Lecture 1

A computer contains 3 main hardware components:

input device / processing unit / output device

the brain of compu is Central Processing Unit { Arithmetic and Logic Unit (ALU): Performs arithmetics like a calculator
Control Unit (CU): Directs the operations of the processor in executing a program

Both machine and assembly language are low-level languages.

① difficult to write for complicated tasks (requiring many lines of code)

② platform-specific: the sets of instructions and their binary codes can be different for different types of CPUs
different operating systems use different assembly languages/styles.

Programming: AKA software development, is the process of writing program for the computer to execute different tasks.

machine languages written in binary sequences is known as 1st generation Programming language.

assembly is 2nd high-level is 3rd

{ A program written in high-level gets converted automatically to a low-level machine language code for the desired platform.
This abstracts away low-level details that can be handled by the computer automatically.

{ Compilation means converting a program well before executing of the program : C++, Java
Interpretation means converting a program on-the-fly during the execution of a program : JavaScript, Python.

lecture 2

\n creates a new line when printing str

\t creates a tab

Errors:

Python interpreter can detect syntax error even before executing the code

TypeError: Like str + int. Python interpreter can only detect a type error at runtime.

Python is { strongly-typed: Python does not force a type conversion to avoid a type error
Dynamically-typed: Python checks data type only at runtime after translating the code to machine code.

float corresponds to the floating point representation.

Python stores a floating number with finite precision, usually 64-bit/double precision.

Why 2.675 rounds to 2.67 instead of 2.68? { A float is actually represented in binary
A decimal may not be represented exactly in binary

{ right associativity: ** , −
left associativity: * , / , // , % , + , −

Lecture 3

Condition

Conditional execution means running different pieces of code based on different conditions.

All the comparision operators have the same precedence lower than that of + and -

The expression x or y first evaluates x. If x is True, then its value is returned.

python uses indentation to indicate code blocks or suite.

to leave the block empty, Python uses pass

Iteration.

In python, str regarded as a sequence type.

python provides While statement to loop until a specified condition is false.

- for loop is a definite loop which has a defined number or iterations before the execution of the loop
- while statement is useful for an indefinite loop where the number of iterations is unknown before the execution of the loop.

Break:

pass:

continue: If the condition is satisfied, the continue statement will skip to the next iteration.

Lecture 4

using Function:

code reuse gives the code an elegant structure { can be excuted efficiently by computer / interpreted easily by a programmer.

A python function may have side effects and return the value None.

{ the argument can also be a function call like function composition in mathematics.
Before a function call is executed, its argument are evaluated first from left to right.

the function from library is efficently implemented and thoroughly tested/documented.

What's function in programming?

A function is simply a 'chunk' of code that you can use over and over again, rather than writing it out multiple times. Function enables programmers to break down or decompose a problem into smaller chunks, each of which perfams a particular tasks. Once a function is created, the details of how it works can almost be forgotten about. In this way, the detail is abstracted, allowing the programmer to focus on the bigger picture.

Also, the import statement polluted the namespace of global frame and caused a name collision.

writing Function

A function is defined using def

#. Those comments are not usage guide, they're intended for programmers who need to maintain/extend the function definition

Lecture 1

Python is a class-based object-oriented programming (OOP) language

- Each object is an instance of a class/type, which can be a subclass of one or more base classes.
- An object is a collection of members/attributes, each of which is an object.

OOP 封装 encapsulates implementation details while.

Almost everything in Python is an instance of type-object.

Python treats functions as first-class objects that can be
- passed as argument to other functions
- assigned to variables
- returned as values.

an object is a type, a type is an object

an object can be an instance of more than one types.

bool $\xrightarrow{\text{subclass}}$ int $\xrightarrow{\text{subclass}}$ object

type $\xrightarrow{\text{subclass}}$ object

the structure and behavior of an object is governed by its attributes.

different objects of a class have the same set of attributes as that of the class.

A subclass also inherits the attributes of its base classes.

Different objects of the same class can still behave differently because their attribute values are different

An attribute can also be a function, which is called a method or member function

A method can be accessed by objects of the class.

open is a function that creates a file and assigns it to f (f = open("....csv"))
- read returns the entire content of the file as a string.
- close flushes and closes the file.

open(xx, "w") as ..._file . xx_file.write(...)	argument 'w' for open sets the file object to write mode. method write writes the input string to the file.

What is overloading?

① Having an operator perform differently based on its argument types is called operator overloading

② + is called a generic operator

③ we can also have function overloading to create genetic function

The strategy of checking the type for the appropriate implementation is called dispatching on type.

The OOP techniques involved are formally called:

Polymorphism (多态性): Different types can have different implementations of the same method such as __add__.

Single dispatch: The implementation is chosen based on one single type at a time, + calls __add__ of the first oprand, and if not properly implemented for the second operand type, __radd__ of the second operand.

A method with starting and trailing double underscores in its name is called a dunder method.

Class:

the __init__ method of the class is called with the new object as its first argument (self), along with any additional arguments provided in the call expression.

```
class Account:                          # constructor
    def __init__(self, account_holder):
        self.balance = 0
        self.holder = account_holder

    def deposit(self, amount):          # Methods are defined in the suite of a class statement
        self.balance = self.balance + amount
        return self.balance
>>> tom = Account('Tom')
>>> tom.deposit(100)
100
```

dot Expression

Invoked with one argument

Objects receive message via dot notation. Dot notation accesses attributes of the instance or its class.

<expression> . <name>

Python distinguishes between:

1. Functions, which we have been creating since the beginning of the course.
2. Bound methods, which couple together a function and the object on which that method'll be invoked.

Object + Function = Bound Method.

Assignment statements with a dot expression on their left-hand side affect attributes for the object of that dot expression.

- If the object is an instance, then assignment sets an instance attribute
- If the object is a class, then assignment sets a class attribute

```
class Account:
    interest = 0.02
    def __init__(self, holder):
        self.holder = holder
        self.balance = 0
```

Instance Attribute assignment:

```
tom_account.interest = 0.08
```

Class Attribute assignment:

```
Account.interest = 0.04
```

```
tom_account = Account('Tom')
```

Inheritance:

Inheritance is a method for relating classes together.

A common use: Two similar classes differ in their degree of specialization.

The specialized class may have the same attributes as the general class, along with some special-case behavior

```
class <name>(<base class>):
    <suite>
```

- new class shares attributes with its base class.
- the subclass may override certain inherited attributes.

```
class Checking(Account):
    withdraw_fee = 1
    interest = 0.01
    def withdraw(self, amount):
        return Account.withdraw(self, amount + self.withdraw_fee)
```

Base class attributes aren't copied into subclasses.

To look up a name in a class. ① If it names an attribute in the class, return the attribute value.

② Otherwise, look up the name in the base class. if there's one.

Lab 5:

urlencode creates a string with some special symbols encoded. { ':' is replaced by %3A; '/' is replaced by %2F }

DataFrame object: { the content of the CSV file is displayed as an HTML table conveniently; we can control how much information to show by setting the display option }

lecture 6

Recursion : a function that calls itself (recurs) is recursion

gcd 's Euclidean algorithm for the greatest common divisor.

$$gcd(a,b) = \begin{cases} a, & b = 0 \\ gcd(b, a\%b), & \text{otherwise.} \end{cases}$$

recursion can be replaced by iteration with an explicit call stack, while iteration can be replaced with tail recursion.

benefits of recursion: recursion is often shorter and easier to understand

It can be easier to write code by wishful thinking or declarative programming as supposed to imperative programming.

Rules for global/local variables:

- A local variable must be defined within a function.
- An assignment defines a local variable except after a global statement

UnboundLocalError: local variable referenced before assignment.

using global variable : { codes are less predictable, more difficult to reuse/extend, and text cannot be isolated, making debugging difficult.

- Local function can access (capture) the other local variables by forming the so-called *closures*.
- Similar to the global statement, a nonlocal statement is needed for non-local variables.
- Each local function has an attribute named __closure__ stores the captured local variables.

Due to redundant computations. A better way to define the generator is to use yield.

yield causes the function to return a *generator* without executing the function body.

Calling _next_ resumes the execution, which:

- pauses at the subsequent yield expression, if any, or
- raises the StopIterationException at the end otherwise.

Decorator :

Arguments with default values specified by = ... are called default argument

Rules for specifying arguments

1. Default (keyword) arguments must be after all non-default (positional) arguments in a function definition (call)
2. The value for an argument cannot be specified more than once in a function definition (call)

range is indeed NOT a generator

`type(range) == 'type'` `type(range(10)) == 'range'`

Variable number of arguments

args is a tuple of positional arguments

kwargs is a dictionary of keyword arguments, which is a list of values indexed by unique keys that are not necessary integers.

* and ** are unpacking operators for tuple/list and dictionary respectively.

```
def argument_string(*args, **kwargs):
    return "({})".format(
        ", ".join(
            [
                *(f"{v!r}" for v in args),
                *(f"{k}={v!r}" for k, v in kwargs.items()),
            ]
        )
    )
```

!r convert v to the string representation (repr) that can be evaluated by python eval in particular, 'a' will be converted to "'a'", which has the quotation needed for the string literal

the decoration @functools.wraps(f) of the decorated function wrapper

① makes some attributes (such as __name__) of the decorated function the same as those of original function, and

② adds some useful attributes such as __wrapped__ that points to the original function.

we can use a decorator to make recursion more efficient by caching the return values. cache is a dictionary where cache[n] stores the computed value of F_n to avoid redundant computations.

Send method

```
>>> g = xx (generator)
>>> next(g)
>>> g.send(n)
```

n将会被传递 (类似 ans = yield sth 这种形式)

Lecture 7

- List is mutable so programmers can change items.
- Tuple is immutable like int/float/str.
  - so programmer can be certain the content stay unchanged.
  - python can preallocate a fixed amount of memory to store its content.

If the enclosed sequence has one term, there must be a comma after the term (0,)

The elements of a tuple can have different type

The unpacking operator * can unpack an iterable into a sequence in an enclose (a_tuple = (lambda *args : args)(1,2,3)

(*range(0,2))

zip(*iterables, strict=False) --> Yield tuples until an input is exhausted.

```
>>> list(zip('abcdefg', range(3), range(4)))
[('a',0,0), ('b',1,1), ('c',2,2)]
```

Sequence objects such as str/tuple/list implements the *getter method* __getitem__ to return their items.

Lecture 8

dict has a class method fromkeys . dict.fromkeys(iterable, value=None, /)

A key for dict or set must be a hashable object which.

- has a hash value (returned by __hash__ method) that never changes during its lifetime.
- can be compared (using __eq__ method) to other objects.

Hashable objects which are equal must have the same hash value.

set is not ordered. dict is insertion-ordered.

set has no implement __getitem__ and is therefore not subscriptable.

the function sorted. to a set/dic to return a sorted list of keys.

| : 并集　　& 交集　　^ . 交集的补集

class method	object method	operator
union	update	|
intersection	intersection_update	&
symmetric_difference	symmetric_difference_update	^
is subset		<=
issuperset		>=
isdisjoint		

Lab 8 Information Theory.

we denote a distribution as $P = [p_i]_{i \in S}$ $\begin{cases} S \text{ is the set of distinct outcomes} \\ p_i \text{ denotes the possibility of seeing outcome } i \end{cases}$

definition: $H(p) := \sum_{i \in S} p_i \underbrace{\log_b \frac{1}{p_i}}_{\text{called surprise}} = -\sum_{i \in S} p_i \log_b p_i$ (bits)

with $p_i \log_b \frac{1}{p_i} = 0$ if $p_i = 0$ ($\lim_{x \to 0} x \log_b \frac{1}{x}$)

Lecture 9 Monte Carlo Simulation and Linear Algebra.

every numpy array has a data type: integer: int8 int16 int32 ... /float: float16 float32 ... /complex: complex64, complex128/

boolean: bool8 / Unicode, string / Object, object.

String representation.

```
>>> 12e12
1200...
>>> print(repr(12e12))
1200...
```

{ the result of calling repr on a value is what Python prints in an interactive session.
Some objects do not have a simple Python-readable string.

Interface

Message passing: Objects interact by looking up attributes on each other (passing messages)

Special Method Names in Python.

__init__ Method invoked automatically when an object is constructed

__repr__ ~ to display an object as a Python expression.

__add__ to add one object to another

__bool__ to convert an object to True or False

__float__ to convert an object to a float (real number)

Adding instances of user-defined classes invokes either the __add__ or __radd__ method.

若要找到字符串中第一个字母. ① 方法. text.split("_") 按字符形式创建了一个列表.

② text.find( ) 直接找到一个元素.

大写第1字母. text = text[0].upper() + text[1:]
(大写字母)

methods: .endswith()

word.isdigit() 与 word.isalpha() 可验证是字母/数字

list.index(xx) 可显示出xx元素在list中的具体位置.

word.replace("a","b"). 将字符串内所有a换成b.

word.join(list). 如"".join([1,2,3···]) ⇒ "1,2,3,···"

word.split(). 创建列表

python 中关于日期的函数: datetime 模块用于处理日期与时间.

date: 一个理想化的简单日期

time: 对象表示的日期的理想化时间

from datetime import date, timedelta

使用 date(..,..,..) 与 ···.days 转换为日期

函数 map(A, xx) 可以将 xx 作为 A 的实参.

如: def func(n):
    return len(n)
x = map(func, ("apple"···))

对于单个字符串，a="honey" a[::-1] (=) "yenoh" 可反转

利用 sorted 对字典，列表进行排序，在 sorted(..., ..., reverse = True)

word.find(xx)，找到 xx 在 str(word) 中的具体位置

若是 word.find(xx, int)，意为从 word 的第 int 位找

方法 Counter.(from collections import Counter) 计数器.

Counter([1,99,5,4]) == {9:2, 1:1, 5:1, 4:1}

函数 ord( )，返回该字母/数字的 unicode

函数 range(a, b, c) means. range(start, stop[, step])

.total_seconds() 可计算该时间所有的秒数.

python 支持对字符串，如：verb="Hello"，verb[0] == H

字符串相关方法：

str.capitalize() 把字符串第一个字符大写

str.center(with) 返回一个原字符串居中，并使用空格填充至长度width的新字符串.

str.count(str1, begin=.., end=..) 返回范围内str1出现次数.

str.endswith(obj, begin=.., end=..) 检查范围内是否以obj结束. T/F

str.startswith

str.find(str1, begin=.., end=..) 检查是否在范围内，返回索引值，否返回(-1)

str.index(str1, begin=.., end=..) 同上，否报错

is系列
- isalnum() 至少一个，且全字/数返T
- isalpha() 至少一个，且全字，返T
- isdecimal() 全十进制数返T
- isdigit() 全数返T
- islower() 全小写返T
- isnumeric() 只含数字返T
- isspace() 只含空格
- istitle() 是标题化的
- isupper() 全大写

str.join(seq) 以str作为分隔符，将seq所有元素(的字符串表示)合并为一个新字符串. (seq = str.split("str1"))

str.lstrip() 去除左边的空格

str.partition(str1) 从str1出现的第一个位置起，分割成str和之前和之后的元组

str.maketrans(intab, outtab) 用于创建字符映射的转换表，第一个是要转换的字符，第二个是字符串表示转换的目标

str.replace(str1, str2, num=..) 将str1换成str2，不超过num次.

str.rfind(str1, begin=, end=..) 返回最后一次匹配的字符串的索引号

str.rjust(width) 返回一个原字符串右对齐，并使用空格填充至长度width的新字符串

str.split(str=" ", num=_) 以str作为分隔符切片string，仅分隔num+1个子字符串.

str.splitlines([keepends]) 按照行('\r', '\r\n', '\n')分隔.

str.rapritition(str1) 从str1最后一次起，分割成三个元组

set(str)，将字符串内每个字提出来组建个列表(乱序)

str.r[l]just(num, str) 在不长于num的字符串中从str2一侧给str1用余符补为 'ABCD'.rjust(5, "K") == "KABCD"

数据类型转换：

int(x[,base]) 将x转换为一个整数。
float(x) 将x转换为浮点数。
complex(real[,imag]) 创建一个复数
str(x) 转换为字符串
repr(x) 将x转换为表达式字符串
eval(str) 计算在字符串中有效的Python表达式，返回一个对象。
tuple(s) 将序列s转换为一个元组。
list(s) 将序列s转为一个列表
set(s) 转换为可变集合
dict(d) 创建一个字典（d为(key,value)元组序列）
frozenset(s) 转换为不可变集合
chr(x) 将整数转换为字符
ord(x) 将一个字符转换为它的整数值
hex(x) 将一个整数转换为一个十六进制字符串
oct(x) 将一个整数转换为一个八进制字符串。

数学函数：

abs(x) 绝对值

ceil(x) 返回比它大的整数 4.9 -> 5

exp(x) 返回e的x次幂

fabs(x) 返回绝对值

floor(x) 返回数字下舍整数 4.9 -> 4

log(x) math.log(math.e)

log10(x) -> lgx

max(x,..xn) min(x1..xn)

modf(x) 返回x的整数部分与小数部分，两部分的数值符号与x相同，整数部分以浮点型

pow(x,y) x**y运算后的值

round(x[,n]) 返回x的四舍五入值

sqrt(x) 返回x的平方根

随机数函数：

choice(seq) 从序列的元素中随机挑选一个元素

randrange([start],stop[,step]) 从指定范围内，按指定基数递增的集合获取一个随机数，基数默认值为1

random() 随机生成下一个实数，它在[0,1)范围内

seed([x]) 改变随机数生成器的seed

shuffle(lst) 将序列的所有元素随机排序

uniform(x,y) 随机生成下一个实数，在[x,y]范围内

三角函数：

asin(x) 反正弦 ...

atan2(y,x) 返回给定的x及y坐标值的反正切值

hy.pot(x,y) 返回以x,y为直角三角形斜边 sqrt(x^2+y^2)

degrees(x) 将弧度转换成角度 radians(x) 将角度转换为弧度

python中 and 为从左往右计算表达式，若1与2都为真，则默认输出第二个值

列表函数&方法：

len(list) 元素个数。

max(list) list(seq)将元组转换成列表

list.append(obj)在列表末尾添加新的对象

list.count(obj)统计某个元素在列表中出现的次数

list.extend(seq)在列表末尾一次性追加另一个序列的多个值

list.index(obj)从列表中找到某个值第一个匹配项的索引位置

list.insert(index, obj)将对象插入列表

list.pop([index])移除列表一个元素(默认最后一个)

list.remove(obj)移除列表中某个值的第一个匹配项

list.reverse()反向列表中元素

list.sort(key=None, reverse=False)

list.clear()清空列表

list.copy 复制列表

元组：

tuple(iterable) 将可迭代系列转换为元组

字典

① 字典中各同一个值被赋值两次以上，则后一次会被记住。

② 键是不可变的，所以可用数字、字符串或元组充当，不可以使用列表。

len(dict) 计算元素个数

str(dict) 输出字典，用可打印的字符串表示。

dict.clear() 删除字典内所有元素

dict.copy() 返回一个字典的浅复制？

dict.fromkeys() 创建一个新字典，以序列seq中元素做字典的键，val对字典所有键的初始值

.get(key, default=None) 返回指定键的值，若键不在字典中则返回默认值

key in dict 若键在dict中则返回true，不然返回false

dict.items() 以列表返回一个视图对象？

dict.keys() 返回一个视图对象，同 dict.values()

dict.setdefault(key, default=None) 与get()类似，若键不在字典中，则会添加键并将其设为default

dict.update(dict2) 把字典dict2的键值对更新到dict里

dict.values() 返回一个视图对象

pop(key[,default]) 删除给key所对应值，返回被删除的值

popitem() 返回并删除字典中最后一对键与值。

日期与时间.

#x import time / datetime

time.asctime([tupletime]) 接受时间元组并返回标准时间.

xx.time. 一个被转化后的时间. h, m, s

xx.timedelta. 两个时间之差

内置函数：

abs() 返回绝对值

divmod() 把除数与余数结合起来，返回一个包含商与余数的元组(a//b, a%b)

staticmethod() function. 返回函数的静态方法。 ???

all() 判断可迭代参数 iterable 中所有元素都是为 True，否则返回 False

enumerate(sequence[,start=0]) 将一个可遍历的对象组合作为一个索引序列，一般用于for循环中。enumerate(seasons)==[(0,'Spring'),(1,'Summer')...]

ord() ord()是chr()函数(对应ASCII)的配对函数或unichr函数的配对函数，返回ASCII或Unicode。 eg. ord('a')==97 ord('b')==98

any() 同all()，类似 or。 all像and

eval() 执行一个字符串表达式，并返回计算值 eval('3*7') == 21

isinstance() 判断一个对象是否是一个已知的类型，子类是一个父类类型，考虑继承关系 (类似 type())

pow() 返回 x^y 的值，计算x的y次方

sum() 对序列进行求和计算。

execfile() 执行一个文件。

issubclass() 判断是否是类创参数 classinfo 的子类。issubclass(class,classinfo) 返回T/F

super() 用于调用父类的一个方法 ??

bin() 返回一个整数int或者长整数long int的二进制表示。

file() 用于创建一个file，别名open()

iter() iter(object[,sentinel]) 用于生成迭代器。

property() 在新式类中返回属性值

bool() 将参数转换为布尔类型。

filter() 用于过滤序列，返回迭代器对象。

type() 只有一个参数则返回对象的类型。

bytearray() 返回一个新字节数组，元素的值是可变的

list() 将元素转换为列表

raw_input() 用来获取控制台的输入

unichr() 返回unicode的字符

callable() 用于检查一个对象是否可调用的。

format() 增强字符串格式化的功能

locals() 以字典类型返回当前位置的全部局部变量。

reduce() 对参数序列中元素进行累积

chr() 用一个范围在range(256)内的整数作参数.

frozenset() 返回冻结后的集合,冻结后集合不能添加或删除任何元素.

vars() 返回对象的属性与属性值的字典对象.

@classmethod...

getattr() 返回一个对象属性值

单下划线 "_" 作用：

① interpreter 中，返回最后执行的表达式的值。

② 用于忽略值：for _ in range(10):
    print("text")

③ 数字分割：10_000_000.0 作分隔符，增加数字可读性

repr() ≈ str() but it is for python.

>>> half.__repr__()	>>> half.__str__()
'Fraction(1,2)'	'1/2'

疑问：

sprinkle

fastmath ?