Python vs Javascript

李艳生

湖北师范大学

项目	Python	Javascript
命名规则	1.由字母、数字、下划线组成 2.第一个字母不能为数字 3.区分大小写 case sensitive	1.由字母、数字、下划线、美元符号$组成 2.第一个字母不能为数字 3.区分大小写 case sensitive
注释	1.单行# 2.多行”””“““或’’’ ‘‘’	1.单行// 2.多行/* */
常量	1.关键字:无 2.常量名习惯全部大写	1.关键字: const 2.常量名习惯全部大写
变量	1.关键字:无 2.习惯采用蛇型命名 sneak case	1.关键字: let 2.习惯采用驼峰命名 camel case
基本数据类型	1. 数字:int, long, float,complex 2. 字符串: ‘ ‘、 ” “, “”” “””, ‘’’ ‘’’ 3. 布尔：True, False 4. None	1. 数字：int, long, float 2. 字符串: ‘ ‘, “ “, ``` ``` 3. 布尔：true, false 4. null 5. undefined 6. Symbol
复杂数据类型	1. 元组：(v1,v2,...,vn) 2. 列表：[v1,v2,...,vn] 3. 字典：{‘k1’: v1,..., ‘kn’:vn} 4. 集合：{v1,v2, ...,vn}	1. 数组：[v1,v2,...,vn] 2. 对象：{k1:v1,...,kn:vn}
算术运算	1. 加：+ 2. 减：- 3. 乘：* 4. 除：/ 5. 整除：// 6. 求余：% 7. 幂：**	1.加：+ 2.减：- 3.乘：* 4.除：/ 5 求余：% 6.幂：**
关系运算	1. 小于：< 2. 大于：> 3. 小于等于：<= 4. 大于等于：>= 5. 等于：== 6. 不等于：!=	1. 小于：< 2. 大于：> 3. 小于等于：<= 4. 大于等于：>= 5. 等于：==， === 6. 不等于：!=， !==

逻辑运算	1. 与：and 2. 或：or 3. 非：not	1. 与：&& 2. 或：|| 3. 非：!
位运算	1. 与：& 2. 或：| 3. 非：~ 4. 异或：^ 5. 左移：<< 6. 右移：>>	1. 与：& 2. 或：| 3. 非：~ 4. 异或：^ 5. 左移：<< 6. 右移：>>>>7. 无符号右移：>>>
赋值运算符	1. 赋值：= 2. 加法赋值：+= 3. 减法赋值：-= 4. 乘法赋值：*= 5. 除法赋值：/= 6. 取模赋值：%= 7. 幂赋值： **= 8. 整除赋值：//= 9. 海象运算符：:=	1. 赋值：= 2. 加法赋值：+= 3. 减法赋值：-= 4. 乘法赋值：*= 5. 除法赋值：/= 6. 取模赋值：%= 7. 左移赋值：<<= 8. 右移赋值：>>= 9. 无符号右移赋值：>>>= 10. 位与赋值：&= 11. 位或赋值：|= 12. 异或赋值：^= 13. 逻辑与赋值：&&= 14. 逻辑或赋值：||=

分支语句	if expr1:
	    Statement
	elif expr2:
	    Statement
	elif expr3:
	    Statement
	else:
	    Statement
		if(expr1){
		    Statement;
		}
		else if(expr2){
		    Statement;
		}
		else if(expr3){
		    Statement;
		}
		else{
		    Statement;
		}
		
		switch(expr){
		case label1: statement;break;
		case label2: statement;break;
		...
		case labeln: statement;break;
		default: statement;
		}

循环语句	while expression:
	    Statement
	
	for n in range(10):
	    Statement
		while(expression){
		    Statement;
		}
		
		do{
		    Statement;
		}while(expression);
		
		for(let i = 0; i < 10; i++){
		    Statement;
		}
		
		for(let n of list){
		    Statement;
		}

中止语句	break #中止当前循环
	continue #中止本次循环
		break; //中止当前循环
		continue; //中止本次循环

复合语句	用缩进表示
	if expression:
	    Statement1
	    Statement2
	    ...
	    Statementn
		用{}表示
		if(expression){
		    Statement1;
		    Statement2;
		    ...
		    Statementn;
		}

空语句	Pass	;

with 语句	with expression as target:
	    statement
		with (expression){
		    Statement;
		}

异常处理	try:
	    Statement
	except e1:
	    Statement
	except e2:
	    Statement
	else:
	    Statement
	
	raise Exception()
		try{
		    Statement;
		}
		catch(e1){
		    Statement;
		}
		catch(e2){
		    Statement;
		}
		finally{
		    Statement;
		}
		
		throw new Exception()

函数	1. 定义
	def fname(a1, a2=default, *list, **map):
	Statement
	return expression;
	
	2. 调用
	fname()
		1. 定义
		function fname(a1,a2=default, ...a){
		Statement;
		return expression;
		}
		
		2. 调用
		fname();

Lambda	def func(a1, a2):
	Return a1 + a2
	
	func = lambda a1, a2:a1+a2
		function func(a1, a2){
		return a1 + a2;
		}
		let func = (a1, a2) => a1 + a2;

对象	1. 定义
	class CName:
	def __init__(self, name):
	self.name = name
	def sayHi(self):
	print(self.name)
	2. 创建
	c = CName(“liva”)
	c.sayHi()
		1.定义
		class CName{
		constructor(name){
		this.name = name;
		}
		sayHi(){
		console.log(this.name);
		}
		}
		2.创建
		let c = new CName(“liva”);
		c.sayHi();

继承	1.多继承
	2.定义
	class EName(CName):
	def __init__(self, name, age):
	#CName.__init__(self,name)
	super().__init__(name)
	self.age = age
	
	def sayHi(self):
	#CName.sayHi();
	super().sayHi()
	print(self.age);
	3.创建
	e = EName(“liva”, 40)
	e.sayHi()
		1.单继承
		2.定义
		class EName extends CName{
		constructor(name, age){
		super(name);
		this.age = age;
		}
		sayHi(){
		super.sayHi();
		console.log(this.age);
		}
		}
		
		3.创建
		let e = new EName(“liva”, 40);
		e.sayHi();

模块	1. 定义
	将自定义的变量，函数，类放到 mod.py 文件中即可
	#mod.py
	def sayHi():
	　　print(“Hello”)
	
	2. 引用
	import mod
	
	mod.sayHi()
		1. 定义
		将自定义的变量，函数，类放到 mod.js，用 export 关键字导出变量，函数，类
		//mod.js
		export function sayHi(){
		　　console.log(“Hello”);
		}
		2. 引用
		import {sayHi} from ‘./mod.js’;
		
		sayHi();

迭代器	str = “hello”
	It = iter(str)　#创建迭代器
	next(it)　　#迭代
	next(it)　　#迭代
		let str = "Hello";
		let iterator = str[Symbol.iterator]();//迭代器
		iterator.next();　//迭代
		iterator.next();　//迭代

生成器	def gen():　#返回迭代器的函数
	　　yield 1
	　　yield 2
	　　return 3
	
	f = gen()　#调用生成器，返加迭代器
	next(f)　#迭代
	next(f)　#迭代
		function* gen(){
		　　yield 1;
		　　yield 2;
		　　return 3;
		}
		let f = gen();
		f.next();
		f.next();

JSON	json.dumps()　#对数据编码，序列化
	json.loads()　#对数据解码，反序列化
		JSON.stringify(); //将对象转为 JSON 字符串
		JSON.parse();　//将 JSON 字符串转为对象

正则表达式	r’pattern’ re.compile(pattern[, flags])	/pattern/ let re = new RegExp("pattern", "flags");

async/await	单线程实现多任务，用于 IO 密集型任务
	async def f(): #返回 coroutine 协程对象
	　　retrun 1;
	
	f.send(None)　#调用
	
	#await 只能用在 async 函数中
	async def await_f():
	　　result = await f()
	　　print(result)
		单线程实现多任务，用于 IO 密集型任务
		async function f(){ //返回 promise 异步对象
		　　return 1;
		}
		f().then(alert); //调用
		
		//await 只能用在 async 函数中
		async function await_f(){
		　　let result = await f();
		　　alert(result);
		}

多线程	threading _thread	Worker
网络	socket httplib urllib	WebSocket Fetch XMLHttpRequest
数据库	Sqlite Mysql Mongodb	Web storage Indexdb mongodb

Web	Django flask	Express.js Kio Angular React Vue Layui
游戏	pygame	Three.js
GUI	pyQt	Electron
机器学习	Numpy Scipy Pandas Matplotlib Scikit-learn Tensorflow keras	Echarts Tesorflow.js

2020-07-24