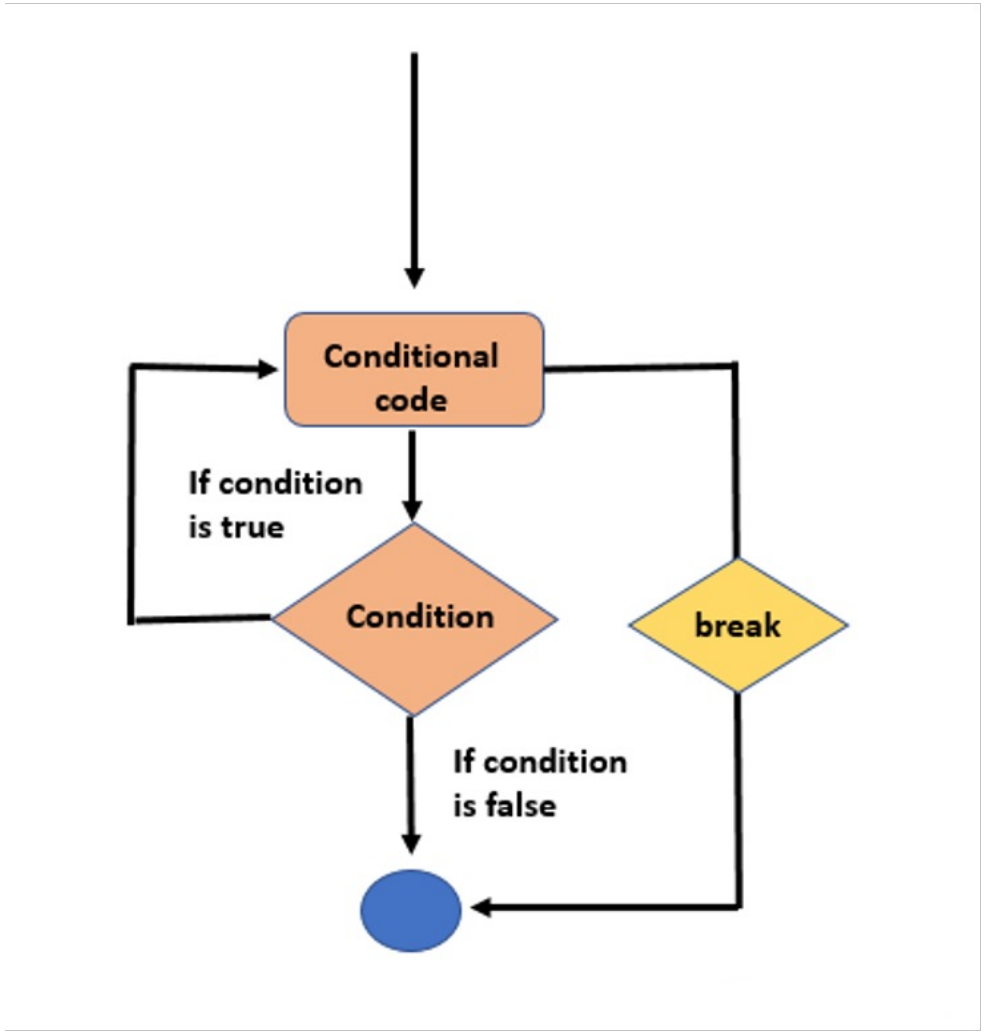
- Just like `return`, any other code under them will be ignored.
- In nested loops, the keywords will only perform the action to the loop with corresponding indentation level.
- In function definition, `return` can also terminate. What is the difference?
- How to draw the flow charts of code having `break` and `continue`?

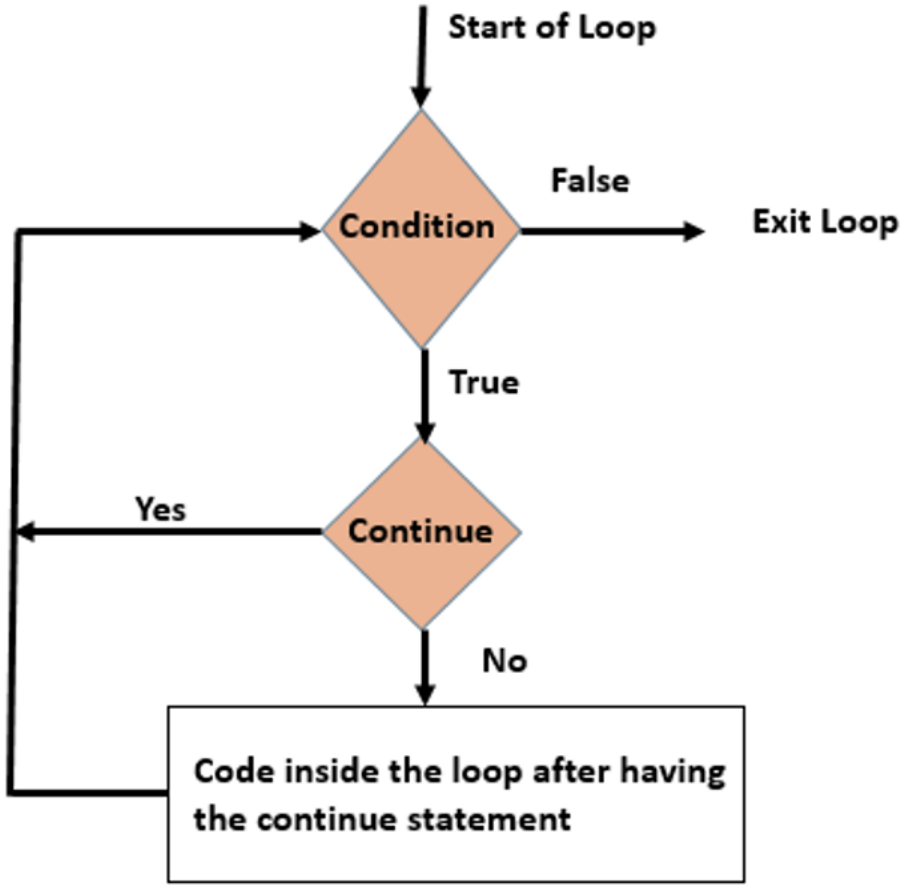
```
def f():
  i = 0
  while i < 10:
    if i == 5:
      break
    print(i)
    i += 1
  print("Hello")
```

```
def f():
  i = 0
  while i < 10:
    if i == 5:
      return
    print(i)
    i += 1
  print("Hello")
```

考点2. break and continue

HD EDU





考点2. Iteration Revisit

HD EDU

- 现在有了break和continue，在event-controlled loop里我们就不一定需要一个flag variable来实现了。

```
stop = True
help = '''Command List
EXIT --- Shutdown the program
HELP --- Get the list of all available commands
ECHO --- Print the given message to the screen'''

while stop:
    command = input(">>>")

    if command == "EXIT":
        stop = False
    elif command == "HELP":
        print(help)
    elif command.startswith("ECHO "):
        print(command[5:])
    else:
        print(f"Command {command} not found.")
```

```
stop = True
help = '''Command List
EXIT --- Shutdown the program
HELP --- Get the list of all available commands
ECHO --- Print the given message to the screen'''

while True:
    command = input(">>>")

    if command == "EXIT":
        break
    elif command == "HELP":
        print(help)
    elif command.startswith("ECHO "):
        print(command[5:])
    else:
        print(f"Command {command} not found.")
```

考点2. for loop

HD EDU

- Index based iteration
  - Range包含起点不包含终点(同slicing)

```
for i in range(start, end, step):
  print(i)

i = start
while i < end:
  print(i)
  i += step
```

- Assignment，Final大概率无法使用

- For-each loop

```
For <variable> in collection:
    print(variable)
```

```
>>> l = [1, 3, 5, True, -1.3]
>>> for e in l:
...     print(e)
...
1
3
5
True
-1.3
>>> for c in "abcdefg":
...     print(c)
...
a
b
c
d
e
f
g
```

考点2. for loop - drawback

HD EDU

❖ 以下程序运行后会输出什么?

```
l = [1, 3, 5, True, -1.3]
for e in l:
  e = 2
print(l)

i = 0
while i < len(l):
  l[i] = 2
  i += 1
print(l)
```

考点2. for loop - drawback

HD EDU

❖ 以下程序运行后会输出什么?

```
l = [1, 3, 5, True, -1.3]
for e in l:
  e = 2
print(l)

i = 0
while i < len(l):
  l[i] = 2
  i += 1
print(l)
```

```
l = [1, 3, 5, True, -1.3]
for e in l:
  e = 2
print(l)

i = 0
while i < len(l):
  l[i] = 2
  i += 1
print(l)

for i in range(len(l)):
  l[i] = 2
print(l)
```

考点3. Error Handling - raise

HD EDU

- Sometimes our program would run into some situations that it makes no sense to continue execution. However, it also should not be no differece between normal program termination.
  - We raise (throw) an error and let the program crashes!

```
def divide(a, b):
  # 除数为0
  if b == 0:
    # What should we do?
  ret = a / b
  return ret
```

```
>>> raise ZeroDivisionError
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError
>>> raise ZeroDivisionError("b cannot be zero!")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: b cannot be zero!
```

考点3. Error Handling – Error Catching

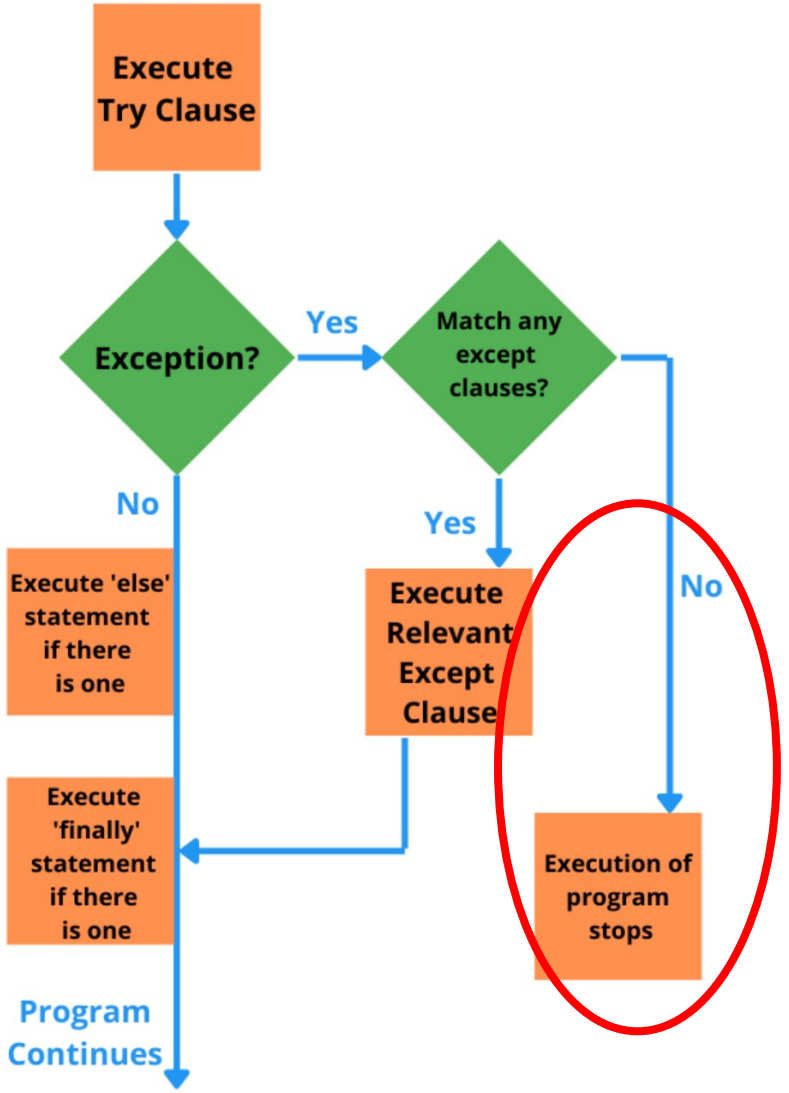
HD EDU

- Sometimes we may suspect some part of our code might run into errors, when this does happen we want to handle it specifically rather than just letting it crashes and ends the whole program.

```
try:
  <Code>
  <With>
  <Potential>
  <Error>
except:
  # When the error does occur, do...
else:
  # When the error doesn't occur, do
finally:
  # When the handling finishes, all do...
```

```
def divide(a, b):
  try:
    ret = a / b
  except ZeroDivisionError:
    print("b cannot be zero!")
  return ret
```

考点3. Error Handling – Error Catching



❖ Didn’ t catch the error

考点3. Error Handling – Exception Hierarchy (*)

HD EDU

```
BaseException
 ├── BaseExceptionGroup
 ├── GeneratorExit
 ├── KeyboardInterrupt
 ├── SystemExit
 └── Exception
      ├── ArithmeticError
      │    ├── FloatingPointError
      │    ├── OverflowError
      │    └── ZeroDivisionError
      ├── AssertionError
      ├── AttributeError
      ├── BufferError
      ├── EOFError
      ├── ExceptionGroup [BaseExceptionGroup]
      ├── ImportError
      │    └── ModuleNotFoundError
      ├── LookupError
      │    ├── IndexError
      │    └── KeyError
      ├── MemoryError
      ├── NameError
      │    └── UnboundLocalError
      ├── OSError
      │    ├── BlockingIOError
      │    ├── ChildProcessError
```

- Inheritence (*)
- E.g., Car
  - Tesla
  - Truck
  - Etc.

```
>>> try:
...     a = 1
...     b = 0
...     c = a / b
... except BaseException:
...     print("Caught the zero division error!")
...
Caught the zero division error!
```

- The `ZeroDivisionError` was caught because it inherited from BaseException

Ref.: https://docs.python.org/3/library/exceptions.html#exception-hierarchy

考点3. Only do Error Handling when you must

HD EDU

```
>>> a = "1"
>>> a.isdigit()
True
>>> try:
...     int(a)
... except ValueError:
...     print("False")
... else:
...     print("True")
...
True
```

```
a = "1"
b = "2"

try:
  c = a * b
except:
  print("Fail")

# ---------------
a = "1"
b = "2"

if type(a) != int:
  print("Fail")
if type(b) != int:
  print("Fail")
c = a * b
```

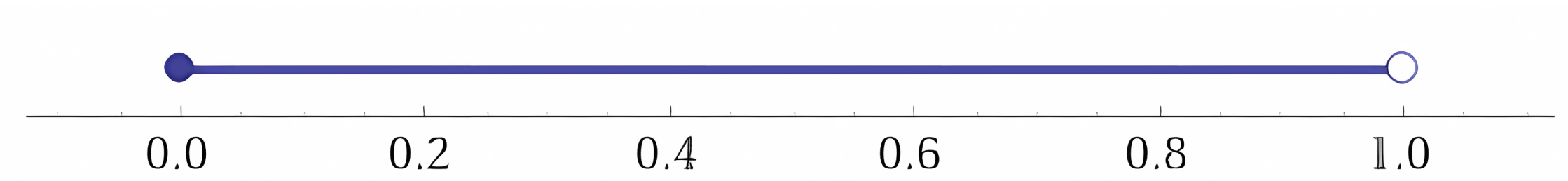
- Error Catching是最后的最后的手段，能用正常control flow解决的问题就不要用try-except

考点4. Random

HD EDU

`random.random()`

Return the next random floating point number in the range [0.0, 1.0).



- 需要多少的概率，就把这段范围多少等分

`random.randint(a, b)`

Return a random integer *N* such that `a <= N <= b`. Alias for `randrange(a, b+1)`.

- 注意random.randint(a, b)的随机范围是既包含a也包含b的([a,b])

Ref.: https://docs.python.org/3.10/library/random.html

考点5. enumerate( ) and zip( ) *

HD EDU

❖ Usually use along with for loops.

```
>>> a = [1, 2, 3, 4]
>>> b = ['a', 'b', 'c']
>>> zip(a, b)
<zip object at 0x1011340c0>
>>> for e in zip(a, b):
...     print(e)
...
(1, 'a')
(2, 'b')
(3, 'c')
```

```
>>> l = [1, -9.99, False, "abc", None, (2, 2)]
>>> print(enumerate(l))
<enumerate object at 0x101014810>
>>> for e in enumerate(l):
...     print(e)
...
(0, 1)
(1, -9.99)
(2, False)
(3, 'abc')
(4, None)
(5, (2, 2))
>>> for i, e in enumerate(l):
...     print(i, e)
...
0 1
1 -9.99
2 False
3 abc
4 None
5 (2, 2)
```

考点6. Unpack *

HD EDU

- Unpacking in Python refers to the process of extracting elements from collections (like lists, tuples, or dictionaries) and assigning them to individual variables.
- It allows you to easily assign multiple values contained within collection to multiple variables in a single statement.

```
my_tuple = (1, 2, 3)
a, b, c = my_tuple

my_list = [4, 5, 6]
x, y, z = my_list

my_list = [1, 2, 3, 4, 5]
first, *rest, last = my_list
# rest will be a list

def my_function(a, b, c):
    print(a, b, c)

my_tuple = (1, 2, 3)
my_function(*my_tuple)

def f():
  return 1, 2
a, b = f()

# Swap values between two variables
a, b = b, a
```

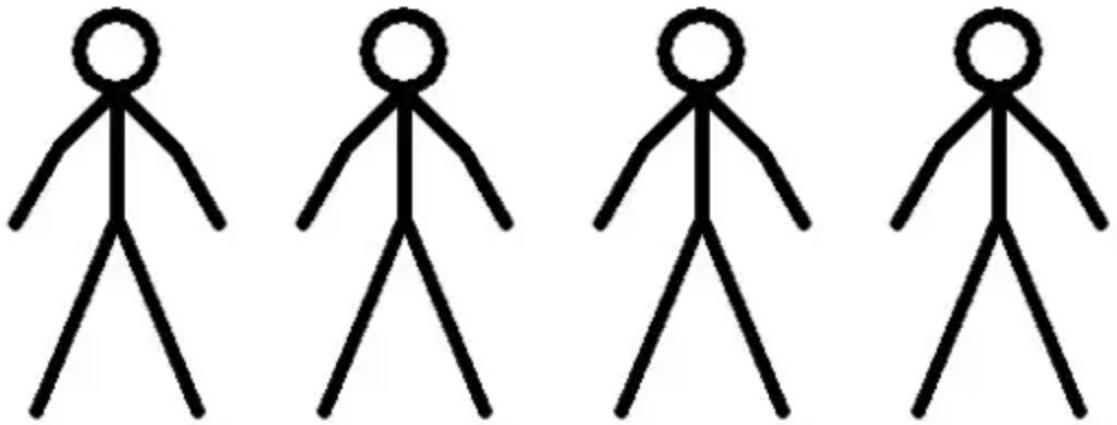


经典题型放送

Classes Communication Example

(自编题)





Classes Communication Example

HD EDU

```python
class Car:
    def __init__(self, name, capacity):
        self.name = name
        self.capacity = capacity
        # Array
        self.seats = []
        i = 0
        while i < self.capacity:
            self.seats.append(None)
            i += 1

    def count(self):
        num = 0
        i = 0
        while i < self.capacity:
            if self.seats[i] is not None:
                num += 1
            i += 1
        return num

    def pickup(self, person):
        if self.count() < self.capacity:
            i = 0
            while i < self.capacity:
                if self.seats[i] is None:
                    self.seats[i] = person
                    return True
                i += 1
        else:
            return False

    def kickout(self, person):
        i = 0
        while i < self.capacity:
            temp = self.seats[i]

            if person == temp:
                self.seats[i] = None
                return
            i += 1
```

```python
class Person:
    def __init__(self, name, is_happy, age):
        self.name = name
        self.is_happy = is_happy
        self.age = age

    def __repr__(self):
        return self.name

car = Car("五菱宏光", 3)
p1 = Person("Ada", True, 12)
p2 = Person("Bill", True, 25)
p3 = Person("Cherry", False, 12)
p4 = Person("David", False, 12)
print("Start with an empty car", car.count())
print("Add Ada", car.pickup(p1))
print("Count", car.count())
print("Add Bill", car.pickup(p2))
print("Count", car.count())
print("Add Cherry", car.pickup(p3))
print("Count", car.count())
print("Add David", car.pickup(p4))
print("Count", car.count())
print("Car seats:", car.seats)
```

Random Password Generator

HD EDU

(自编题)

Write a function that outputs a randomized password base on two parameters `char` and `length`.

`char`: A list of characters where it contains all the possible characters to be used in the password string generated.

`length`: the length of the password.

```
>>>generate_password(['a','b'], 4)
'aabb'
```

Random Password Generator

HD EDU

```
import random

def generate_password(chars, length):
  pass

chars = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k',
'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
'L', 'M', 'N', 'O', 'P', 'Q', 'S', 'T', 'U', 'V', 'W', 'X', 'Y',
'Z', '!', '@', '#', '$', '%', '^', '&', '*']

print(generate_password(chars, 8))
print(generate_password(chars, 16))
print(generate_password(chars, 32))
```

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
```

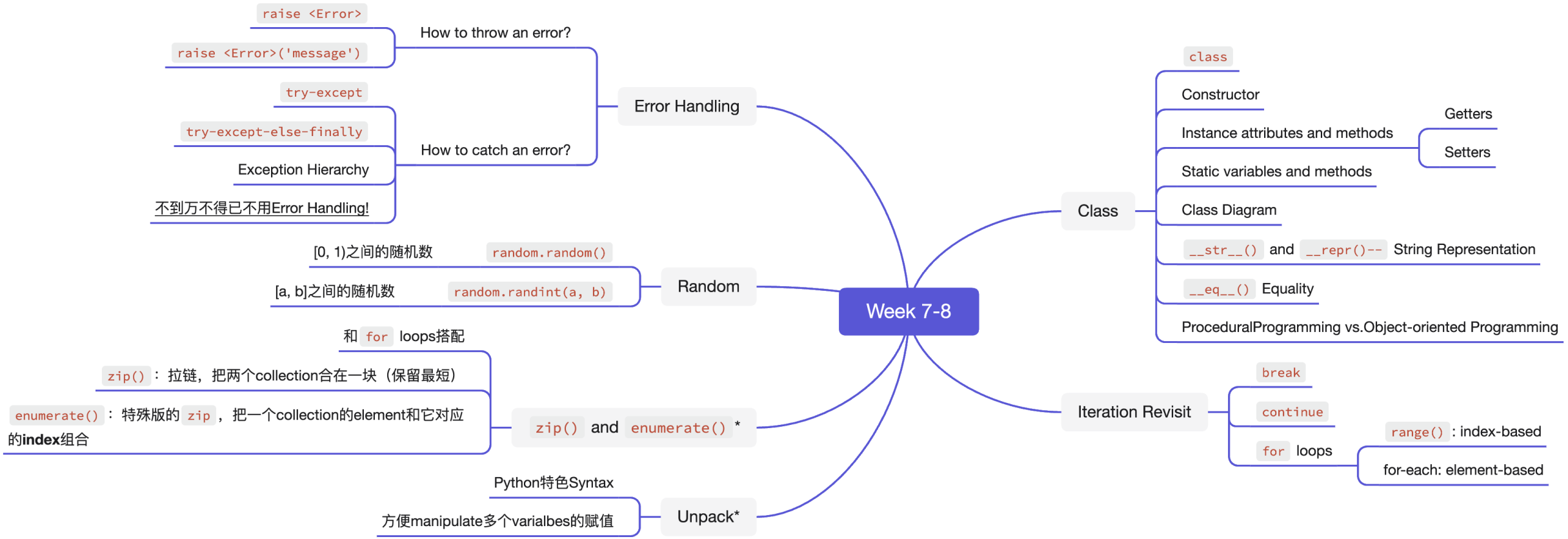
Descending order?





重难点总结

重难点总结


Week 7-8
Error Handling
How to throw an error?
raise <Error>
raise <Error>('message')
How to catch an error?
try-except
try-except-else-finally
Exception Hierarchy
不到万不得已不用Error Handling!
Random
[0, 1)之间的随机数
random.random()
[a, b]之间的随机数
random.randint(a, b)
zip() and enumerate()*
和 for loops搭配
zip()：拉链，把两个collection合在一块（保留最短）
enumerate()：特殊版的 zip，把一个collection的element和它对应的index组合
Unpack*
Python特色Syntax
方便manipulate多个varialbes的赋值
Class
class
Constructor
Instance attributes and methods
Getters
Setters
Static variables and methods
Class Diagram
__str__() and __repr()-- String Representation
__eq__() Equality
ProceduralProgramming vs.Object-oriented Programming
Iteration Revisit
break
continue
for loops
range() : index-based
for-each: element-based



下节课预告

下节课预告

HD EDU

- Testing
  - Black box vs White box testing
  - Unit Testing
  - Edge cases / Corner cases
- 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



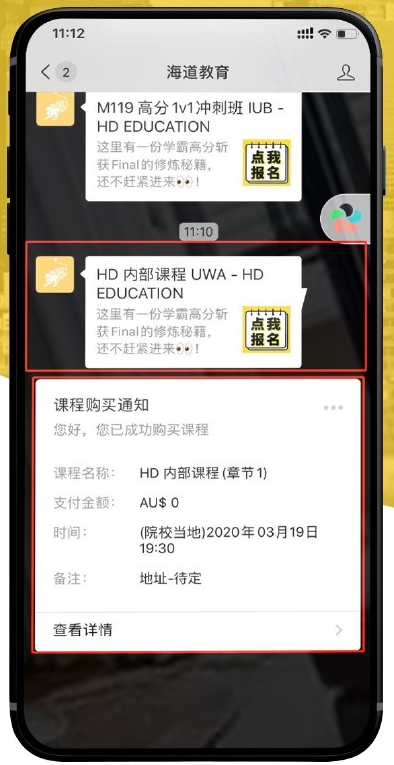
帮我开个张吧，求求你了！



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

HD EDU



Q & A