跟我玩Blockly

主编：周庆国

编者：邓文博、李鹏、胡轶凛、尹航、
漆昱涛、郅朋、魏小辉

前 言

以计算机技术为先驱的科技革命深刻地影响着我们的生活生产方式、管理方式以及思维方式，推动着人类社会的蓬勃发展。有人说计算机技术就像人大脑的延伸，帮助人们计算、设计、创造、解决各种各样的问题。随着智能时代的到来，我们应尽早的学习和掌握计算机知识，同时也希望大家能拥有像写作一样普遍的编程技能。

随着计算机技术的不断发展和进步，编程过程和编程方法已经变得简单而有效。Blockly 不需要你考虑冗杂的代码和一大堆语法规则，类似堆叠积木块一样，用户只需将所需模块拖拽到编程区域进行逻辑拼接，即可在 WEB 页面直接运行，并查看运行结果。既然 Blockly 编程可以如此方便，那么让我们先来了解一下关于 Blockly 这本书的内容以及学习建议吧：

1、本书共分为六个章节，每章节都提供了详尽的程序案例，学习者可以仿照程序案例来运行，从而加深理解，并结合每章的课后习题进行练习，这样才能更好的理解编程过程。

2、建议学习者按照教材章节内容依次学习编程，认真阅读教材、合理应用，理解每个单元的模块功能和编程思路，积极结合生活实践拓展，深入编程学习。

3、每章最后提供本单元的知识汇总图，对本章节知识技能和编程要点进行梳理。在每章学习结束后，可以参考知识汇总图来总结自己的知识掌握情况，反思学习效果。

本书可以作为算法与程序设计初学者的教材参考书，也可以是编程爱好者快速了解 Google Blockly 的使用指南，同时可以作为高中信息技术课程算法与程序设计课程的教学活动参考书。

非常感谢帮我们完成这本书的朋友，我们在此向他们致谢：周庆国，主编；邓文博,Google Blockly 教学应用手册;李鹏,Blockly 编程基础;胡轶凛,Blockly 顺序程序设计；尹航，Blockly 之选择结构；漆昱涛，循环结构；郅鹏，Blockly 的进阶；魏小辉，进行整本书的修订和审核。

在本书编辑过程中不免有纰漏，欢迎读者批评指正。

编者按

兰州

2017 年 3 月 20 日

目 录

第一章 Google Blockly 教学应用手册

1.1 Google Blockly 概述

美国计算科学教育一直认为计算机作为当今各个领域的基础技术工具，有必要让学生尽早的了解和学习计算机工作原理，理解基础的编程思维。在 2006 年周以真教授提出计算思维概念，这很大程度上促进了人们对计算机编程教学的认识。图形化编程摒弃了繁琐的代码语法，以直观的图形化模块进行编程，其思想最早来源于西摩尔·帕伯特的心理学家在从事儿童学习的研究中发明的 LOGO 语言。近年来随着计算机技术的快速发展，出现了很多优秀的图像化编程工具提供给学生学习计算机程序设计，比如：Alice、Scratch、AppInventor 等著名软件。在 2012 年 6 月，Google 发布了完全可视化的编程语言 Google Blockly，这是一款完全开源的，集合多种编程语言的编程工具。很多图形化编程平台都是基于 Google Blockly 二次开发的，譬如：APPInventor、Wyliodrin、Earsketch 等优秀编程平台。Google Blockly 作为一种易于掌握的图像化开源编程环境，是编程初学者学习和掌握程序设计方法的有力工具，非常有必要予以了解和掌握。

1.2 什么是 Google Blockly

1、一种基于网页的可视化程序

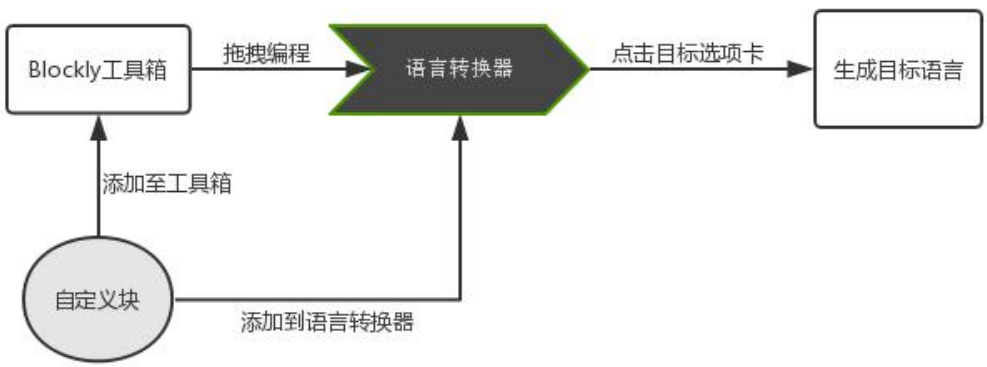
Google Blockly 是基于网页的可视化编程工具库。用户可以以离线或者在线的方式在 Windows、Linux、MC 和 Android 平台上的浏览器端进行编程操作。可以使用计算机端、手机或平板移动端进行随时随地的完成编程设计，教学编程方式多种多样。

2、多种开发语言环境库

Blockly 基于图形化编程设计可以导出 Javascript、Python、PHP、Lua、Dart 等多种语言。通过图形化编程完成程序设计，在 Blockly 中有一个类似语言转换器的工具箱，可以将图形化编程语言转化成多种编程语言代码。用图形化编程方式去理解多种程序语言。

3、开源的自定义编程环境

Blockly 是开源的编程工具，用户可以根据自己编程的特点要求，对 Blockly 工具箱进行自定义设计。同时，Blockly 开发工具能让用户自定义块导出至工具箱，并在工作区工厂完成对代码的封装。如图所示。

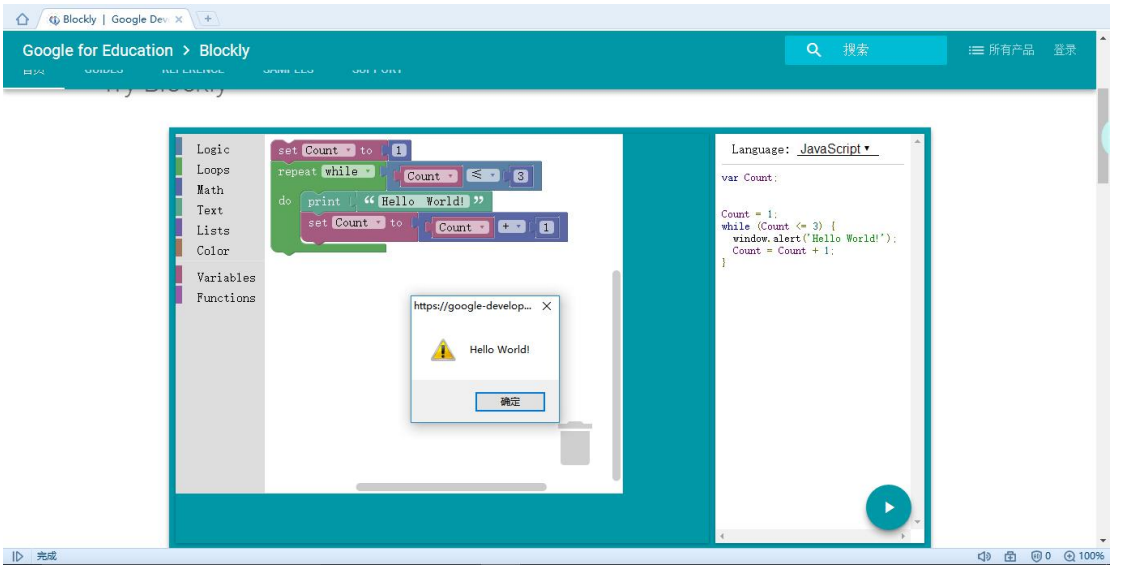


1.3 Google Blockly 的编程环境

Blockly 是一个可用于 Web、Android、iOS 的可视化代码编辑器库

1. Blockly 在线使用

打开浏览器，在地址栏输入 https://developers.google.com/blockly/，前往 Blockly 官网，即可体验编程。如图所示



2、Blockly 的离线环境搭建

在 Github 网站或者 Blockly 主页上找到对应系统的文件包，下载后，无需安装，解压，进入 demos 目录，打开 index.html，选择相应的选项，即可体验。

Linux 系统，可下载 TAR Ball，在终端进行文件解压即可；

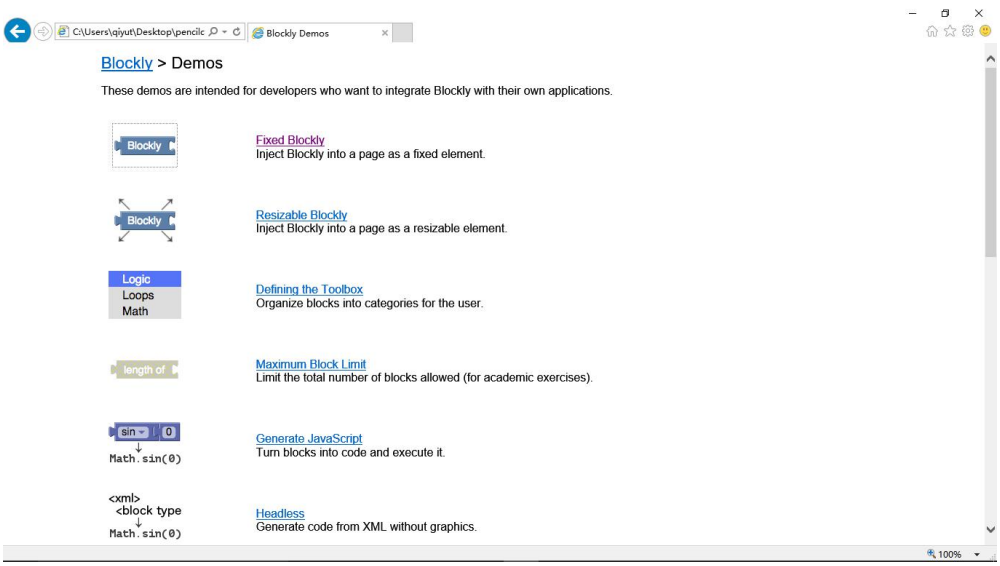
Windows 系统，可下载 ZIP File，并继续解压即可;

Github Blockly 地址: https://github.com/google/blockly

TAR Ball 地址: https://github.com/google/blockly/tarball/master

ZIP File 地址: https://github.com/google/blockly/zipball/master

Blockly 离线使用都是免安装的，只需 Clone 或解压后，进入 demos 目录，打开 index.html，选择相应的选项，即可体验。



1.4 Google Blockly 在中学信息技术课程教育应用优势

Blockly 是一个基于 Web 的可视化编程工具，以图形化语言编程，这个新语言的主要目的是为 web 应用提供宏(或脚本编程)的支持。可以把生成的脚本输出成 Javascript, python 等，还可以自定义图形化编程工具。目前可以在中小学利用 blockly 开始信息技术课程的编程学习，同时结合 Arduino 开发板，进行创客教育的探索与实践。

1、基于 WEB 跨平台的图形化编程软件

在中小学信息技术课程中，编程算法的教学工具的选择一直是教师专家探究的主要问题。代码程序的封装使得图形化编程工具的优势越来越明显，在计算机编程方面出现许多优秀的图形化编程工具。Blockly 是基于 WEB 平台的图形化编程工具，可以在任何系统平台进行在线与离线版操作使用编程，也可以在教室主机现场搭建服务器平台，WEB 界面实时编程，测试编程结果。

2、定制自己的编程工具—项目式教学探索

目前的 Scratch、PencilCode 以及 APPInventor 等流行的图形化编程软件中，

由于图形程序模块的局限性，一些特殊功能很难在平台实现。Blockly 提供的代码自定义工厂功能，使得用户可以利用集成块编辑器，创建符合自己创作要求的应用程序块。Blockly 只是一种生成代码的方法，你的应用程序的核心在于如何处理该代码。这样一种自定义图形化块的方式可以方便师生在教学任务过程中自定义符合教学功能的模块，实现教学工具的再次开发。

3、结合开源 Arduino 进行创新体验编程教学

中小学阶段使用 Arduino 开源硬件版进行创客课程教学非常普及，使用 Blockly 平台结合 Arduino 开源硬件，是构建高中信息技术课堂的 STEAM 教学项目新探索。学生可以基于图形化编程跨平台创造作品，通过程序功能设计的硬件外化过程，建构跨学科知识课堂。

课后练习

1. 进入 Blockly 官网，熟悉 Blockly，并使用在线 Blockly 输入“Hello Blockly”。

2. 在本地配置离线版 Blockly，并完成“Plane”游戏的练习。

第二章 Blockly 编程基础

2.1 Blockly 的数据类型

2.1.1 数据的含义

在计算机程序的世界里，程序的基本任务就是处理数据，无论是数值还是文字、图像、图形、声音、视频等信息，如果要在计算机中处理的话，就必须将它们转换成所谓的数字信息，因为计算机中只能存储数字，甚至连计算机程序都是由数字组成的，所以在使用计算机程序解决问题的时候，首先需要把需要处理的信息数字化，即使用数字表示需要处理的信息。如果我们要处理图像信息，可以把一副图像可以看做是由 m 行 n 列的点组成的，每一个点是一种颜色，每一种颜色可以使用三个数据（R、G、B）来表示，R 表示红色的比例、G 表示绿色的比例、B 表示蓝色的比例，这样就可以用 $m \times n \times 3$ 个数据表示一副图像了。如果我们需要处理文字信息，例如英文，文字是由字母及一些标点符号组成的，我们可以将每一个符号用一个数字来表示，例如在 ASCII 编码标准中，用 65 表示字母“A”，用 66 表示字母“B”等等，只要把我们所使用的每个符号都进行编码（数字化），我们就可以在计算机中处理文字信息了，总之，不论什么信息，如果要使用计算机程序来处理的话，就必须进行数字化，即将它们转换成数字表示的形式，才能够在计算机中处理。如果我们使用 BLOCKLY 书写的程序处理数据，由于任何信息在计算机中都是以数据的形式存在的，表示数据的形式可能就多种多样，于是就出现了这样一些基本问题，在 BLOCKLY 可以使用那些种类的数据？每一种类型的数据的书写形式是怎样的？本章的其余部分将详细讨论这些问题。

2.1.2 Blockly 中的数据

1.计算机中数据的表示形式

(1)二进制表示

众所周知，在计算机中，采用二进制代码表示字母、数字字符以及各种各样的符号、汉字等。在处理信息的过程中，可将若干位的二进制代码组合起来表示各种各样的信息。但由于二进制数不直观，人们在计算机上实际操作时，输入、输出的数值数据都是十进制，而具体转换成二进制编码的工作则由计算机软件系统自动完成。字母和各种字符在计算机中的传输普遍采用 ASCII 码，即美国标准信息交换码，它用了 7 位二进制数来表达字母和各种常用字符。对于汉字信息的表示比较复杂，我国有汉字几万个，常用的汉字也有 7000 多个，为了统一，我国制定了汉字编码标准，规定了一、二级汉字共 6763 个，用两个字节来表示一个汉字。

(2)十进制表示

十进制形式是我们最熟悉的表达形式，十进制数的书写规则是由正负号开头，后跟一个自然数的形式，如果是正数，正号可以省略。例如：-213、0、415、76、+83 都是合法的整数。在 BLOCKLY 中，如不特殊定义，所有数一般默认为十进制。

(3)八进制形式

使用八进制形式表示一个整数，八进制数的书写规则是以数字 0 开头，后跟一个八进制形式的数，如果是负数，则以负号开头。例如：0123、-087、00、+0327

等都是合法的八进制形式。

(4)十六进制形式

以“0x”开头，后跟一个十六进制数，例如：0xFF03、0x123、0xAC7 等都是合法的十六进制形式，而 x37、287 都是非法的十六进制形式。

2.Blockly 中的数据类型

(1)实数

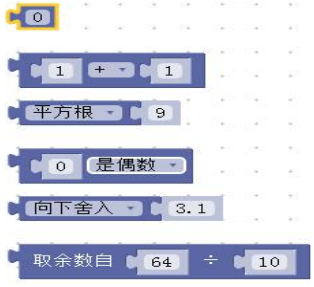
程序中所有的数据都有特定类型，数据的表示方式、取值范围以及对数据可以使用的操作都由数据所属的类型决定。类型可以帮助编译程序生成高效的目标代码，编译程序在生成目标代码时，可按需分配存储空间可如何引用这个数据。一个数据属于某个特定的类型后，在数据上允许操作的运算也确定了。例如，整数可以做四则运算等；字符串则可以进行比较、连接、判断子串等，但不能做四则运算。

例如：

在 BLOCKLY 中，提供了数据输入模块 

，其默认一定的存储长度，默认数值为 0。

在一些计算公式中提供了的数据输入模块，如下。



各模块数据输入只区分数字、字符类型，也就是说在允许输入数字的模块中不允许输入字符，可以输入任何数字。但是在程序具体执行过程中，程序会对输入数字类型的合法性进行检查。

(2)字符

字符类型的数据,字符在内存中存储的是该字符的 ASCII 编码值，由于字符数据存储的就是一个字符的编码数值，所以字符数据也可以当做一个整数。在 BLOCKLY 中的基本表示形式是用“”引用起来，比如“A”、“”Q、“a”、“b”、“#”、“-”、“。”等。使用双引号将一个数字放在引号里面，其意义也表示该数字，和不用引号表示的意义相同,如“65”和 65 的意义相同。但是大写字母和对应的小写字母对应的 ASCII 编码值不同，因此为不同的字符。如“A”、“a”为不同字母。BLOCKLY 中的字符输入模块为  。

同样的，在 BLOCKLY 的字符输入模块中，允许输入任何形式的字符和数字，只要不超出特定的长度都是合法的。只有在程序执行的时候才检查输入是否正确。

(3)字符串

在 BLOCKLY 中字符串的表示和单个字符的表示形式是一样的，输入模也是  。

下面几个模块是 BLOCKLY 提供的字符串输入模块。

如字符串输入模块：

如求字符串的长度：

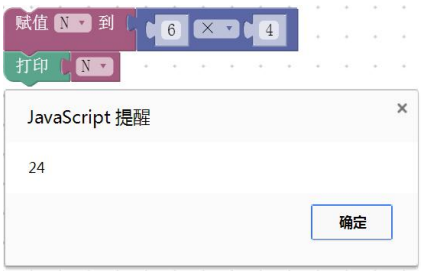
如小写字符变大写字母：

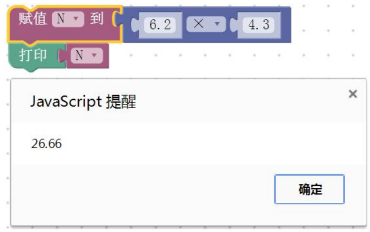
如添加字符串：

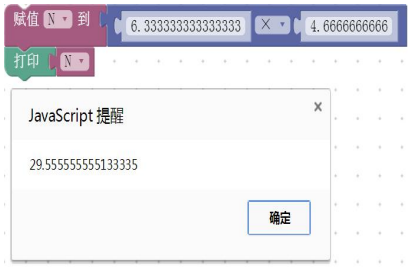
(4)数据使用

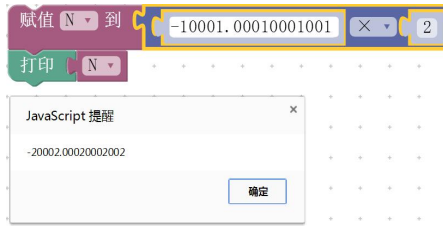
前面讲过，在 BLOCKLY 中，数据和字符定义过程中程序给了一定的存储空间，

BLOCKLY 不计较输入数据或字符的类型和长度，程序员不需要考虑非法输入带来的麻烦，这给了程序员极大的方便。如下：









但是，在程序运行过程中，如果数据输入错误或当数据超过程序可表示的范围，数据将会产生错误。例如

如图可见，当超过 16 位后的数据不再进行计算，而是原数据输出。

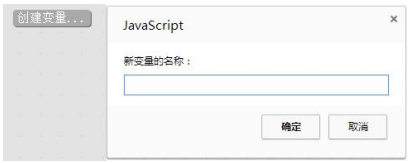
例如：

在该循环语句中的重复次数输入模块默认输入数据为正整数。如果输入负数，程序默认为 0；如果输入小数，默认在其整数部分上加 1。

2.2 变量的定义和赋值

上一节讲的数据和字符，当给定一个值后，在程序中是确定的不能改变的量，我们称为常量，而与之对应的就是变量，顾名思义，就是在程序中经常需要改变的量。

2.2.1 变量定义



上图为 BLOCKLY 提供的变量表定义方式，变量的命名方式比较随便，不受限于数字或字符，但是我们为了使用方便，尽量选用简单明了的字符，避免与程序中的其他名称重复。并且 BLOCKLY 提供的变量定义不区分类型，只是在内存中分配一定的存储空间。

2.2.2 变量的初始化

部分变量需要进行初始化，BLOCKLY 提供的变量初始化模块为 

。变量的初始化不是必须的，根据需要可初始化或者不初始化。

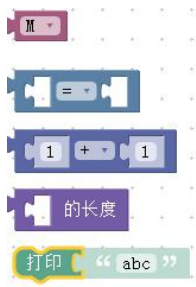
2.3 表达式和数据的运算

2.3.1 表达式的概念

运算符是指用来表示在数据上执行某些特定操作的符号。参与运算的数据称为操作数。

根据参与运算的操作数的个数是一个、两个或三个，运算符分为一元运算符、二元运算符和三元运算符。表达式是指用运算符和圆括号把常量、变量和函数等运算成分连接起来的有意义的式子。单个常量、变量和函数也都可以看成是一个表达式。表达式经过计算后都会得到一个确定的值，这个值就是表达式的值。每个表达式都具有唯一确定的值和唯一确定的类型。

BLOCKLY 中每一个单独的模块即为一个表达式。如:



以上模块在输入正确的数据后均为合法的表达式。

2.3.2 运算符运算表达式

BLOCKLY 中含盖了日常使用的所有运算符，我们主要认识一下常用的几类运算符：

（1）算术运算符 (+ - * / % ^)

（2）关系运算符 (> < >= <= = !=)

（3）逻辑运算符 (&& || !)

（4）赋值运算符 (=)

（1）基本算术运算符

基本算术运算有 6 种运算符：

（1）+ 加法运算符

（2）- 减法运算符

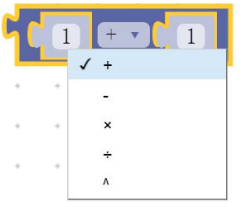
（3）* 乘法运算符

（4）/ 除法运算符

（5）% 求余数运算符

（6）^ 次方幂运算

基本算术运算符的表达式格式为：**<操作数>运算符<操作数>**。BLOCKLY 中给出的模块为：



（2）关系运算

关系运算符有 6 种：

（1）= 等于运算符

（2）≠ 不等于法运算符

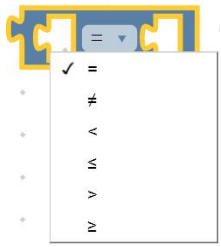
（3）≤ 小于等于运算符

（4）< 小于运算运算符

（5）≥ 大于运算符

（6）> 大于运算符

关系运算符的表达式格式为：**<操作数>运算符<操作数>。**BLOCKLY 中给出的模块为：



（3）逻辑运算

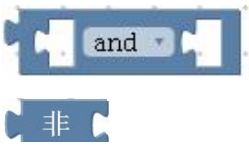
逻辑运算符有 3 种：

（1）&&　　逻辑或运算符

（2）||　　逻辑与运算符

（3）！　　逻辑非运算符

逻辑运算符的表达式格式为：**<操作数>运算符<操作数>和运算符<操作数>两种形式。**BLOCKLY 中给出的模块为：



（4）赋值运算

BLOCKLY 中赋值运算与变量初始化的表达式相同。

2.3.3 运算表达式的值

（1）单个常量或变量的表达式，其值为常量或变量的值。如 98、“8”、X，其值分别为 98、8、X 的值。

（2）算术运算表达式的值为其运算结果。如 3+2、5-6、4*8，值分别为 5、-1、32。

（3）关系运算表达式的值只有两个：1 和 0（真或假）。如 4<2、2>1、1=2,值分别为 0、1、0。

（4）逻辑运算表达式的值只有两个：1 和 0（真或假）。如(x<10) ||(x>20)、!(3>2)，其值为 0、0。

（5）赋值运算的值即为所赋的值。如 a=3,b=6,其值为 3、6。

2.3.4 运算符的优先级

BLOCKLY 中与其他编程语言不同，不需要太多的考虑运算符的优先级问题，因为 BLOCKLY 将不同的运算符集成在不同的模块中，在使用中以模块嵌套的形式形式出现，因此其运算顺序只能是由里到外。如：

课后练习

1. 对于计算机而言，无论是数字、字母、符号，在计算机中都是以 0、1 的形式存储和计算，但是他们在 BLOCKLY 中有不同的运算规则，为什么？

2. 分别求出 a=3，b=a+3，b>a 三个表达式的值和变量 a 或 b 的值，认真思考表达式的值和变量的值有什么区别？


第二章 Blockly编程基础
核心概念
数据类型
二进制
八进制
十进制
十六进制
字 符
运算符
表达式
变量和常量
本单元知识点
数据类型
运算符
表达式
技巧
不同运算符间的嵌套
表达式的值
涉及工具块
变量
数学
文本

第三章 Blockly 顺序程序设计

3.1 什么是 Blockly 语言

2012 年 6 月，Google 发布了完全可视化的编程语言 Google Blockly，整个界面清晰明了， 你可以如同在玩拼图一样用一块块图形对象构建出应用程序。每个图形对象都是代码块，你可以将它们拼接起来，创造出简单功能，然后将一个个简单功能组合起来，构建出一个程序。这个程序与我们平时使用的编程语言最大的差别是无需自己编写代码，在我们所使用的代码块的背后就是已经编写好了的代码，而用户并不需要关心这些，整个过程只需要鼠标的拖曳，不需要键盘敲击。

我们首先通过一个小游戏来了解 Google Blockly,在 Blockly Games:Maze 这个小游戏中，每个关卡都会提供一个地图，地图上有起点和终点，而我们要做的事就是利用每个关卡提供的语句模块构成一个小程序，使得程序运行起来时起点处的小人能够顺利从起点到达终点。

例如第二关，小人需要经过两次转弯才能到达终点，需要注意的是，第一次转弯后小人必须前进一部分路程才能进行第二次转弯。

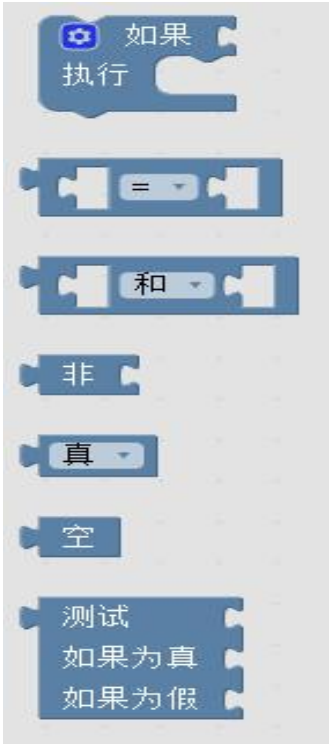
所有组成的模块如下：

在这个游戏中，我们最需要弄清楚的是每个数据块执行的顺序，以及我们应该如何摆放它才能让小人顺利的到达终点，这就引出了本章的重点，Blockly 语言的顺序程序设计。

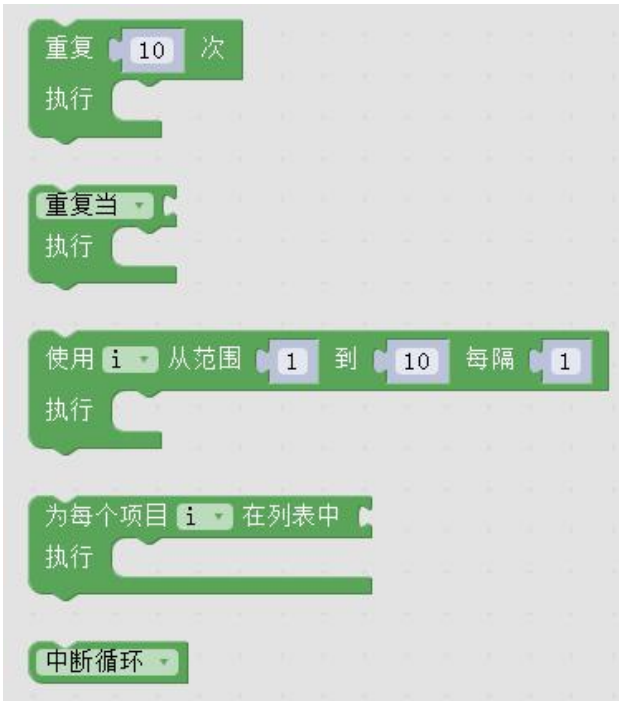
在本章的学习中，我们将学习到几种顺序执行的语句，在这些语句的执行过程中不会发生流程控制的转移，比如赋值语句，输入输出语句。

在讲解 Blockly 语言的顺序程序设计之前，让我们先来对 Blockly 语言做一个总的概述，Blockly 语言总共分为 8 个板块。

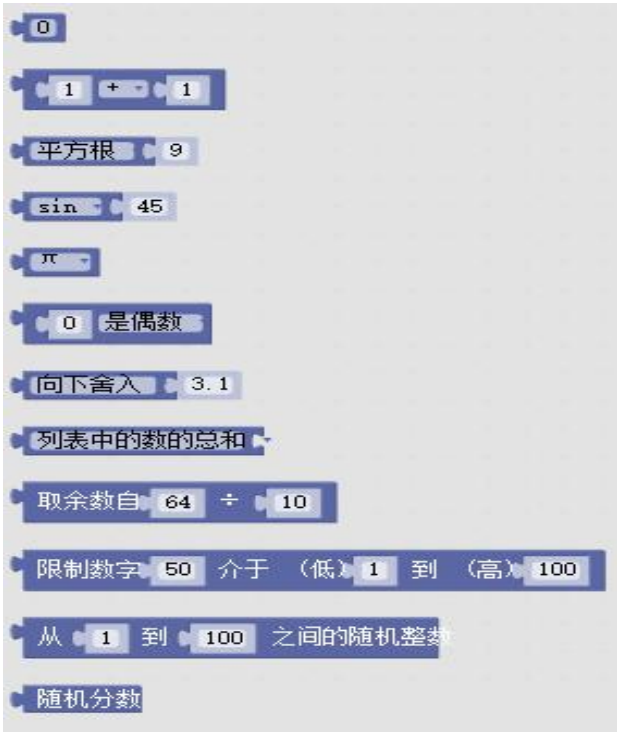
（1） Logic ，表明数据间的逻辑关系。



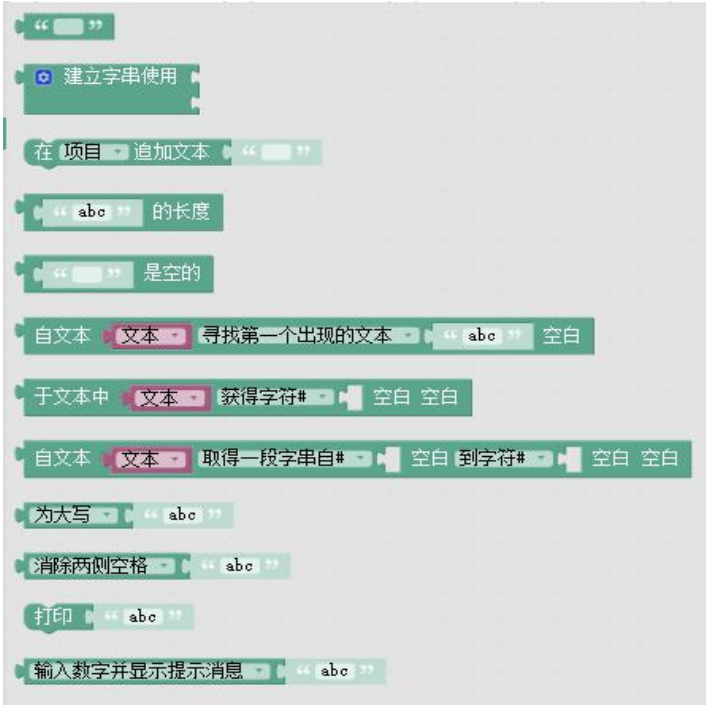
（2） Loops 循环控制



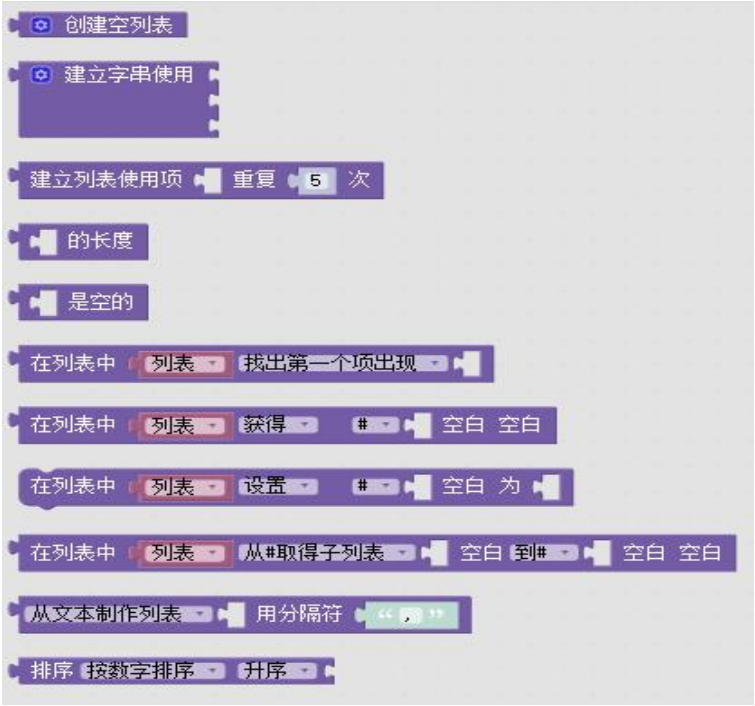
（3） Math 数学运算模块


0
1 + 1
平方根 9
sin 45
π
0 是偶数
向下舍入 3.1
列表中的数的总和
取余数自 64 ÷ 10
限制数字 50 介于 (低) 1 到 (高) 100
从 1 到 100 之间的随机整数
随机分数

（4） text 文本块


""
建立字串使用
在 项目 追加文本 ""
"abc" 的长度
"" 是空的
自文本 文本 寻找第一个出现的文本 "abc" 空白
于文本中 文本 获得字符# 空白 空白
自文本 文本 取得一段字串自# 空白 到字符# 空白 空白
为大写 "abc"
消除两侧空格 "abc"
打印 "abc"
输入数字并显示提示消息 "abc"

(5) list 列表块



(6) Colour 颜色块

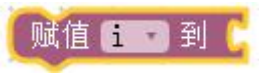
(7) Variables 变量块

(8)Functions 函数块

当学生学习了新的函数或者命令，就可以使用这些 Blockly 语句块进行联系，所有的块被组织排放在左侧的列表中，使用时根据正确的语法和适当的缺口对接就能实现预定的功能。因此，我们可以通过对块进行适当的组织就能轻松的实现每一个新的想法和创意。

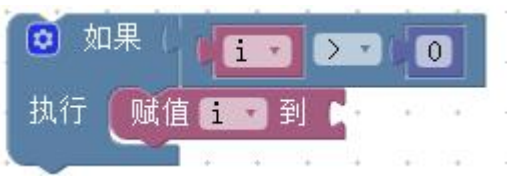
3.2 赋值语句

在 c 语言的学习中，我们可以知道赋值语句是由赋值表达式加上一个分号构成。而在 Blockly 语言中，赋值语句是由如下一个语句块构成：



其中 i 指的是一个变量，也可以用其它字母代替，在 to 后面紧跟着的是要赋给 i 的值。同样的，这个赋值表达式也可以包括在其它表达式中，例如：

if 后面跟着的是一个条件，例如可以是



其作用是当 i 大于零时，将一个值赋给 i。

3.3 Blockly 语言的输入与输出

当计算机被用于和外界交互时才是最有趣的，所谓的输入与输出是以计算机主机为主体而言的。

输入就是将数据从输入设备带入计算机（如键盘，磁盘，光盘，扫描仪等）

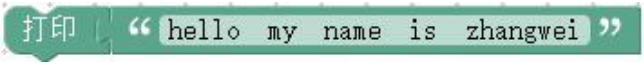
输出就是将数据从计算机发送到外部输出设备（如显示屏，打印机，磁盘等），输入输出有时候并称为 I/O。有许多种类的 I/O，包括人机界面，网络接口，存储设备接口和自动机器接口。计算机在处理各种输入输出上有许多共同点，无论

是与一个人，一个文件或其他一些设备进行交互。用户可以通过只学习如何创建用户界面来学习重要的 I/O 技术。

我们先来介绍 Blockly 语言的输出语句块：

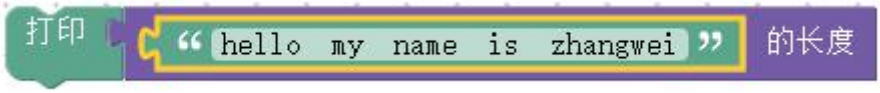
它可以根据后面所接的不同的语句块而输出不同的数据，例如：

它输出的是两个数字的和。

而 

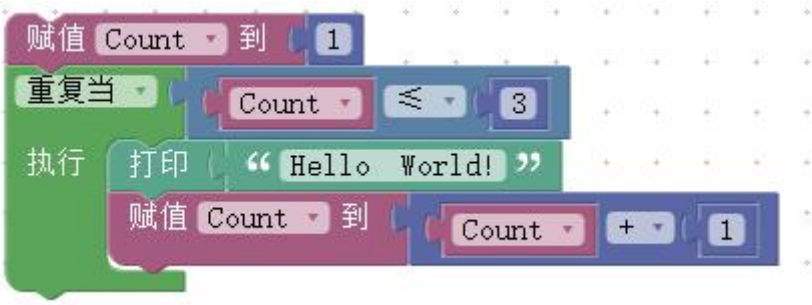
则是在屏幕上输出这一段文本。

而在这一段文本的前面加上另一个数据块则又是另外一种效果 如：



上面这一行数据模块则是输出的这段文本的长度。

例 输出字符串“Hello World!”



这段程序将会输出三次 Hello World!，在程序中，我们先将步长的值设置为 1，当步长的值小于等于 3 的时候进入 repeat 循环，在屏幕上输出 Hello World! 的字样然后将步长的值加 1 加赋值给步长,直到步长的值大于 3 时将不再进入循环，所以我们不难知道，这段程序将会在屏幕上输出三次 Hello World!的字样。

介绍完输出的语句块之后，让我们再来介绍输入的语句块，Blockly 语言的输入语句块为

这个输入语句块输入的既能是文本，也可以是数字，通过语句中间的选项可以进行不同的选择，当我们运行这句语句块的时候，屏幕同样会弹出一个文本框，这个时候我们就能在文本框里面输入我们想带入计算机的数据，点击确认后，我们所输入的数据就将会进入我们所设置的变量里面，下图便是我们输入数据的界面:

通过上面简单的介绍，同学们可能对输入的理解还不够深刻，下面让我们来举一个具体的例子。

首先，让我们设置一个变量 a， 

然后再将上面的输入语句块连接在设置变量语句块的后面，点击运行，在出现的为文本框里面输入我们想要输入的数据，点击确认以后，数据就会被赋值给 a 了，如果同学们想确认 a 的值是不是真的是我们所输入的数据，可以在这段数据块的下面加上输出数据块，将 a 的数据输出到屏幕上，这样我们就能确认 a 的值了。

3.4 顺序结构程序设计举例

例 从键盘输入一个大写字母，要求改用小写字母输出。

看到这个题目，同学们首先想到的是什么呢?在前面介绍的几个 Blockly 模板中大家会首先想到哪个语句块?或许记忆力好的同学已经想到，在我们介绍到的 Text 板块中就有一个语句块是用来转换大小写的， 

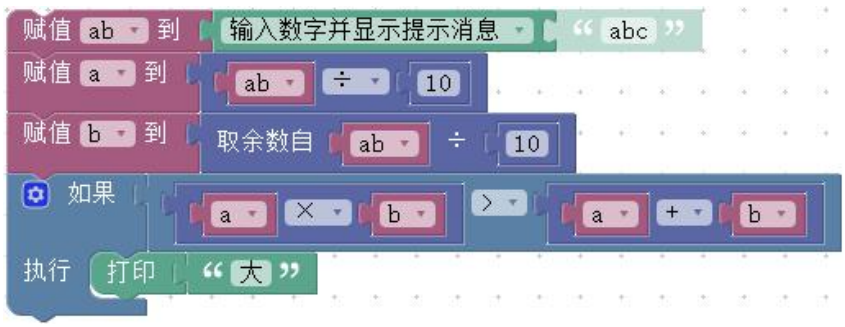
这个语句块使用起来相当简单，只需要将你需要转换的文本连接在此语句块的后面就行了，此语句块同样能根据需求不同产生三种不同的效果，我们所做的只要改变语句块中的选项罢了。

既然已经找到了这个问题所需要的核心语句，那么后面的问题就简单了，不难看出这个题目同时用到了输入和输出，所以我们只需要设置一个变量用来存放我们所输入的数据，然后将输入的数据转化成小写并输出，这个问题就解决了。所组成的数据块如下：

运行结果如下：

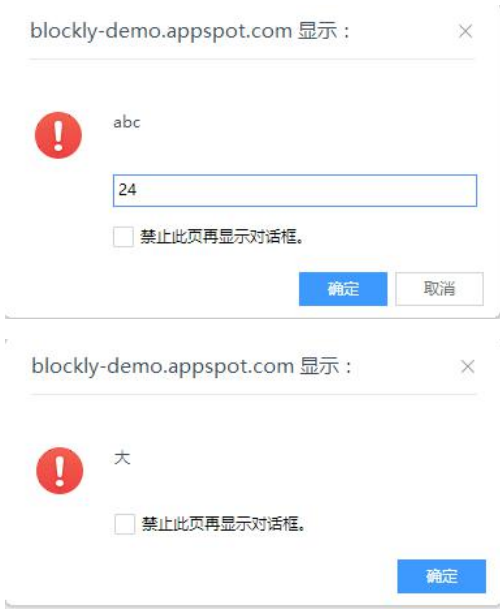
例 输入一个两位数，如果这两位相乘大于两位相加，则输出“大”这个字。

在同学们第一次见到这个题目时，可能会感到有点手足无措，但其实我们一步步分析就会发现这个题目不过如此，在解决这个问题前，我们首先要搞清楚怎么根据输入的俩位数分别得到个位数和十位数，如果大家曾经接触过其它语言，就会知道俩位数 除以 10 得到的商就是十位上的数字，而得到的余数就是个位上的数字，在弄清楚这个问题以后，这个题目同样变得不堪一击了。具体数据块如下：



在这个组好的数据块里，我们首先将输入的俩位数存到 ab 这个变量里面，然后将计算得出的个位数和十位数分别赋值给 b 和 a，再利用我们前面提到的 logic 模板里面的 if 语句块判断大小，最后输出。

运行过程与结果如下：



通过本章的讲解，相信大家对 Blockly 语言的顺序程序设计有了大概的了解，也对输入输出有了清晰的认识，语言的顺序程序设计在同学们今后的语言学习中起着相当重要的作用，希望能引起大家的重视。

课后练习

1、对两个整数变量的值进行互换。

2、如果是做单项选择题，请根据给定的选项，输出对应的结果。

举例：

总共有 4 个字符。A，B，C，D。

你给出字符 A，输出：你选择了 A

你给出字符 B，输出：你选择了 B

你给出字符 C，输出：你选择了 C

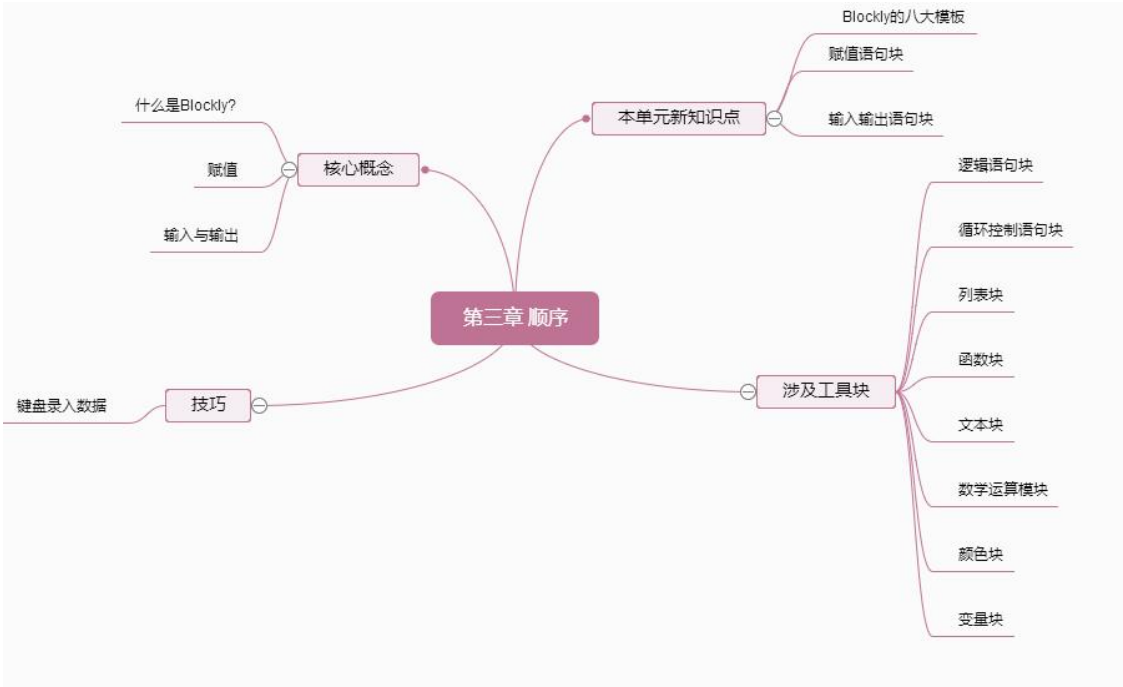
你给出字符 D，输出：你选择了 D

3、根据输入的值，判断是星期几。

举例：

输入：1

输出：星期 1


第三章 顺序
本单元新知识点
Blockly的八大模板
赋值语句块
输入输出语句块
核心概念
什么是Blockly?
赋值
输入与输出
技巧
键盘录入数据
涉及工具块
逻辑语句块
循环控制语句块
列表块
函数块
文本块
数学运算模块
颜色块
变量块

第四章 Blockly 之选择结构

今天，我们通过一个游戏来学习选择结构，游戏的地址如下：

https://blockly-games.appspot.com/bird?lang=en

本游戏分为 10 关：主要游戏规则如下：

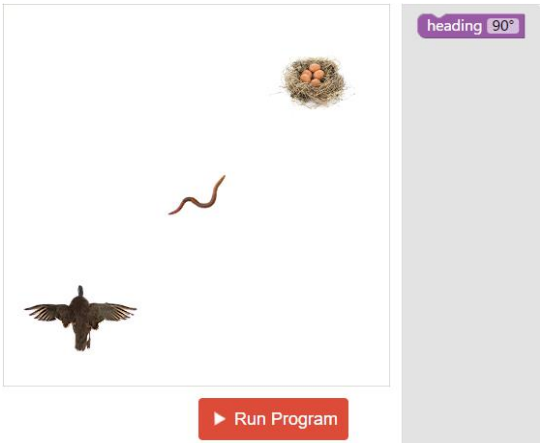
①主界面是游戏的运行界面，我们需要通过控制代码来让鸟叼完虫子后回到鸟窝，并保证不撞到墙。

②代码主要由选择结构和逻辑判断组成。

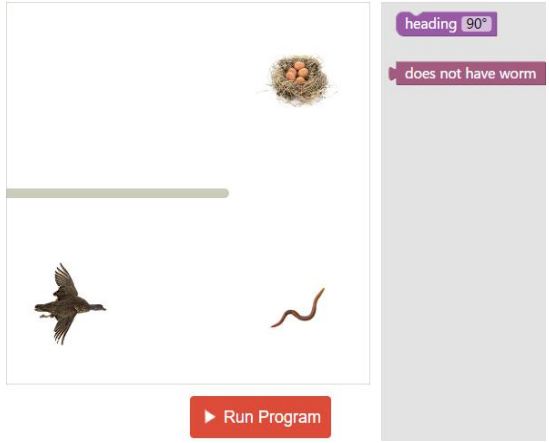
③点击下方的 Run Program 按钮后程序就会执行右侧的代码。回到鸟窝后，游戏结束，顺利通关。

你可以先尝试着看看鸟应该怎样去运动，并如何调用代码块。稍后我们将会详细介绍关于选择结构的各种语法。

1.

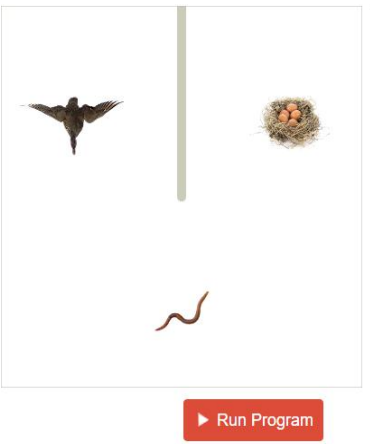


2.

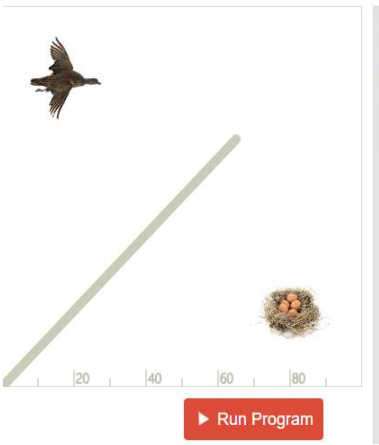


3.

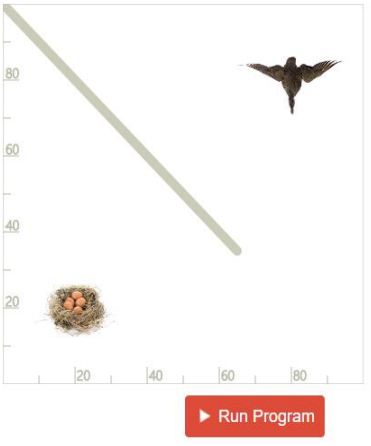
4.


heading 90°
does not have worm
Run Program

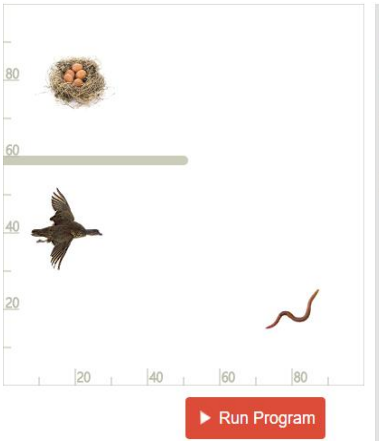
5.


heading 90°
does not have worm
x < 50
20
40
60
80
Run Program

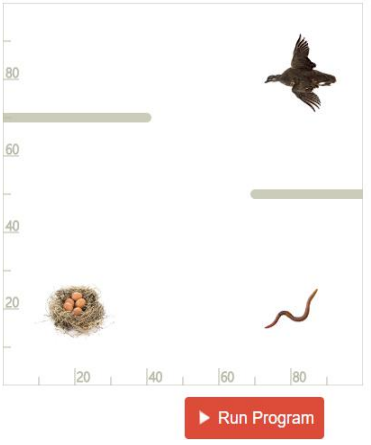
6.


heading 90°
does not have worm
x < 50
y < 50
80
60
40
20
20
40
60
80
Run Program

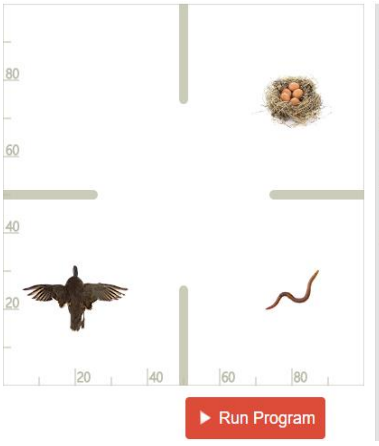
7.


heading 90°
does not have worm
x < 50
y < 50
80
60
40
20
20
40
60
80
Run Program

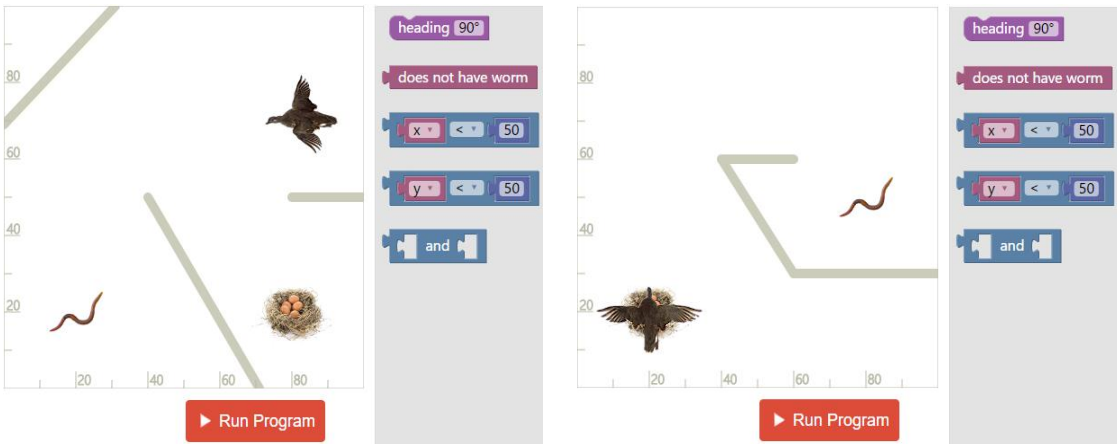
8.


heading 90°
does not have worm
x < 50
y < 50
80
60
40
20
20
40
60
80
Run Program

9.


heading 90°
does not have worm
x < 50
y < 50
and
80
60
40
20
20
40
60
80
Run Program

10.



4.1 关系运算符和关系表达式

关系运算是逻辑运算中比较简单的一种，所谓的关系运算实际上就是比较运算，将两个值进行比较，从而判断比较的结果是否满足符合的条件，比如关系表达式 a>5，如果 a 为 6，那么表达式成立，结果就是真，反之，如果 a 的值为-1，那么表达式不成立，结果就是假。

4.1.1 关系运算符及其优先顺序

我们有如下 6 种关系运算符：>，<，≥，≤，==，!=

优先级次序如下：

（1）前四种的优先级顺序高于后两种，前四种优先级相同，后两种优先级相同。

（2）关系运算符的优先级低于算术运算符。

（3）关系运算符的优先级高于赋值运算符.

举几个例子：

$$c > a+b \quad \Longleftrightarrow \quad c > (a+b)$$

a > b == c ⟷ (a > b) == c

a == b<c ⟷ a == (b<c)

a=b>c ⟷ a=(b>c)

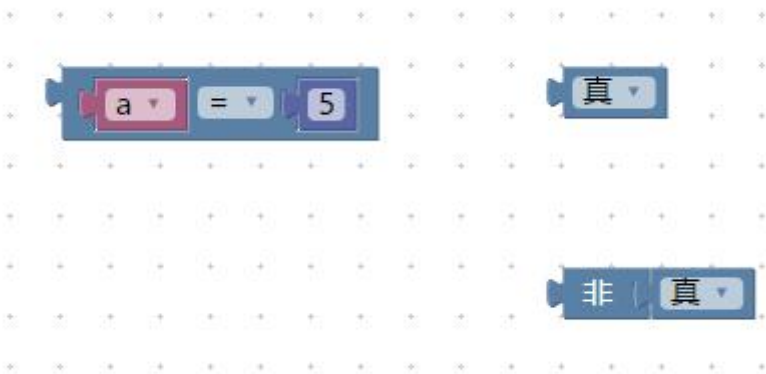
4.1.2 关系表达式

关系表达式的结果有两个，如下图所示，真和假（非真）。

我们需要注意“=”和“==”的区别：

“=”表示赋值语句，比如 a = 5，是把 5 赋值给变量 a

“==”是逻辑判断，比如 a == 5，是表示变量 a 的值是否和 5 相等，如果相等就返回真，反之返回假，我们的 blockly 如下图所示，逻辑判断“==”写成了“=”，是为了方便大家去理解，但实际上你要知道这两者是有区别的，细心的读者打开 blockly 转换到代码就会发现代码中显示的都是“==”。

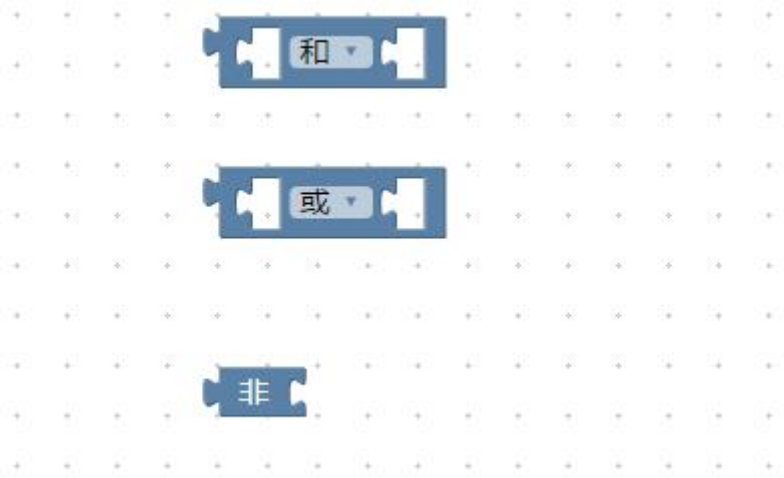


4.2 逻辑运算符和逻辑表达式

4.2.1 逻辑运算符及其优先级

逻辑运算符包括与、或、非，在我们的 blockly 中，如下图所示去表示分别

为和、或、非。



逻辑和，a 和 b，当且仅当 a，都为真时，结果为真。

逻辑或，a 或 b，只要 a，b 中有一个为真，结果就为真。

逻辑非，非 a，如果 a 为真，则结果为假，如果 a 为假，那么结果为真。

我们用图表的形式来表示如下：

a	b	a and b	a or b	not a
真	真	真	真	假
真	假	假	真	假
假	真	假	真	真
假	假	假	假	真

（1）非逻辑，即非在三种运算符中优先级最高。

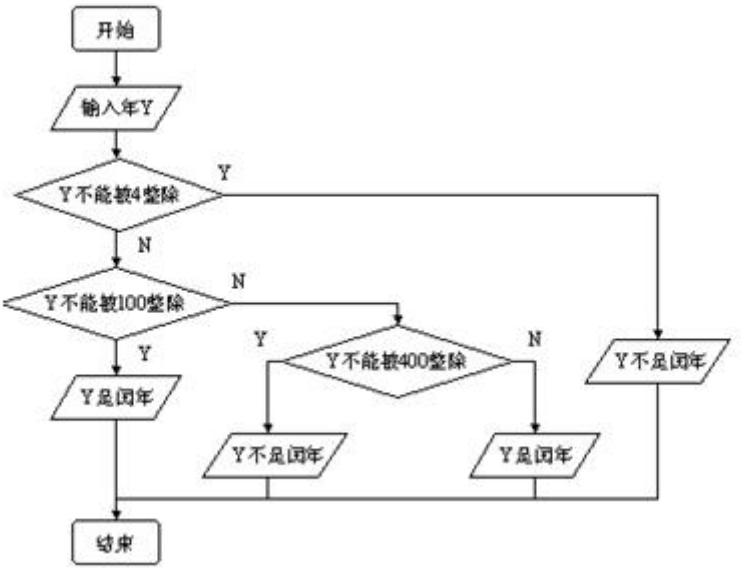
（2）逻辑“和”和“或”优先级相等，低于非逻辑

4.2.2 逻辑表达式

逻辑表达式的值应该是真或假，下面我们一起通过一个例子来了解下逻辑表

达式在选择结构中的作用：

我们知道在平年中 2 月份是 28 天，闰年中 2 月份是 29 天，那么怎样去区分平年和闰年呢？我们可以通过如下的流程图来进行判断：



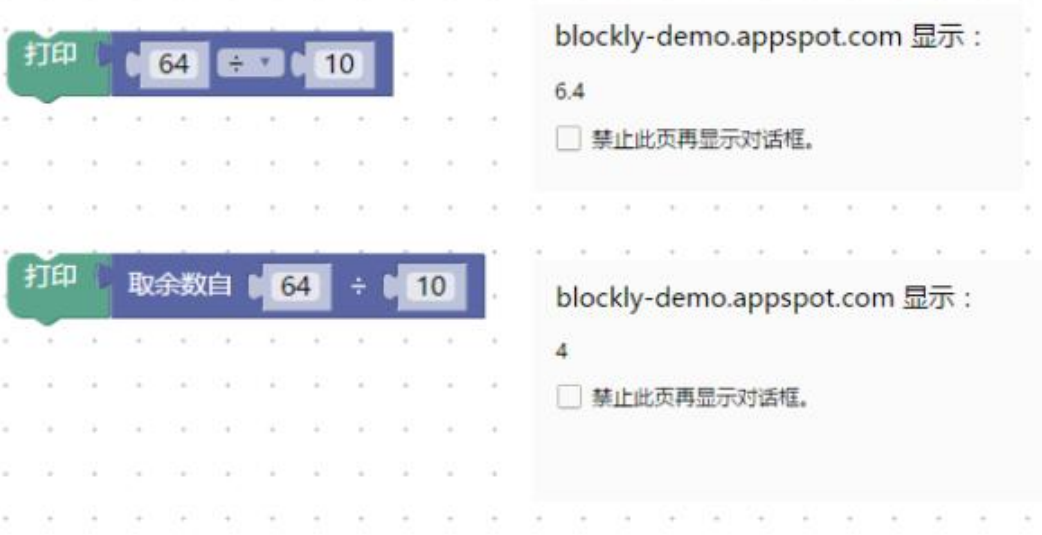
闰年的条件是符合下面二者之一：

①能被 4 整除，但不能被 100 整除

②能被 4 整除，又能被 400 整除

在 blockly 中有两种除法，一种是普通的除法，一种是取余除法，如下图所示：

普通除法直接进行计算，而取余除法将是将未整除的数保留下来，只取整数部分，并没有小数部分。



我们用符号/来表示除法，用%来表示取余，&&表示“和”，||表示“或”，！表示“非”。

那么闰年就可以这样来表示：

（year % 4 == 0 && year % 100 ！=0）|| year % 400 == 0

那么非闰年就直接在前面加一个！符号就好了

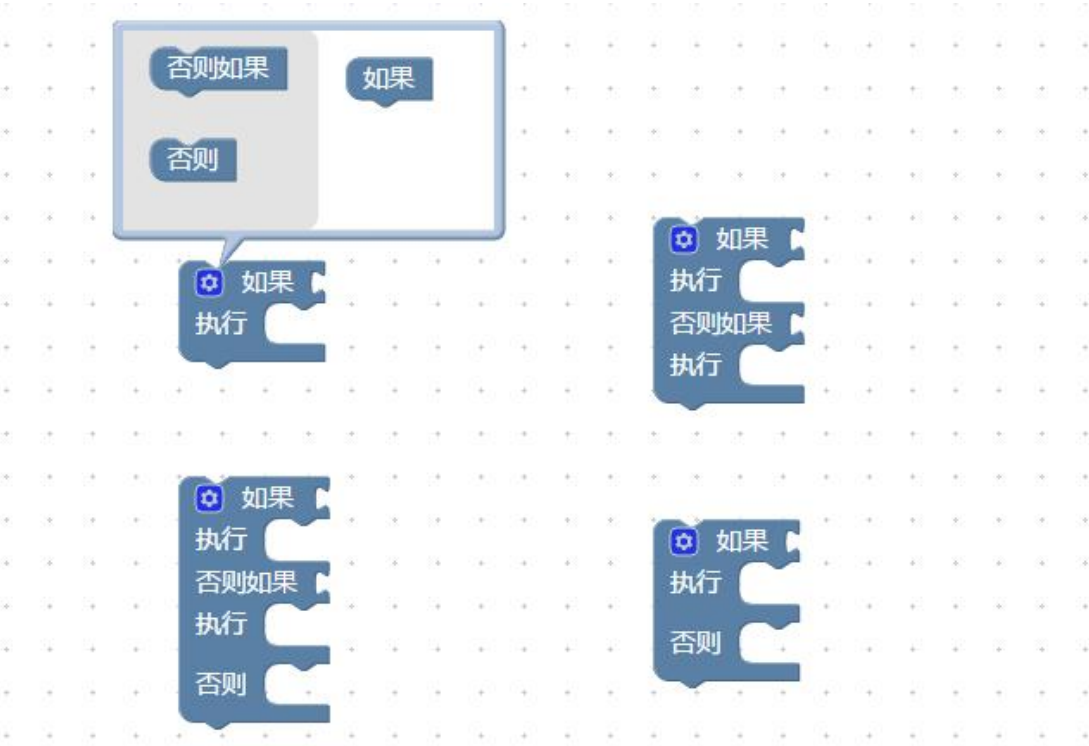
！（（year % 4 == 0 && year % 100 ！=0）|| year % 400 == 0）

转化为代码就是这样：

4.3 条件语句-if

4.3.1 选择结构之 if 语句

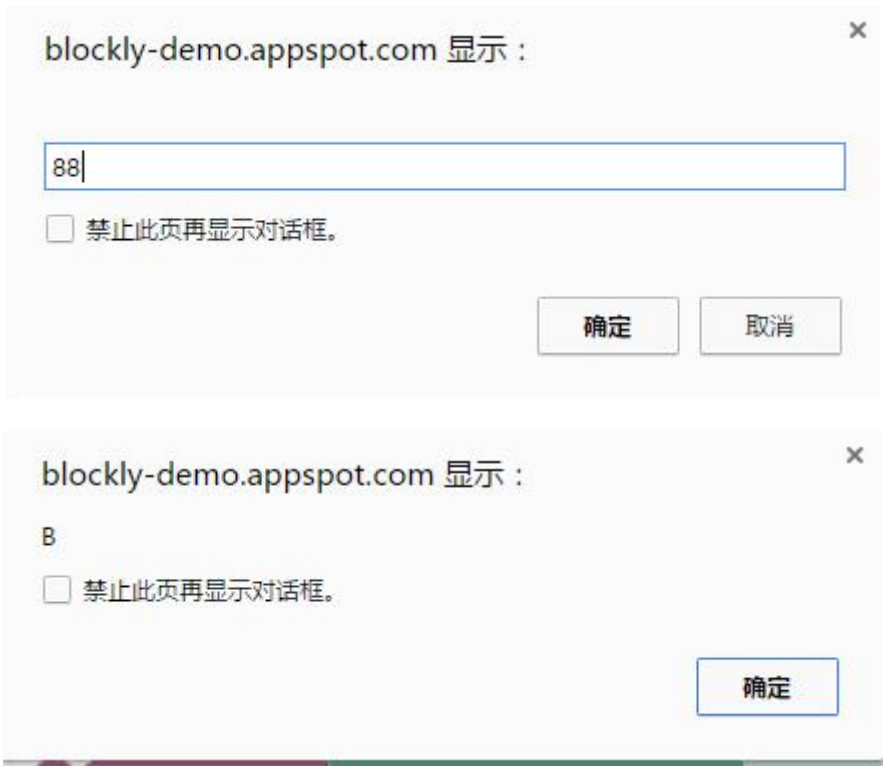
Blockly 中的 if 语句表示选择条件，如果满足条件就执行“.....”语句，在 if 的左侧有一个设置按钮，点开后可以添加 else if 和 else 语句，表示多重判断，你可以将左侧的模块儿拖到 if 下面就形成了其它三种语句。



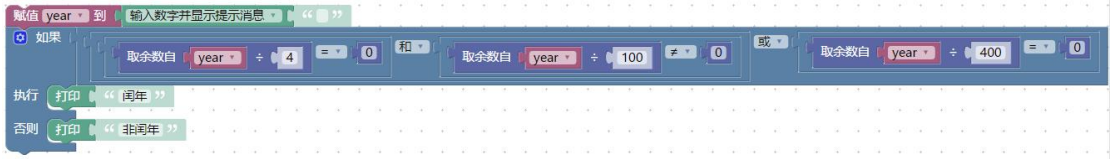
比如这个例子：

班里面要做一个成绩统计，成绩在 90 分以上就输出 A，在 80-90 分之间就输出 B，80 分以下输出 C：

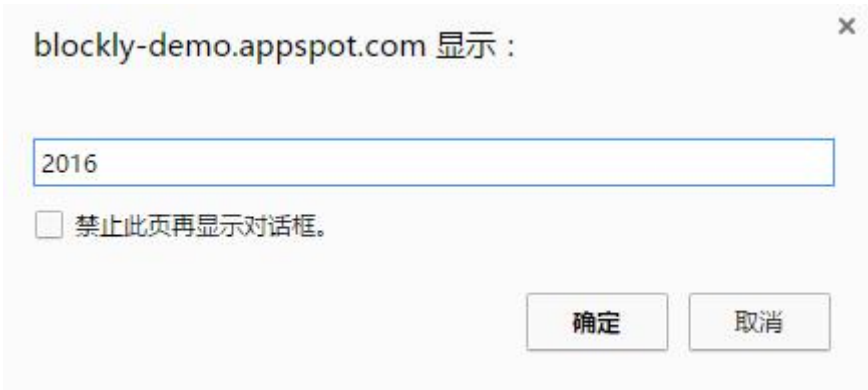
我们进行到这里这个程序你大致就应该明白了上面的代码究竟写的是什么内容，第一句话就是从键盘获取一个分数赋值给 score 这个变量，如果 90 < score < 100 那么就输出 A，在 80 < score <90 之间输出 B，其余的输出 C。



我们把刚才闰年的那个例子用编程来实现：



运行结果如下：



是不是挺简单的呢？下面我们将前面学习的东西巩固下，并将训练的难度进行升级：

①创建两个变量，并比较这两个数的大小。

②创建三个变量，分别通过键盘复制，并将这三个数由小到大输出。

解析：

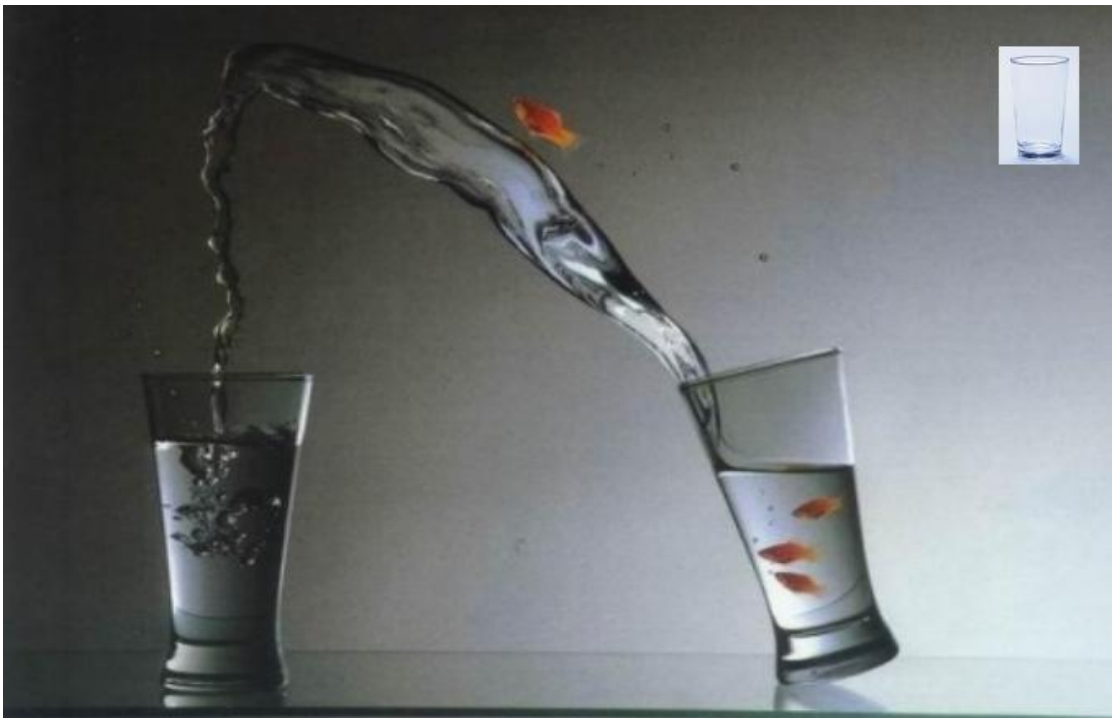
①我们比较两个数的大小时，有三种情况，大于小于和等于，因此就要用到 if 条件语句进行两次判断，剩下的一种情况直接用 else 即可，

“to result append text”这个语句是字符串追加的意思，就是我们不仅仅只输出一个数字，而是输出整个结果，比如 2>1，2=2，2<3 这样的结果。

输出结果如下所示：

②三个数比较大小时，情况就比较复杂了，还要从小到大进行输出，这就不仅仅需要进行比较了，而且还需要进行排序，这就需要了解在程序中怎样去交换两个变量，我们来举一个现实中的例子，现在我们有两杯水，我们怎样去交换这两杯

水呢？



答案是显然的，我们需要借助第三个空杯，我们把他们编号 A，B，C。

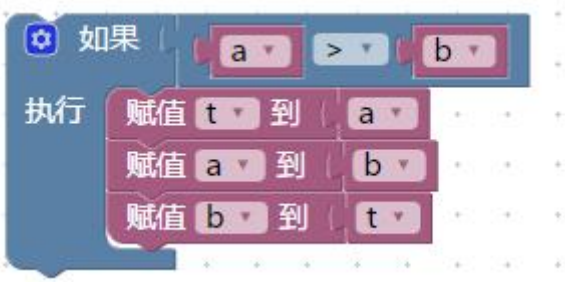
那么交换 A，B 两杯水，需要以下步骤：

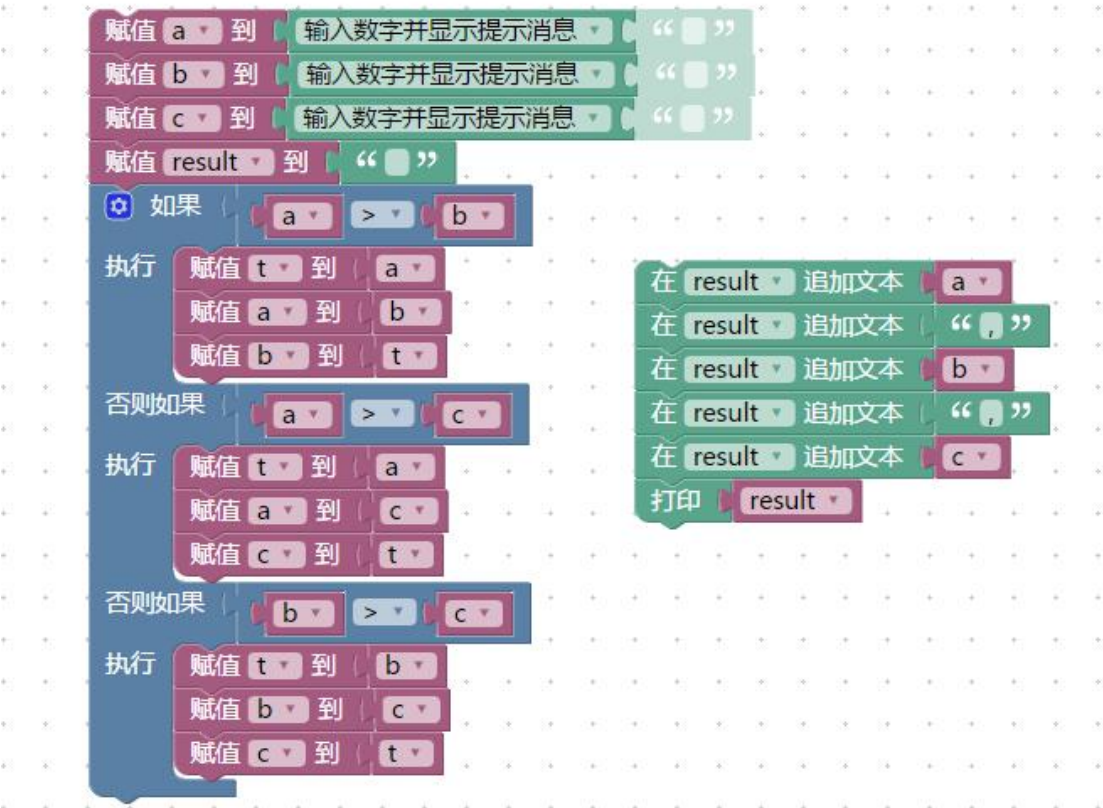
（1）把 A 中的水倒入 C

（2）把 B 中的水倒入 A

（3）把 C 中的水倒入 B

我们的程序也是这样，交换两个变量也需要借助第三个变量，剩下的就简单了，只要两两比较就可以了，因为要从小到大输出，所以只要前者大于后者将两个数交换就可以了。





4.3.2 选择结构的嵌套-if 嵌套

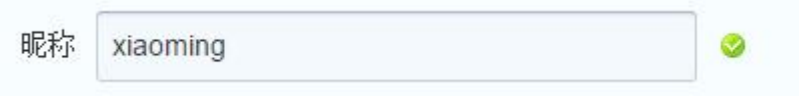
if 嵌套语句实际上就是在 if 语句里面还有 if 语句，实现了嵌套，我们刚才讲的平年闰年的问题其实也可以用嵌套的方法来解决：

首先判断 year 能否被 4 整除，如果能被 4 整除，再判断能否被 100 整除，这时可能还不是闰年，还要再进行判断，如果能被 400 整除那就是闰年，不能就是平年。

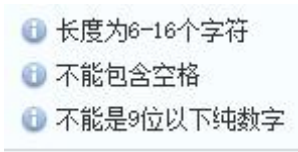
4.4 选择结构在生活中的应用

其实我们在平时上网的时候网页中存在着很多的选择结构，比如说登陆注册界面，我们以 qq 的登陆界面为例子，如下图所示：

①qq 中的昵称是可以随意的，你可以和别人的昵称是相同的，所以在昵称这栏我们在后台只需要判断是否存在非法字符就可以了，如果存在非法字符就在后面显示一个小“×”，如果昵称合法，就显示一个对号“√”。



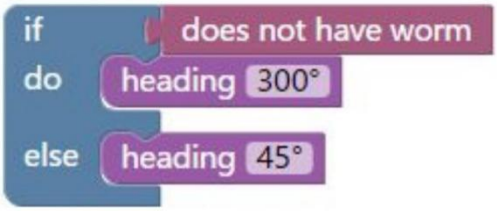
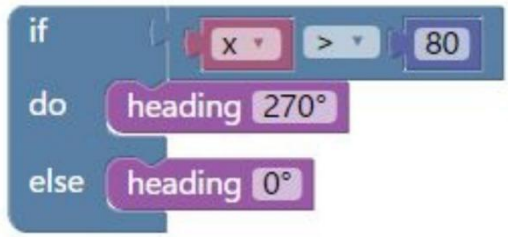
②qq 中密码的判断较为复杂，它分为三个等级，每输入一个字符后程序都会对其进行判断，如下图所示，每次判断后合格的项就会打勾，而且两次密码必须是一致的，判断是在第二次输入密码确定的。

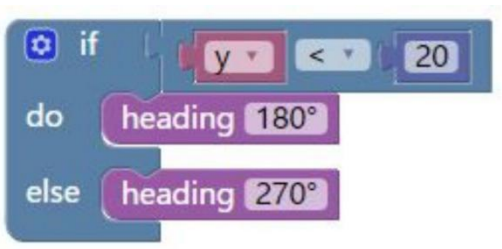


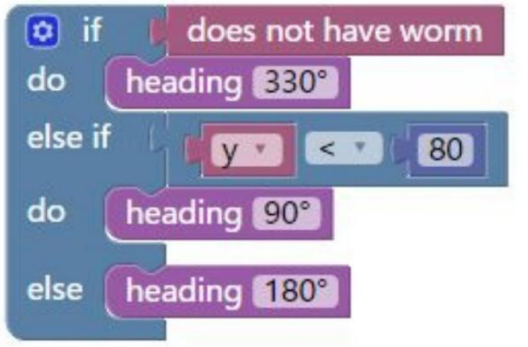
③生日、所在地这些都是需要进行选择的，我们需要在后台将所有的信息输入到选择文本中去，这样用户只要直接进行选择就可以了。

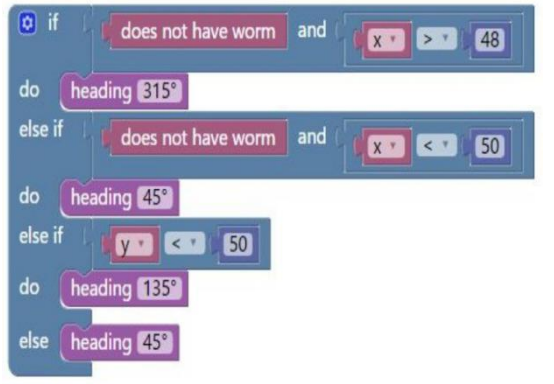
④当所有的信息全部填写完成，还要进行一次大的判断，就是所有条件全部满足时，即所有条件全都为真时，就会显示注册成功，否则就会弹出必须还要填写什么必要信息。

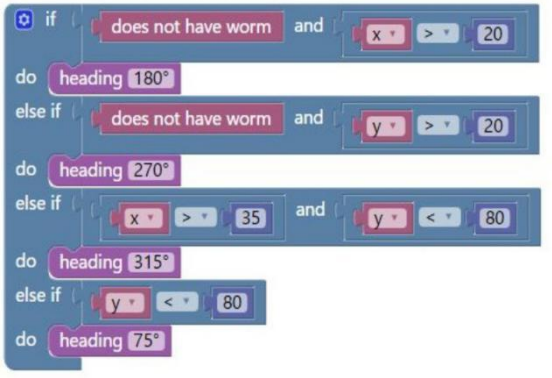
理解嵌套循环了么，现在我们再来回顾开篇的例子，看看你能否做出答案。参考答案如下：

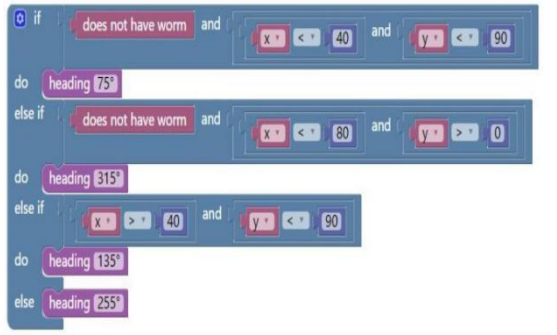
关卡	答案
	
	











课后练习

1. 写出下面各逻辑表达式的值。设 a=3, b=4, c=5。

(1) a+b>c && b==c

(2) a || b+c && b-c

(3) !(a>b) && !c||1

(4)!(x=a) && (y==b) && 0

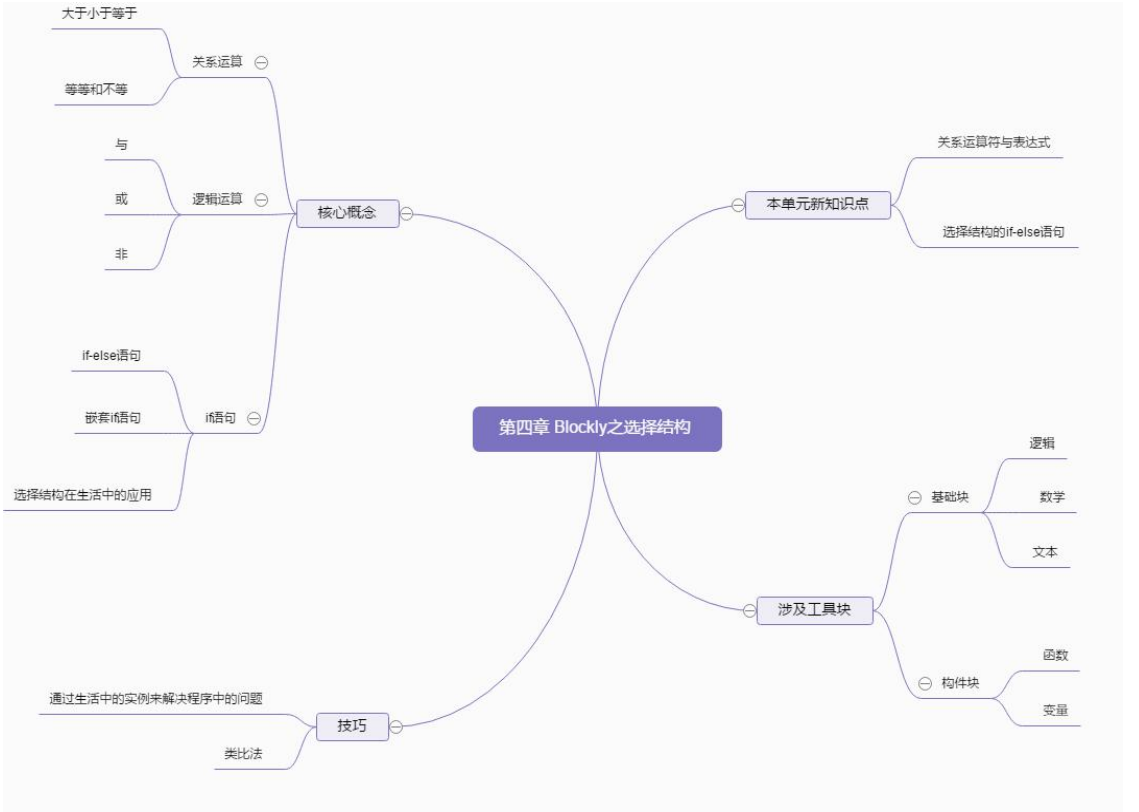
(5)!(a+b)+c-1 && b+c/2

2.给一个不多于五位的正整数，要求：

(1)求出它是几位数；

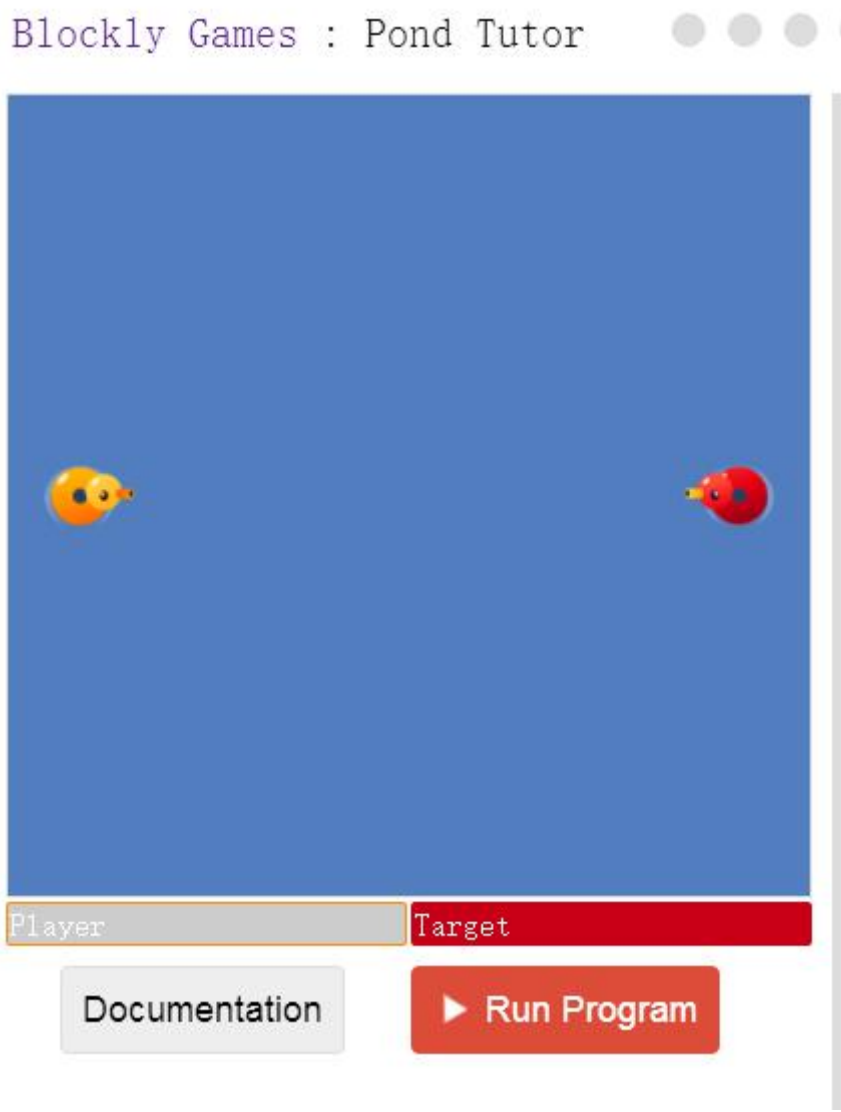
(2)分别阀引出每一位数字；

(3)按照逆序打印出各位数字，例如原数为 321，应输出 123。


第四章 Blockly之选择结构
核心概念
关系运算
大于小于等于
等等和不等
逻辑运算
与
或
非
if语句
if-else语句
嵌套if语句
选择结构在生活中的应用
本单元新知识点
关系运算符与表达式
选择结构的if-else语句
涉及工具块
基础块
逻辑
数学
文本
构件块
函数
变量
技巧
通过生活中的实例来解决程序中的问题
类比法

第五章 循环结构

在这里，我们将介绍一个新游戏--Pond Tutor



在 Pond Tutor(https://blockly-games.appspot.com/pond-tutor)这个游戏中，我们将扮演黄色的鸭子，通过不断的发炮弹去攻击红色的鸭子，当红色的鸭子血条减为 0 时则玩家获胜。

在这个游戏中为我们提供了四个功能模块：

Swim 模块：向给定方向游动；

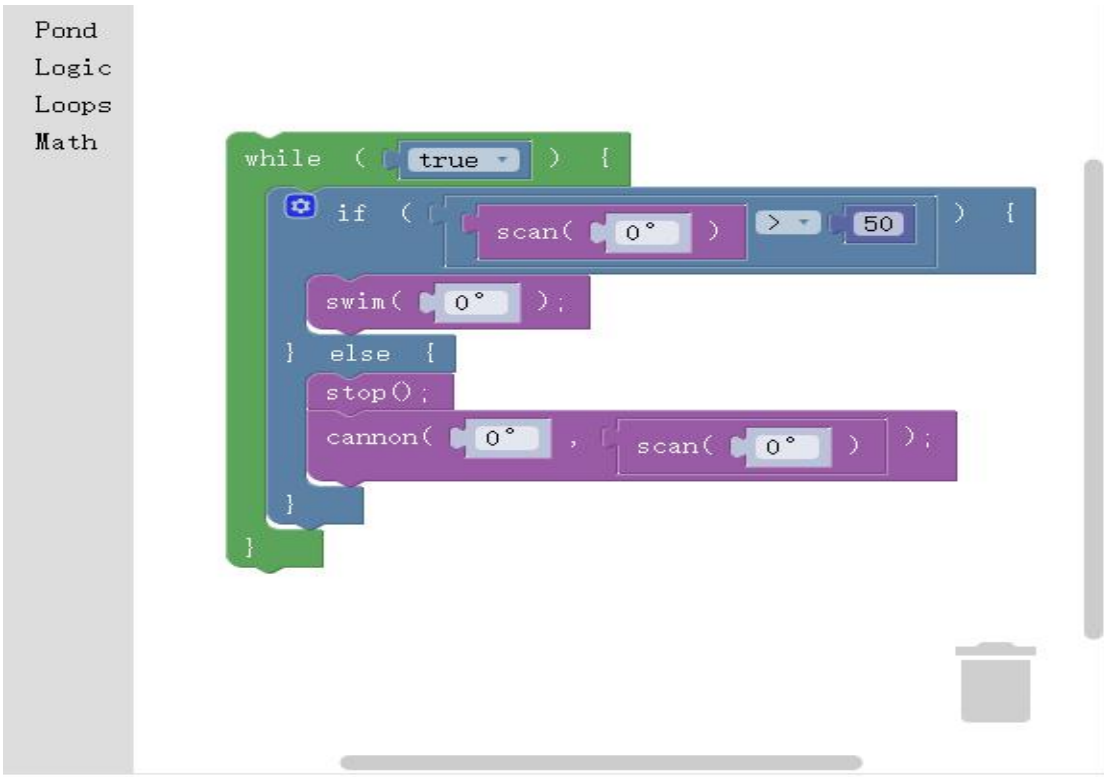
Scan 模块：向给定方向扫描，扫描到敌方时返回二者相距的距离；

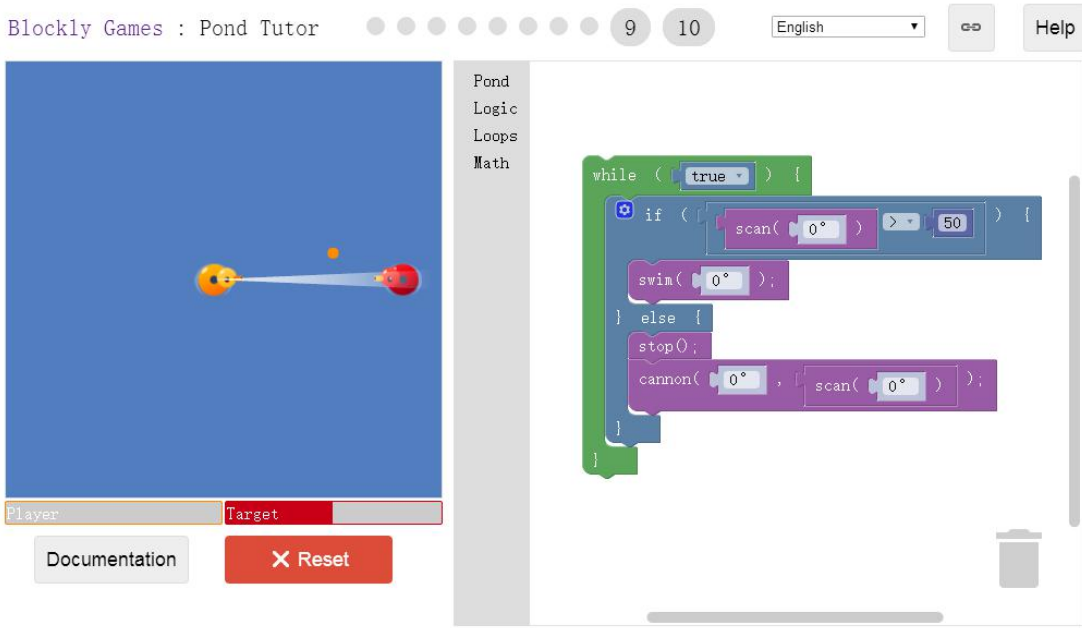
Cannon 模块：向给定方向和距离发射炮弹；

Stop 模块：配合 swim 模块，使我方停止。

通过结合这几个模块，我们可以想到：首先我们需要朝敌方游动，同时不断的进行扫描；当扫描到敌方时，我们应该停止，同时用扫描到的距离设定炮口的攻击距离。

这里唯一的问题是如何重复不断的进行扫描，所以也就牵出了我们接下来要讲的问题--循环。





循环是一种基本的程序结构，凡是需要通过不断重复执行才能得到答案的许多问题中需要用到循环控制。循环结构是结构化程序设计的基本结构之一，它和顺序结构、选择结构共同作为各种复杂程序的基本构造单元。因此熟练掌握选择结构和循环结构的概念及使用是程序设计的最基本的要求。本章将介绍四种形式的循环，都是程序中基本的用于表示重复动作的结构。在本章学习结束后，学生应当具备的能力有：分析一个循环的重复执行次数和终止条件，并能在自己的程序中应用到重复次数, 重复, 步长, 列表循环循环。

5.1 基本概念

迭代:通过包括重复序列允许通过一个简短的程序来代替一系列的重复的步骤

程序中重复执行的一段指令叫做**循环**。

每一个循环由两部分组成:

条件： 控制循环重复次数

循环体： 循环执行时始终运行的代码段。

我们称一个无休止运行的程序为**死循环**，下图所示的就是一个典型的死循环

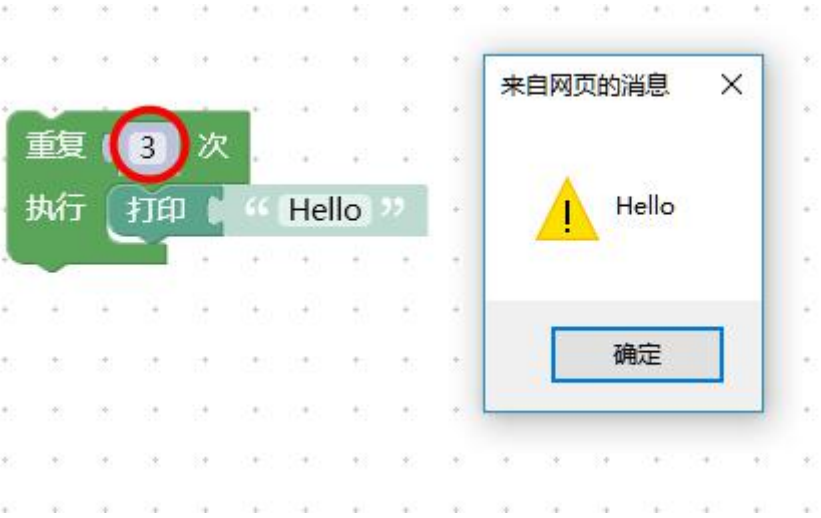
块。死循环将会阻止一个程序完成其执行，所以含有死循环的程序通常是不可取的。 在浏览器内，一个死循环甚至可以导致程序对鼠标点击失去响应或无限输出对话框，因此要避免出现死循环，控制循环执行次数的程序必须编写正确。

5.2 重复次数模块

重复次数模块用来实现计数循环，通过在模块中修改次数来规定重复执行的次数。

例如我们要循环三次输出“Hello World”，则只要在重复次数模块的数字块中

写入“3”即可。



5.3 重复模块

重复模块中包含了两种不同类型的循环模式：重复-当和重复-直到。

重复-当模块用来实现“当型”循环结构，只要程序的执行条件为真，就会重复执行语句的一系列程序。在未运行前，实际的重复次数是未知的。

它需要与逻辑语句组合使用，当条件满足时它将重复不断的执行直到条件不满足，所以可能出现的情况是：由于一个逻辑错误，循环将相应代码段重复运行无数次，这对初学者来说是很常见的。

当检查一个循环条件时循环将会执行。如果循环条件为真，将执行循环体中的代码并重复执行这些步骤，在每次循环前将额外检查循环条件。条件为假时退出循环。

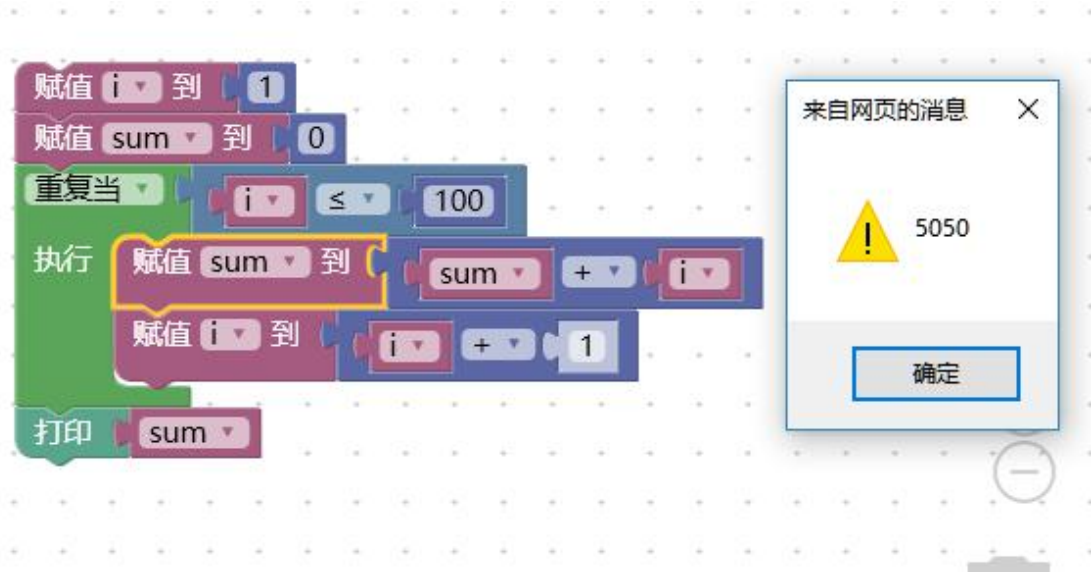
循环程序都包涵了三个内容：

*一个清晰的起始条件

*一个循环条件，指示程序是否应该继续执行循环部分

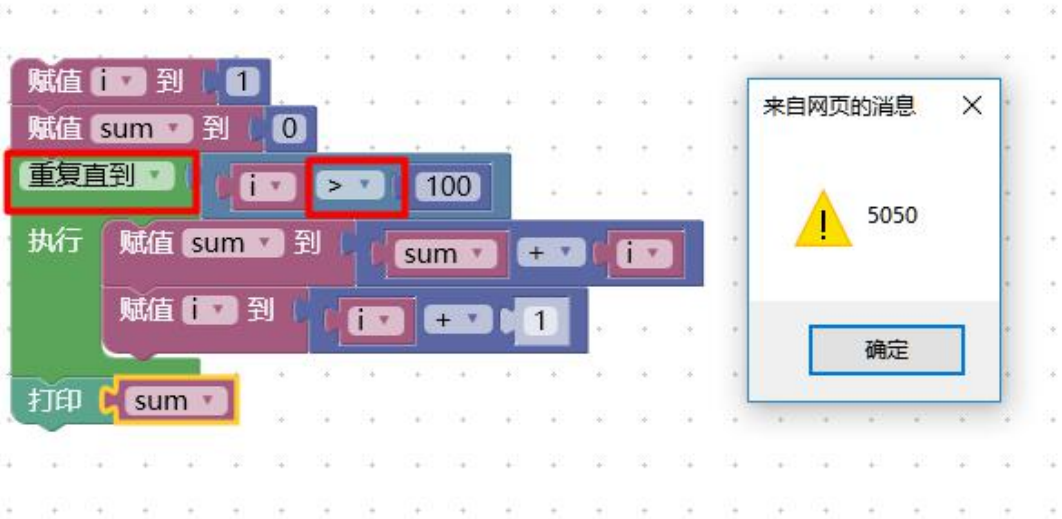
*一个条件改变最终导致循环条件变为假。

例如，我们使用重复-当模块来实现一个非常常见的问题：从 1 一直加到 100，$\sum_{n=1}^{100} n$。对于这问题同学们一定不陌生，甚至能有多种不同的技巧去解这个问题。但在这里，我们使用循环就可以用最笨的方法：从 1 一个一个的加起来，一直加到 100。



而对于重复-直到这个模块来说则与重复-当模块刚好相反：它是当条件满足时就结束循环。

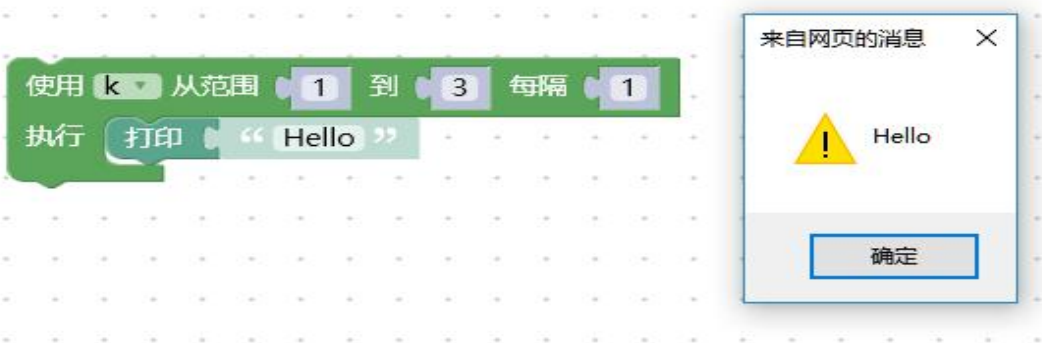
所以同样针对上面从 1 加到 100 的例子，我们只需要将条件 i<=100 改为 i>100 即可。



5.4 步长循环模块

步长循环模块使用最为灵活，不仅可以用于循环次数已经确定的情况，而且可以用于循环次数不确定而只给出循环结束条件的情况，它完全可以替代重复模块。

在固定增量下有着固定循环次数的一段程序使用步长循环模块为宜。下面的循环将输出三次单词“Hello”。



如果你已经接触过一些高级程序设计语言如：C/C++/Java 等，你会发现步长循环模块其实就是其中的 For 循环语句。

步长循环模块中“使用”后的 k 为循环的控制变量，这个 k 可以自己命名并且可以在循环块内充当变量使用；“从范围”后的数字代表循环控制变量 k 的初始值；“到”后的数值表示循环的结束条件:当 k 大于这个值时退出循环；“每隔”后的数字为 k 的步长，表示每循环一次，k 增加多少。以上所有的数值都是可以根据自己的实际需要进行修改的。

下面我们大家都很熟悉的一道应用题来使用步长模块：一球从 100 米高度自由落下，每次落地后反跳回原高度的一半，再落下。求它在第 10 次落地时，共经过多少米？第 10 次反弹多高？

在这里我们并没有从第一次下落开始就使用循环，而是以落点-上升-落点为过程进行的，由于落点-上升这一过程与上升-落点的过程所经过的距离相同，所以总路程直接加二倍的反弹距离;而反弹距离直接除 2 即可直接变为原来的一半。

5.5 列表循环模块

列表循环模块是对列表（list）中每一个元素进行循环迭代的模块，也可以称为是对列表元素的遍历。所谓列表，是相同数据类型的元素按一定顺序排列的集合。使用列表循环模块的循环在达到列表的最后一个值后将自动结束。所以使用列表循环模块是不太可能错误地写成死循环的，或许唯一可以的方式是为循环使用一个含有无限元素的列表！

如下图所示，通过使用列表循环模块可以输出列表 numList 中的每一个元素。

虽然使用重复、步长模块也能实现循环输出每一个元素，但用这种方法不但需要知道列表的长度，而且无法像列表循环模块一样直接就定义好了变量去代表列表的元素。

例如求一个列表中的最大元素，我们可以用列表循环模块来实现：

在这里我们定义了一个变量 tmp 去保存最大的值，将 tmp 初始化为 0 保证 tmp 刚开始为最小，之后用列表循环模块循环拿出列表中的元素不断的与 tmp 相比较，一旦当前值大于 tmp 则修改 tmp 为这个值。最终在 tmp 中的值将是最大的值。

5.6 中断/继续模块

中断模块可以用来从循环体内跳出循环体，即提前结束循环，接着执行循环下面的语句。

中断模块适用于当我们不知道循环次数，只有在程序执行过程中满足一定条件的情况下要结束循环的时候。

我们现在尝试使用循环和中断模块来写一个程序：输入一个数，判断这个数是否为素数。

在这里我们让 m 被 2 到根号 m 除，如果 m 能被 2 到根号 m 之中的任何一个整数整除，则提前结束循环，此时 i 必然小于或等于 k（即根号 m）；如果 m 不能被 2 到 k（即根号 m）之间的任一整数整除，则在完成最后一次循环后，i 还要加 1，因此 i=k+1，然后才终止循环。在循环之后判别 i 的值是否大于或等于 k+1，若是，则表明未曾被 2 到 k 之间任一整数整除过，因此输出“是素数”。

继续模块其作用为结束本次循环，即跳过循环体中下面尚未执行的语句，接着进行下一次是否执行循环的判定。

继续模块与中断模块的区别是：继续模块只结束本次循环，而不是终止整个循环的执行。而中断模块则是结束整个循环过程，不再判断执行循环的条件是否成立。

如把 100-200 之间的不能被 3 整除的数输出来：

当 n 能被 3 整除时，执行继续模块，结束本次循环（即跳过 print 模块），只有 n 不能被 3 整除时才执行 print 模块输出数字。

5.7 嵌套循环

一个循环体内又包含另一个完整的循环结构，称为循环的嵌套。内嵌的循环中还可以嵌套循环，这就是多层循环。各种语言中关于循环的嵌套的概念都是一样的。四种循环（重复次数,重复,步长,列表循环）可以互相嵌套，当循环嵌套在循环中，一个程序可以创建一个重复的重复。

5.7.1 内循环和外循环

当使用嵌套循环时，内外循环扮演着不同的角色：

外循环是先开始后结束的运行顺序。它在程序运行中开始一次结束一次。内循环是比外循环后开始先结束的运行。它可以反复地重复运行。

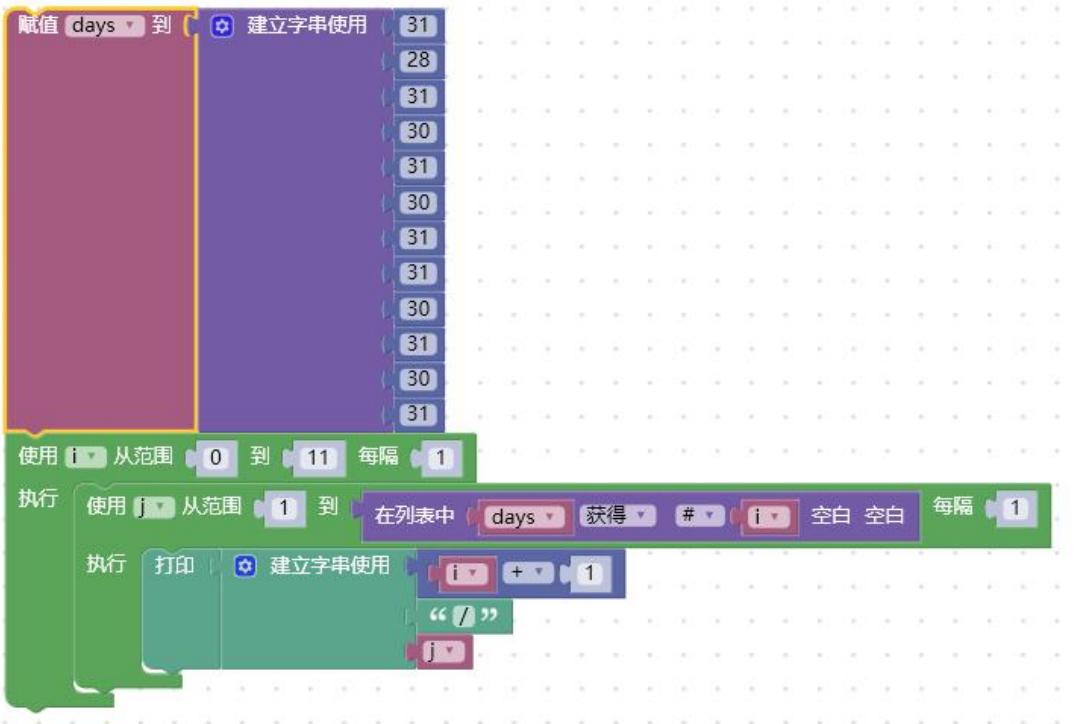
内循环快速运行，每一次外循环,都会完成一个完整的内循环。外循环循环的偏慢。

内部和外部的循环类似时钟指针的运转。例如，一个时钟的秒针就像内部循环同时分针就像外循环.对于每一次分针的转动,秒针都需要转一圈,转动 60 整秒.

当分针转动一整个小时时,秒针将转动 60*60 = 3600 秒。里程表和日历的都是用这样相似的工作方式。

5.7.2 非独立的内循环

有时，内部循环的工作方式取决于外部循环的哪一步运行。例如，创建一个程序打印在一年中所有的日期，您可以使用嵌套循环去打印月份的每天，但内部循环的次数取决于外部循环，这是因为每个月有不同的天数。



在这里，内部循环重复的是不同的次数，这取决于它在哪个月份上，因为它的结束条件是 j<days[i].

对于月份，我们绘制了一个依赖于存放每一月的天数的数组的内循环,但是对于给一个非独立内循环构思一个规则需要更加的泛化:这样一个程序员需要考虑使用的特殊情况,同时要尝试泛化这个规则。

由于内部循环和外部循环的关系，嵌套循环可以使具有挑战性的程序正确。

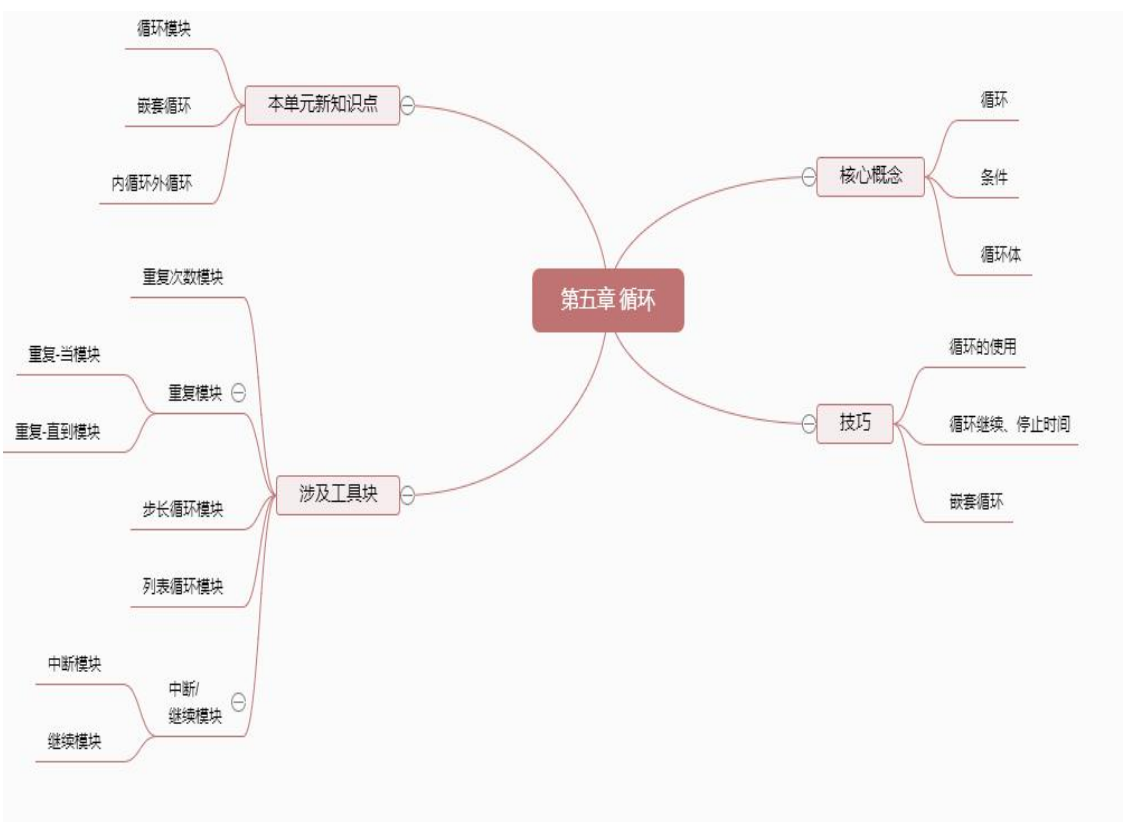
设计嵌套循环的一个很好的策略是在一个例子中仔细地找出你想要的运行效果，然后找到一个方法来归纳它。

课后练习

1. 写一个判断素数的函数，在主函数输入一个整数，输出是否是素数。

2. 设计定义一个自己的工具块。

3. 编写函数，给出年、月、日，计算该日是该年的第 n 天，并尝试将其导出的代码在其他语言环境中调试运行。


第五章 循环
本单元新知识点
循环模块
嵌套循环
内循环外循环
核心概念
循环
条件
循环体
技巧
循环的使用
循环继续、停止时间
嵌套循环
涉及工具块
重复次数模块
重复模块
重复-当模块
重复-直到模块
步长循环模块
列表循环模块
中断/
继续模块
中断模块
继续模块

第六章 Blockly 的进阶

6.1 模块化程序设计

一个较大的程序一般应分为若干个程序模块，每一个模块用来实现一个特定的功能。所有的高级语言中都有子程序这个概念，用子程序实现模块的功能。比如在 C 语言中，子程序的作用是由函数完成的，一个 C 程序可由一个主函数和若干个函数构成，由主函数调用其它函数，其他函数也可以相互调用，同一个函数可以被一个或多个函数调用任意多次。在 Blockly 中，也支持函数的定义和使用。

在程序设计中，常将一些常用的功能模块编写成函数，放在函数库中供公共选用，所以要善于利用函数，以减少重复编写代码的工程量。

6.1.1 函数的定义形式

在 C 语言和别的语言中，函数的一个明显的特征就是使用时带括号()，必要的话，括号中还要包含数据和变量，称为参数(Parameter)，参数是函数需要处理的的数据。根据参数的有无，可将函数简单的分为无参函数和有参函数。

这段话对于没有接触过 C 语言或其他编程语言的同学可能比较抽象，不过不必担心，接下来我们会通过 Blockly 向您详细的解释。

(1)无参函数的定义

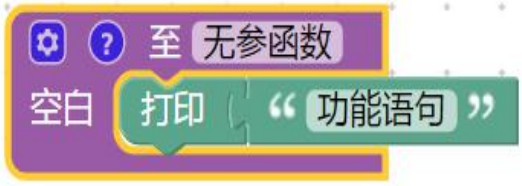
上图是我们从 Blockly 工具箱中拖出的一个函数块，其中:对函数进行参数的设置，无参函数不需要用到此选项

做点什么 这里输入函数的名称

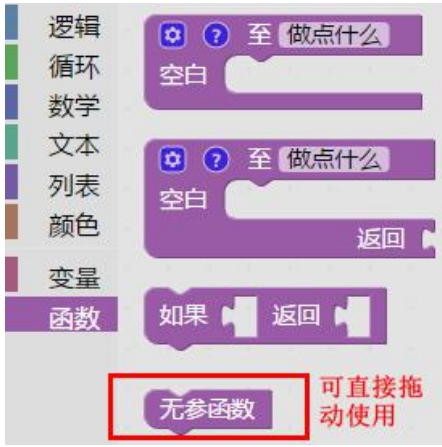
空白 在这里添加函数所实现的功能语句

对函数的功能进行描述

对工具箱中的 Blockly 有了简单的了解之后，尝试动手设计自己的函数。



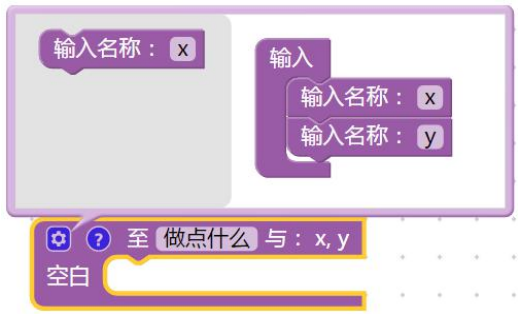
上图是一个简单的无参函数，它的函数名就叫“无参函数”，当你从工具箱拖动一个块在编辑区的同时，在工具箱中的函数选项卡中会生成一个对应的函数块，当再用到此函数时，就可以像使用其他工具箱中的块一样直接使用。



(2)有参函数的定义

与无参函数不同，有参函数需要在  中对参数进行设置，拖动 

至右侧“输入”中，并对参数进行命名即可。



同时，像无参函数一样，你也可以在左侧工具箱“函数”选项卡中看到对应的块。

(3) 函数的返回值

函数的另外一个明显的特征就是返回值，既然函数可以进行数据处理，那就有必要将将处理结果告诉我们，所以很多函数都有返回值，所谓的返回值就是函数的执行结果。

当创建的函数需要返回值时，可直接从工具箱中拖动自带返回值的函数块，可见工具箱中生成的函数块左侧带有凸起的连接。

这种方式生成的函数，只有当函数执行完成后才会返回值，如果在函数执行过程中就已经产生了想要的结果，也可以拖动 

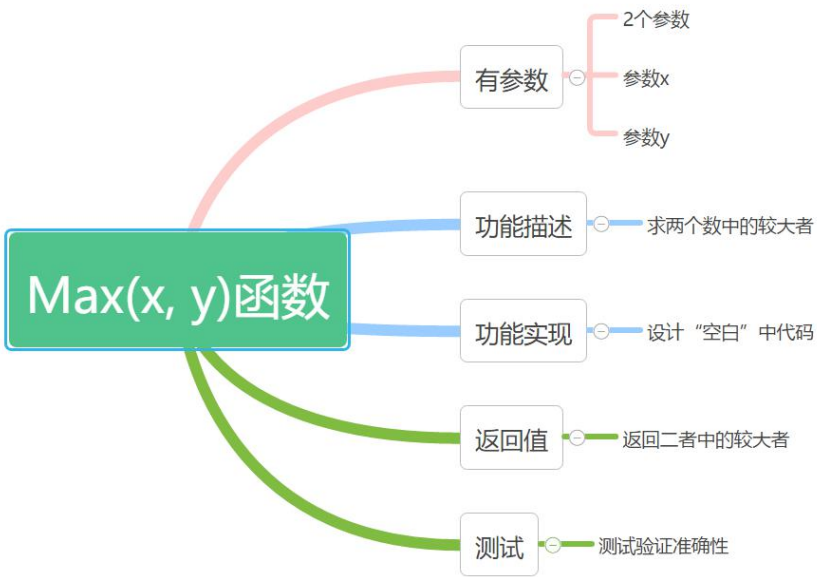
结束正在执行的函数，

并返回执行结果。

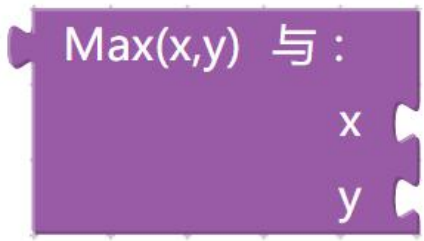
注意：

在使用时，只有当函数最初设计有返回值时才有返回值，否则只是简单的结束正在执行的函数。

下面来做一个小练习，设计一个求 x，y 中最大者的函数，名为 Max(x，y)。

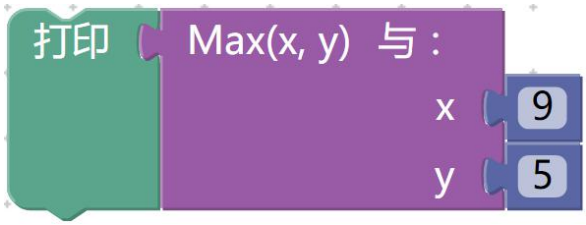


可以按照思维导图，一步一步进行 Max(x，y)函数的设计。在进行程序设计时，无论你是简单的学习，还是进行复杂的开发，在动手之前，一定要对你所设计的程序有一个良好的规划，磨刀不误砍柴工，好的习惯很重要，它可以加速你编程水平的提升，提高你的编程效率。



当你设计完成后，剩下的就是测试验证程序结果了，测试时不需要很复杂，如果可以，最简单的就是使用 

。



6.2 Blockly 开发工具

在前几章的学习中，每章开篇我们的小游戏环节，每个小游戏虽然是可视化编程，和我们学习的 Blockly 很像，但是又有所不同，而这些不同由何而来?这就是我们本章所讲的重点，Blockly 开发工具(Blockly Developer Tools)，通过它用户可以自定义新块，这使得 Blockly 可扩展性大大提升，同时也是 Blockly 的灵活和强大之处。

本节面向希望在 Blockly 中创建新块的开发人员。它的基本要求是，有一个

可以编辑的 Blockly 的本地副本，大体上熟悉了 Blockly 的使用，并且对 JavaScript 有一个基本的理解。

Blockly 带有大量的预定义块，从数学函数到循环结构的一切，然而为了与外部应用程序接口，必须创建自定义块以形成 API。 例如，当创建绘图程序时，可能需要创建“绘制半径 R 的圆”块。而在大多数情况下，最简单的方法是找到

一个已经存在的真正相似的块，复制它，并根据需要修改它。

第一步是创建一个块；指定其形状，字段和连接点。使用 Blockly Developer Tools 是编写此代码的最简单的方法，或者，可以在学习 API 之后手动编写该代码，高级块可以响应于用户或其他因素而动态地改变它们的形状。

第二步是创建生成器代码以将新块导出为编程语言（例如 JavaScript，Python，PHP，Lua 或 Dart）。为了生成既干净又正确的代码，必须注意给定语言的操作列表顺序，创建更复杂的块需要使用临时变量和调用函数，当输入使用两次并需要缓存时，这是尤为重要的。

Blockly 开发工具是一个基于 Web 的开发工具，可自动完成 Blockly 配置过程的各个部分，包括创建自定义块，构建工具箱和配置 Web Blockly 工作区。

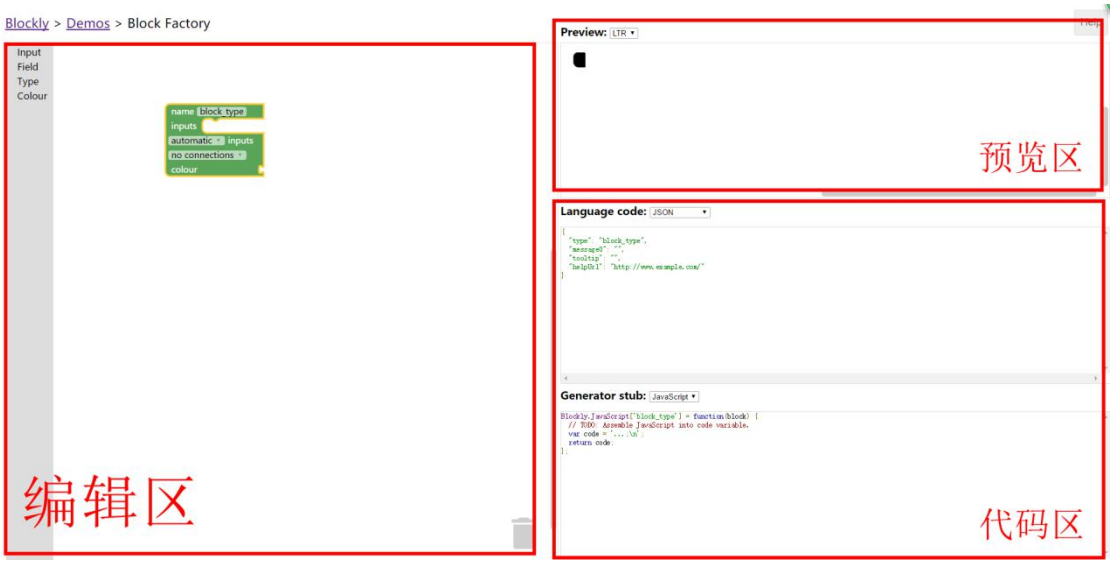
使用该工具的 Blockly 开发者进程包括三个部分：

1、使用块工厂和块导出器创建自定义块。

2、使用 Workspace Factory 构建工具箱和默认工作空间。

3、使用 Workspace Factory 配置工作空间（当前是仅限 Web 的功能）。

6.2.1 定义一个块



定义一个块需要使用到 Blockly 开发工具中的块工厂(Block Factory),块工厂主要分为三个区域：

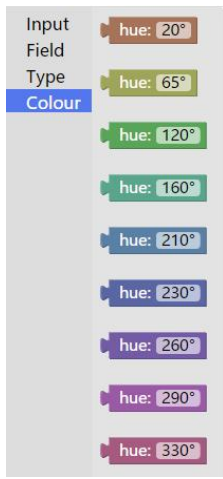
（1）编辑区：对新增块进行设计和编辑

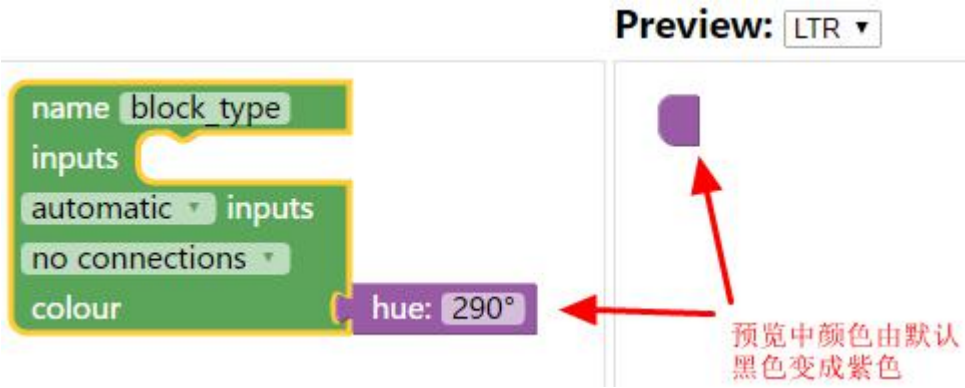
（2）预览区：对编辑区编辑的块进行实时预览

（3）代码区：代码区分为两个部分 Language code 和 Generator stub，其中 Language code 区指定和控制新增块所呈现的形状，Generator stub 区负责新增块所要生成的代码。

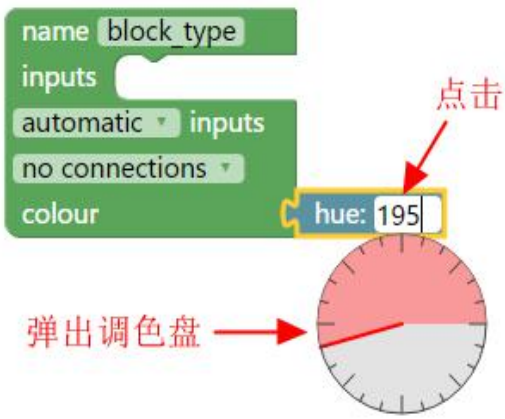
在编辑区的左侧，可以看到 4 个基本块，Input，Field，Type 和 Colour，它们是四个原料库，使用者可以从这些库中获取所需要的任意“原料”，来合成自己的新块。

先从最简单的介绍，颜色(Colour)块，它默认定义了九种基本颜色，直接将你想要的颜色拖到右侧，拼接到对应的 colour 的凹槽，便可立即在预览区看到新块的颜色。





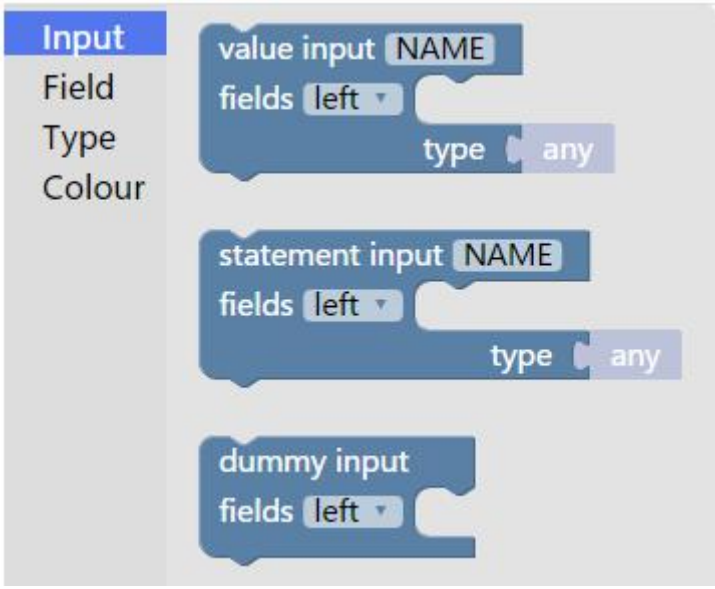
如果默认色彩中没有你想要的颜色，可以拖动任意色彩块到编辑区拼接完成后，点击色块中的数字，在色块的下方出现一个圆形的调色盘，转动调色盘，选择你喜欢的颜色。



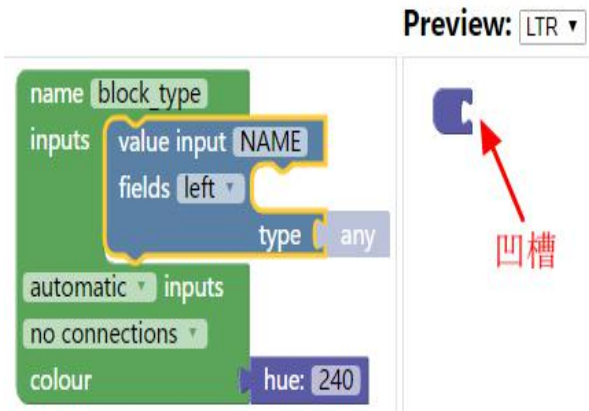
在 Blockly 中，同一类型的块通常采用相同的颜色，所以新块的颜色选择不能仅凭喜欢，还需要前后兼顾。

一个新块不仅需要有颜色，还需要与其他块进行衔接，这就需要设计新增块的输入和输出，它们将决定新增块的功能、属性和类别。

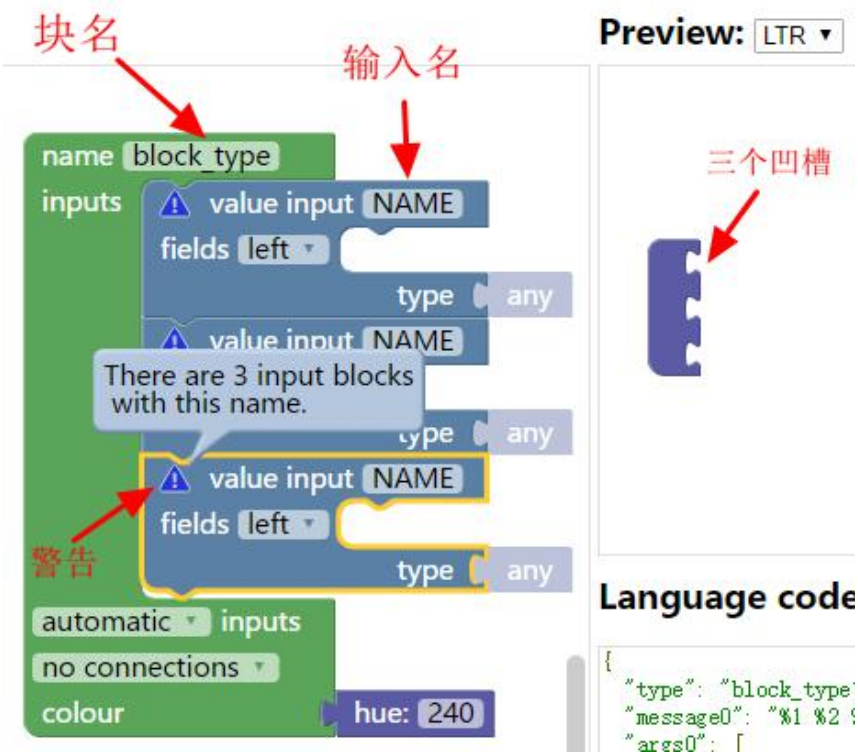
接下来看一看输入(Input)，这是新增块与其他块连接的接口之一。



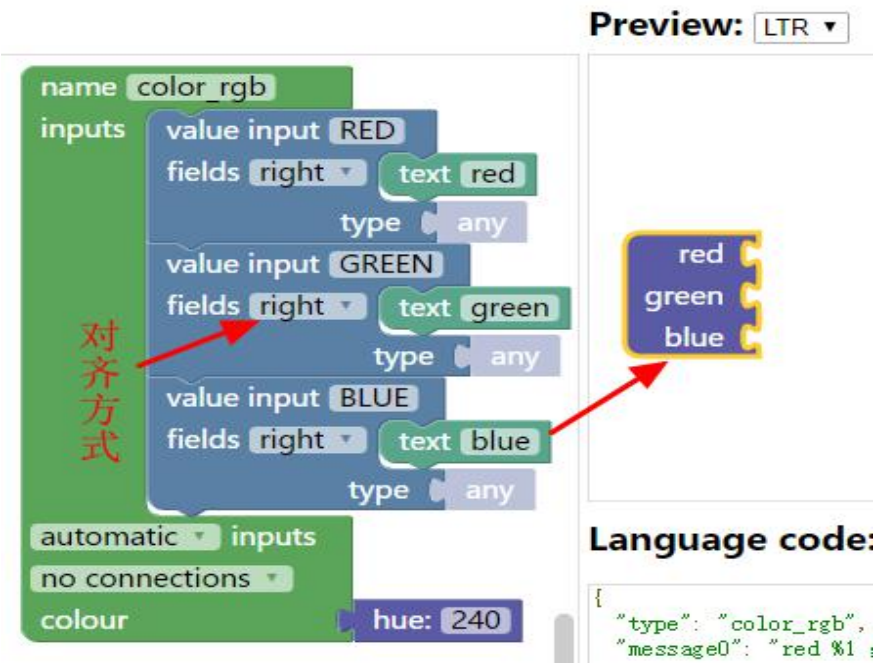
输入可以分为三种类型：值输入(value input)，声明输入(statement input)，模拟输入(dummy input)。首先以值输入为例，拖动值输入至右侧与 Inputs 连接，可看到预览区新增块多了一个凹槽:



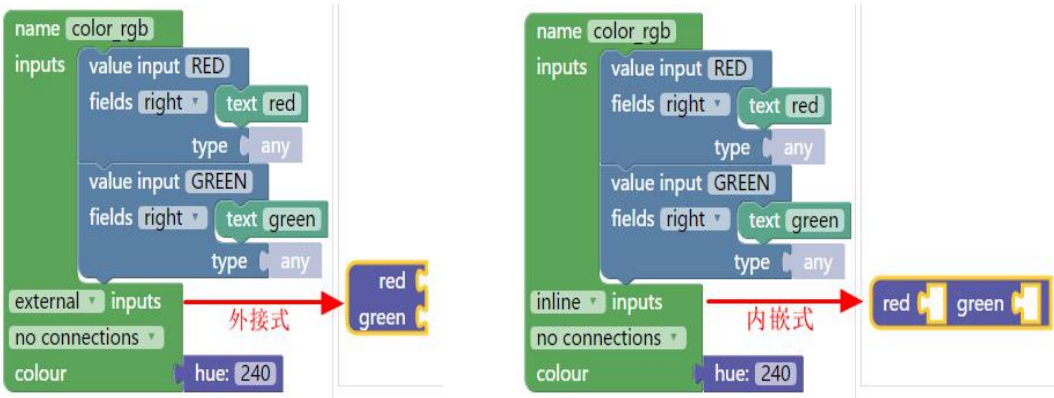
根据需要，使用者还可以添加多个输入值，但要注意多个输入值的名字不能相同，否则会出现警告，而且在后续调用的时候，也会冲突报错，新块名字也是如此，不能与其他块同名，就好比如果班里有两个学生名字一样，那老师点名提问的时候就有可能出现两个同学同时起立的尴尬。



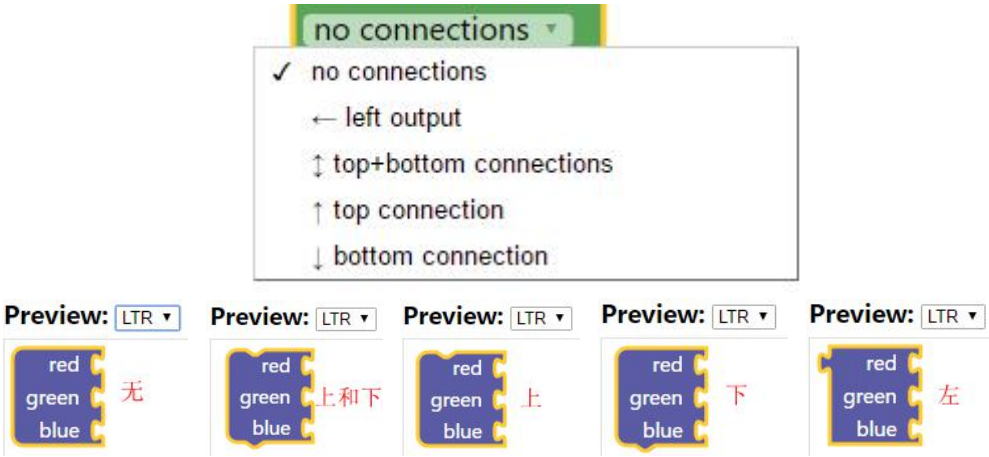
在值输入中还可以添加域(field)，比如加入最简单的文本域，即可在输入中提示对应的文本，域中的下拉选择框可设置文本的对齐方式。



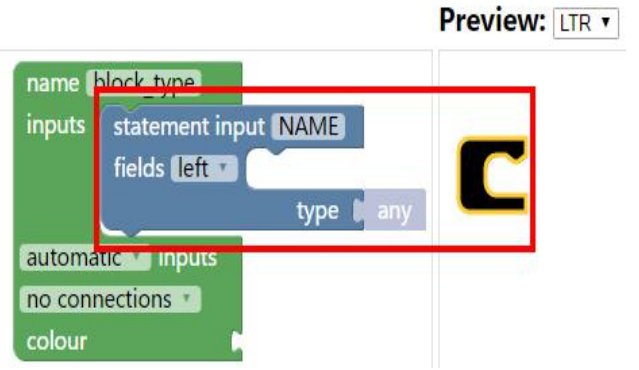
这些设置完毕，选择新块的输入方式，扩展式和嵌入式：



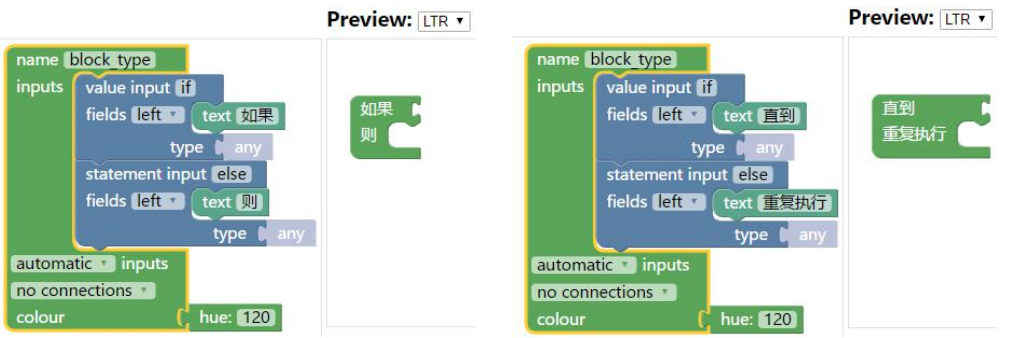
有了输入之后，别的块就可以很容易的通过凹槽加入到新块了，但是，这时另外一个值得考虑的问题又出现了，怎样将新增块加入到其他的块之中呢？我们有五种选择：



看完值输入之后，再一起来看一下另一个常用的输入类型，声明输入(statement input)，它通常用作条件控制和循环控制。



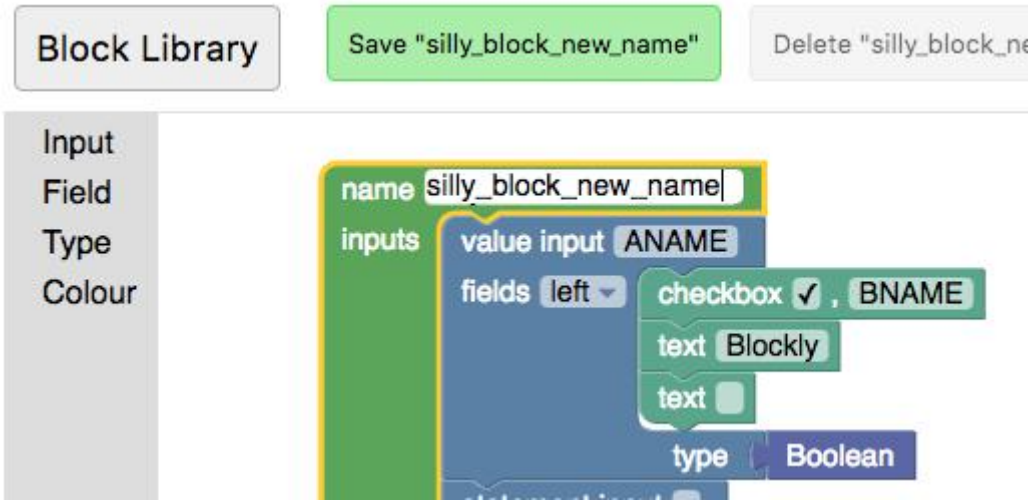
使用值输入和声明输入，可以很容易的设计出编程中常用的条件语句和循环语句：

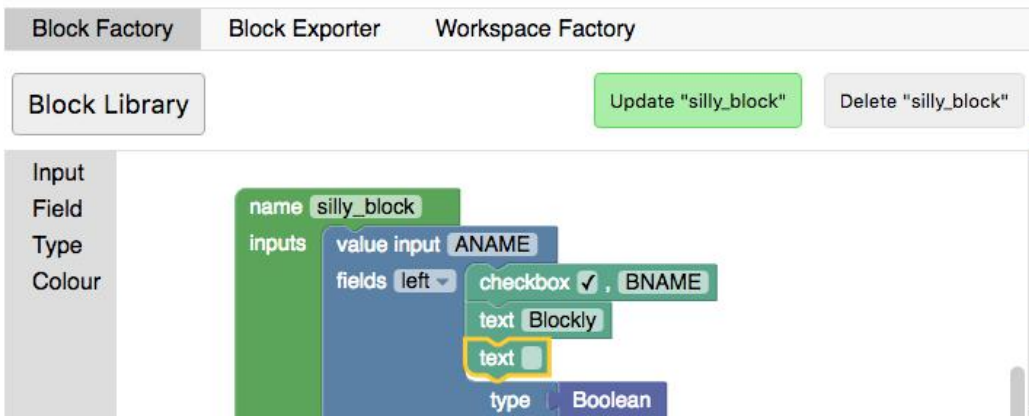


任务一：自己动手定义一个新块，并描述它的功能。

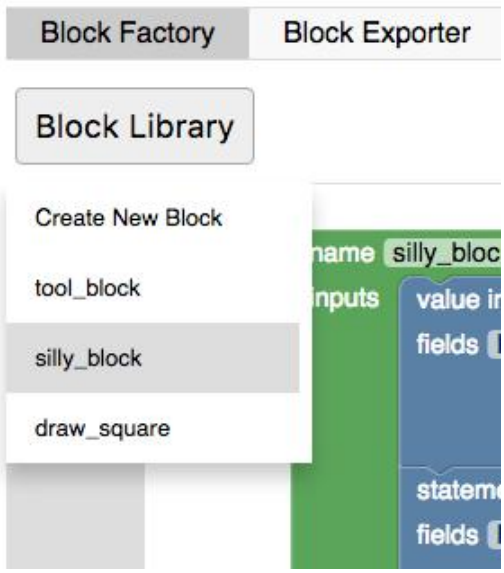
6.2.2 管理库

块由其名称引用，因此要创建的每个块都必须具有唯一的名称。 用户界面强制执行此操作，并在您“保存”新块或“更新”现有块时清除。





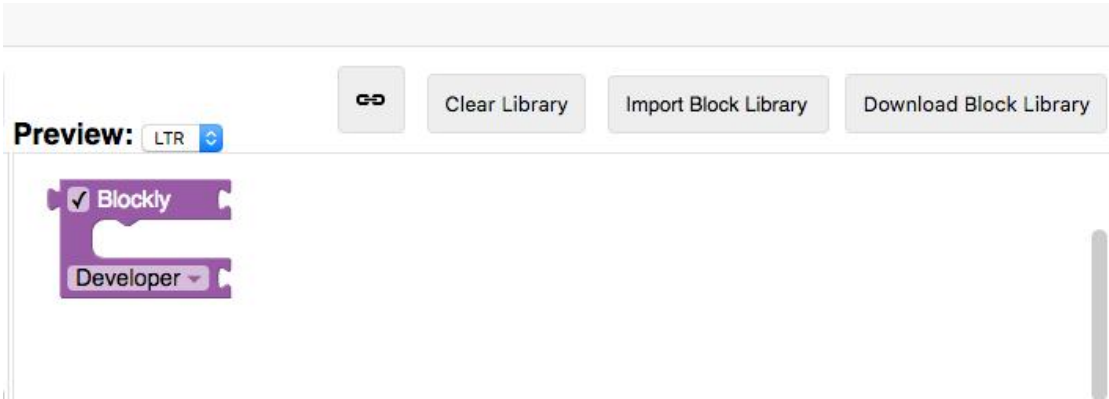
您可以在先前保存的块之间切换，或通过单击库按钮创建新的空块。更改现有块的名称是快速创建具有类似定义的多个块的另一种方法。



6.2.3 导入和导出库

块被保存到浏览器的本地存储，清除浏览器的本地存储将删除您的块。 要无限期保存块，您必须下载库。 您的块库将下载为可导入的 XML 文件，以将您的块库设置为下载文件时的状态。请注意，导入块库将替换当前的库，因此您可能需要先备份导出。

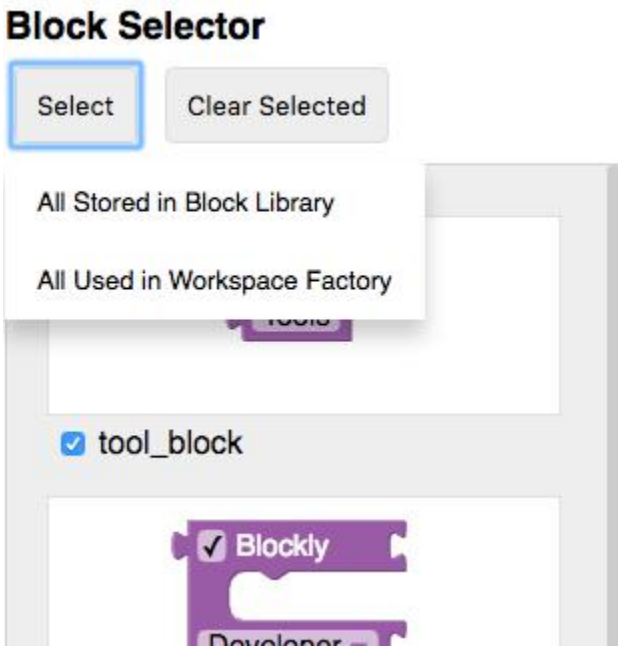
导入和导出功能也是维护和共享不同组自定义块的推荐方式。



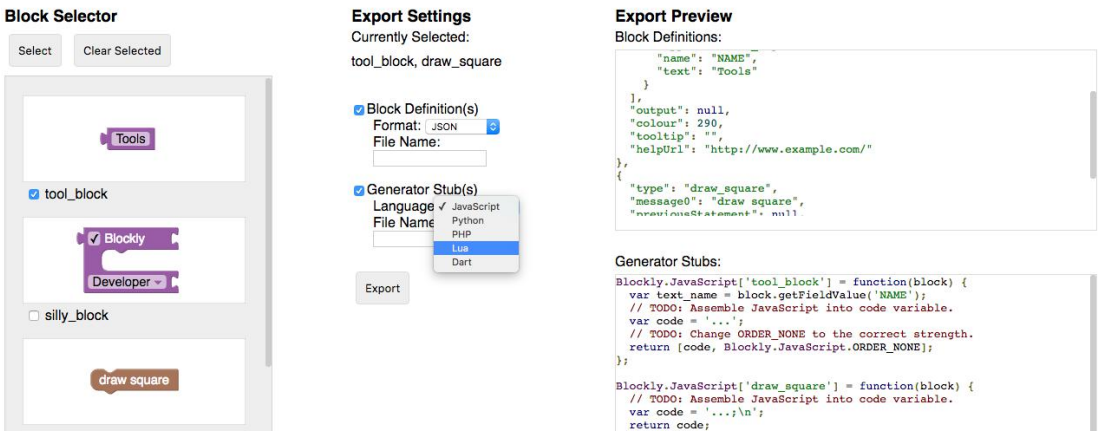
6.2.4 块导出器

如果你设计了块，并且想要在应用程序中使用它们的时候，可以在块导出器重完成块定义和生成器的导出。

存储在块库中的每个块都将显示在块选择器中。 单击块以选择或取消选择要导出的块。 如果要选择库中的所有块，请使用“选择”→“所有存储在块库”选项。 如果使用“工作区出厂”选项卡构建了工具箱或配置了工作区，则还可以通过单击“选择”→“在工作区工厂中使用”选择所有使用的块。



导出设置允许您选择要定位的生成语言，以及是否需要所选块的定义。 选择这些文件后，点击“导出”即可下载文件。



6.2.5 工作区工厂

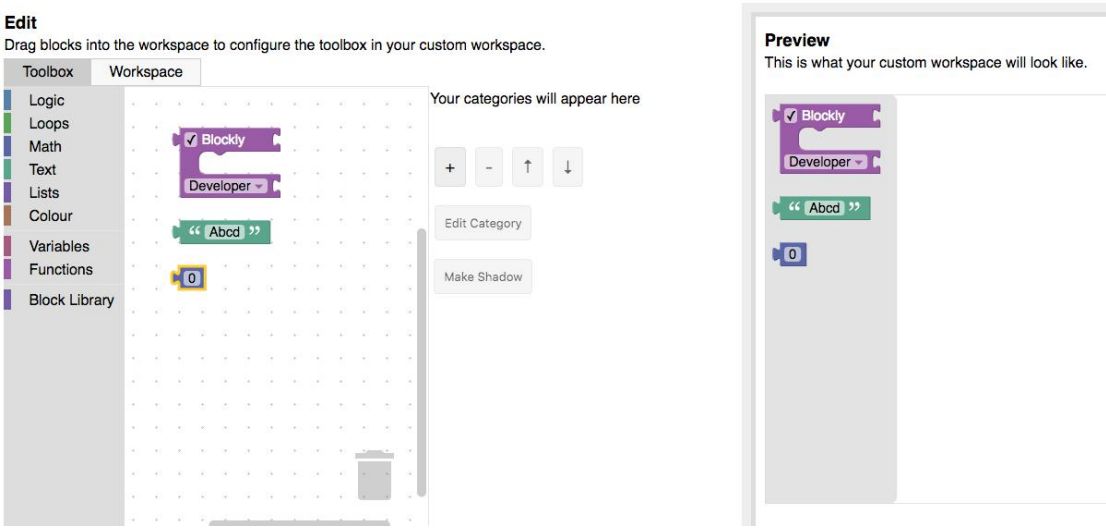
工作区工厂可以方便地配置工具箱和工作区中的默认块组。 您可以使用“工具箱”和“工作区”按钮在编辑工具箱和起始工作区之间切换。

(1) 构建工具箱

此选项卡有助于构建工具箱的 XML，该材料假定使用者熟悉工具箱的功能，如果您在此处要编辑工具箱的 XML 时，可以通过单击“加载到编辑”加载它。

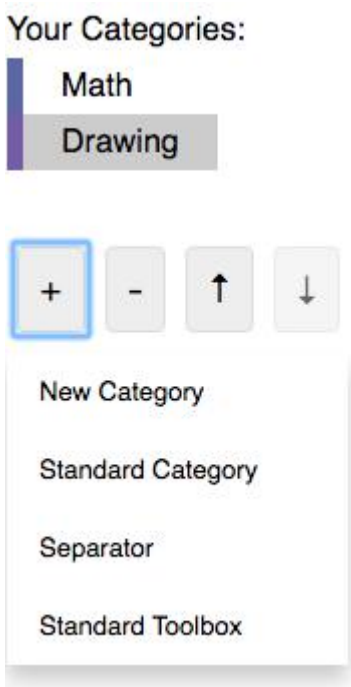
(2) 没有类别的工具箱

如果您有几个块，它们没有任何类别，想要显示它们的时候，只需将它们拖动到工作区中，您将看到您的块出现在工具箱的预览中。



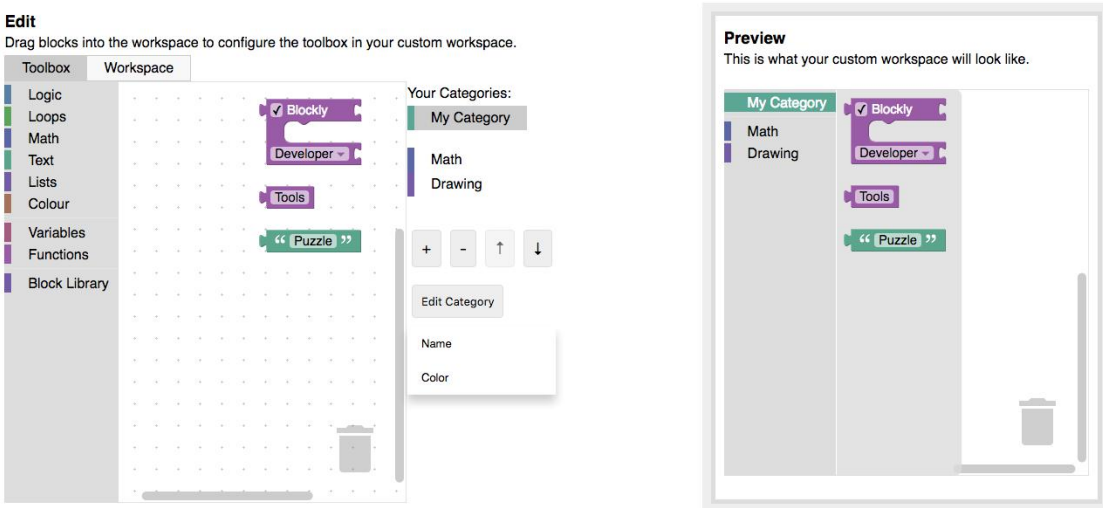
(3) 有类别工具箱

如果你想要显示块类别，点击“+”按钮，并选择下拉项目为新类别。 这将向您的类别列表中添加一个类别，您可以选择和编辑。 选择“标准类别”以添加单个标准块类别（逻辑，循环等）或“标准工具箱”以添加所有标准块类别。 使用箭头按钮重新排序类别。



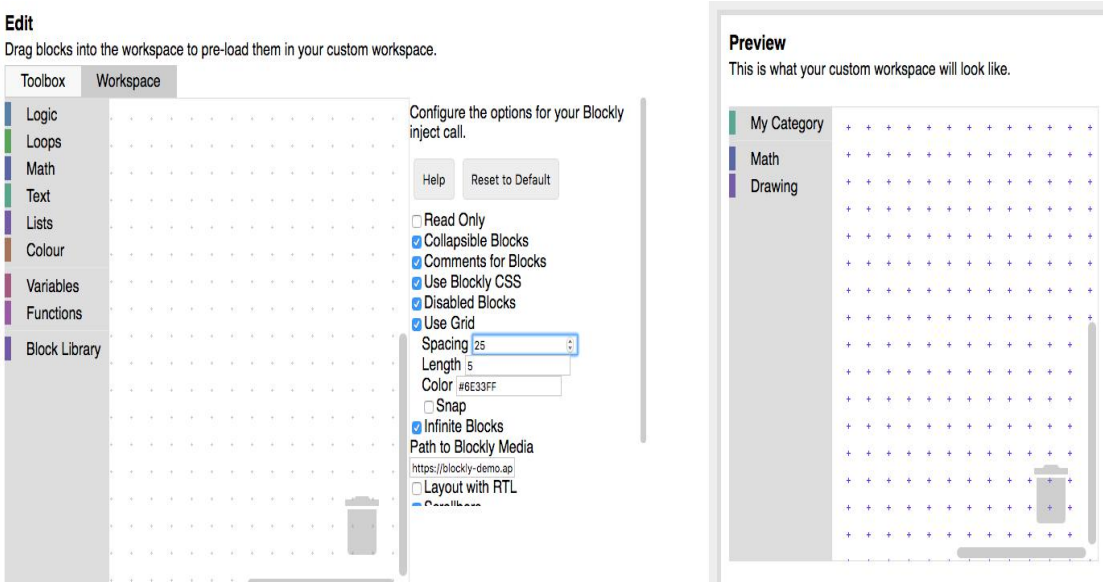
要更改所选类别名称或颜色，请使用“编辑类别”下拉菜单。 将块拖动到

工作区中将将其添加到所选类别。



(4) 选择工作区选项

为配置选项设置不同的值，并在预览区域中查看结果。 启用网格或缩放会显示更多配置选项。 此外，切换到使用类别通常需要更复杂的工作空间；当您添加第一个类别时，会自动添加垃圾桶和滚动条。



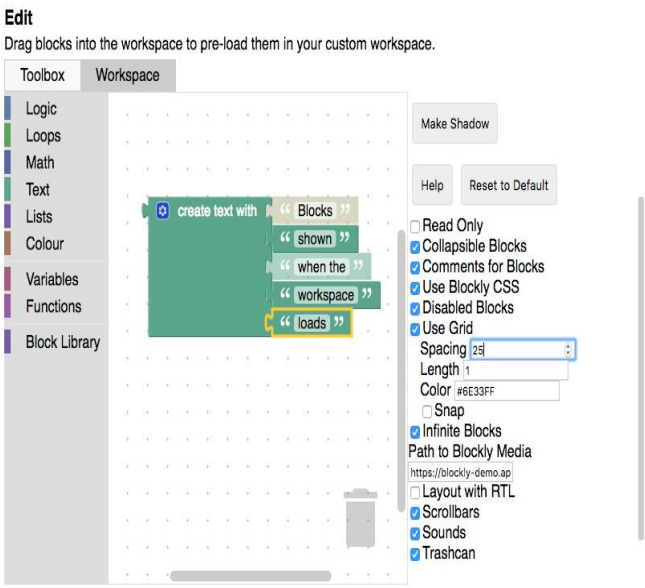
(5) 将预加载块添加到工作区

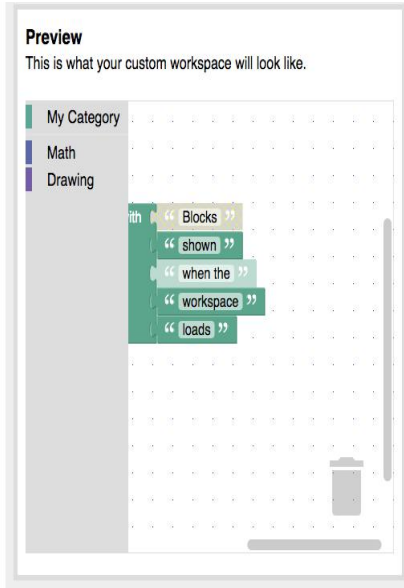
这是可选的，但如果要在工作空间中显示一组块，则可能需要：

a 当应用程序加载时显示

b 当触发事件（提高级别，单击帮助按钮等）时显示

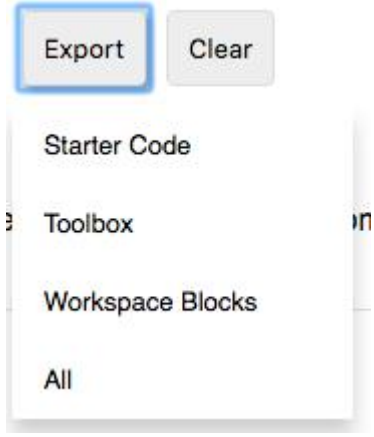
将块拖动到编辑空间中，可以在预览区中查看它们。 您可以创建块组，禁用块，并在选择某些块时创建阴影块。





(6) 导出

工作区工厂提供以下导出选项：



◎Starter Code：生成 html 和 javascript 以注入您的自定义 Blockly 工作区。

◎工具箱生成 XML 以指定您的工具箱。

◎工作区块生成可以加载到工作区中的 XML。

(7) 更多创建自定义块的信息，可参考 Google Blockly: https://developers.google.com/blockly/guides/create-custom-blocks/overview

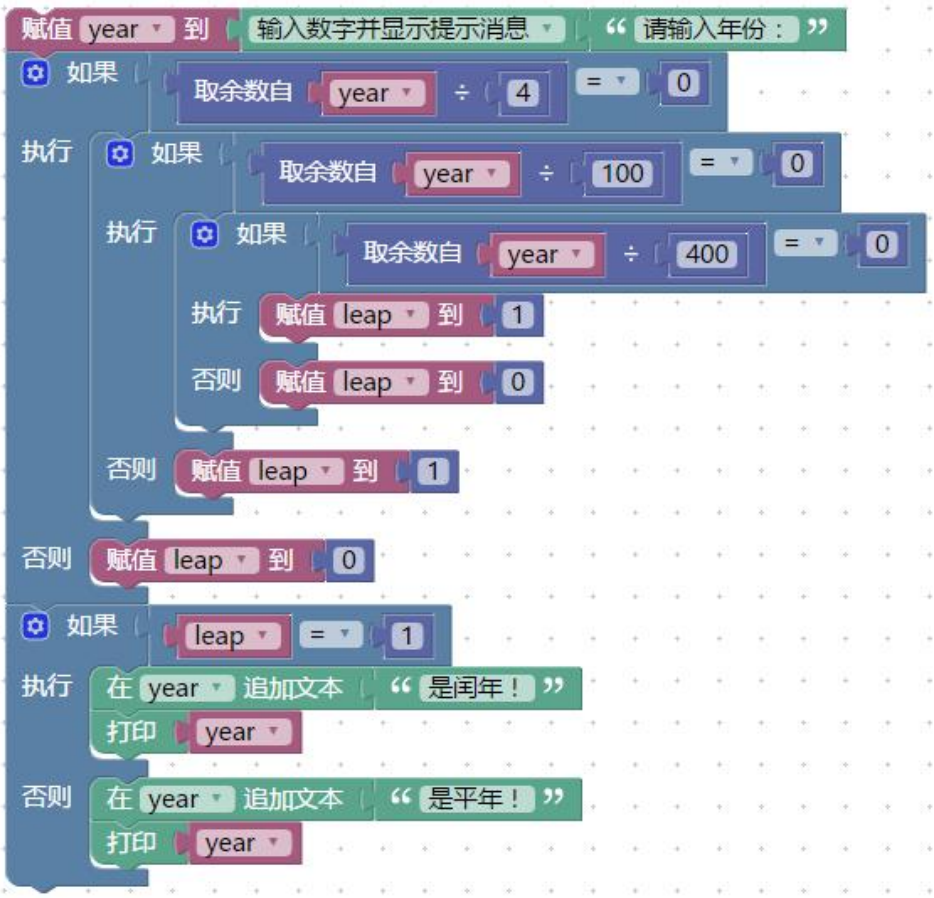
6.3 Blockly 的高级使用

在之前的学习中，我们通过使用 Blockly，学习一些基础程序设计中的经典的例子，并通过 Blockly 的可视化代码编辑器，进行了编程的实践练习,我们所接触和使用的这些，并不是 Blockly 设计的初衷。Blockly 是一个库，它为 Web 和 Android 应用程序添加了一个可视化代码编辑器，Blockly 编辑器使用互锁的图形块来表示代码概念，如变量，逻辑表达式，循环等，它允许用户应用编程原则，而不必担心语法或命令行上闪烁的光标。

6.3.1 将 Blockly 作为代码生成器

每个人不可能精通甚至熟悉每一种语言，但有时候，在学习、工作中又可能会用到所未接触过的语言，如果我们没有额外的时间且精力，尤其当这种语言再极少使用时，我们可能不乐意去花时间和精力去学习，但又不得不用，于是经常陷入两难。针对这一常见现象，我们就可以使用 Blockly 作为代码生成工具。

(1)假如现在我们需要一个判断平年闰年的 Python 代码的小例子，但我们之前又没接触过 Python，我们又不想学习 Python，那么就可以打开 Blockly，在编辑区拖动块来编写:



拖动完成，验证无误，点击 Python 选项卡，复制代码至你的 Python 环境中，即可直接运行。

```
leap = None
year = None
_E9_A1_B9_E7_9B_AE = None

def text_prompt(msg):
    try:
        return raw_input(msg)
    except NameError:
        return input(msg)

year = int(text_prompt('请输入年份: '))
if year % 4 == 0:
    if year % 100 == 0:
        if year % 400 == 0:
            leap = 1
        else:
            leap = 0
    else:
        leap = 1
else:
    leap = 0

if leap == 1:
    year = str(year) + str('是闰年! ')
    print(year)
else:
    year = str(year) + str('是平年! ')
    print(year)
```

Python 环境中的运行结果：

```
"C:\Program Files\Python\Python27\python.exe" "H:/Python
请输入年份：1900
1900是平年!

Process finished with exit code 0
```

(2)假如现在需要一个 JavaScript 的执行脚本，而且我们对 JavaScript 也有所了解，我们也可以尝试在 Blockly 中进行编程开发，比如这个猜数字的小游戏：

上面是程序的主体部分，包括循环、提示和中断。

在程序之前，创建了三个变量并进行了初始化:

步长:用于计数，针对猜的次数进行不同的提示。

Target：存放随机生成的目标数，与所猜数 Number 进行比较。

Flag：开关变量，用于标记是否猜对，从而决定是否提示下方内容。

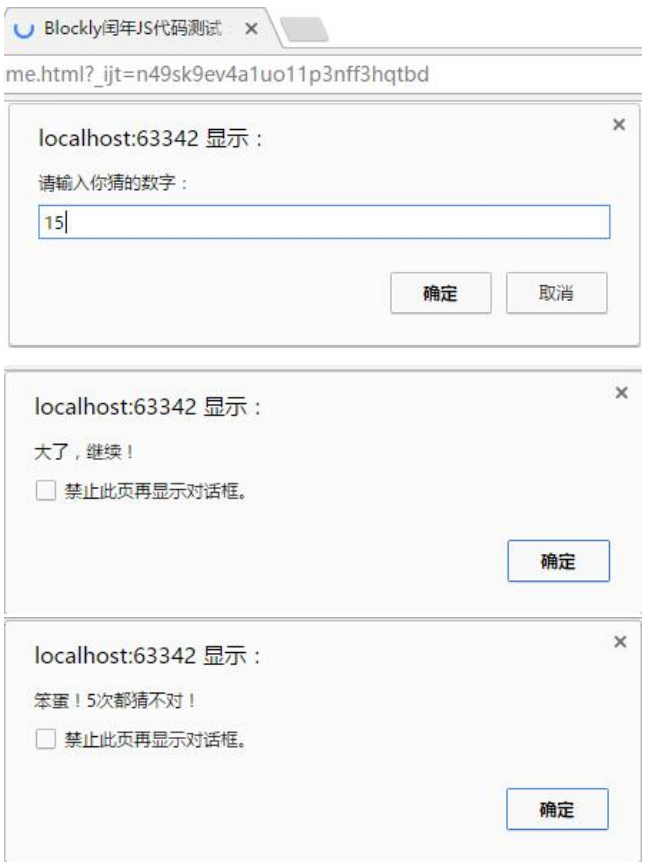
同样，验证无误后，点击 JavaScript 选项卡，复制 js 代码并保存。

```
var count, flag, number, target;

function mathRandomInt(a, b) {
  if (a > b) {
    // Swap a and b to ensure a is smaller.
    var c = a;
    a = b;
    b = c;
  }
  return Math.floor(Math.random() * (b - a + 1) + a);
}


count = 0;
flag = true;
target = mathRandomInt(1, 30);
```

```
for (var count2 = 0; count2 < 5; count2++) {
  number = window.prompt('请输入你猜的数字: ');
  count = count + 1;
  if (number == target) {
    if (count == 1) {
      window.alert('真神了！猜对啦！');
      flag = false;
      break;
    }
    if (count == 2) {
      window.alert('厉害，对啦！');
      flag = false;
      break;
    }
    if (count == 3) {
      window.alert('不错，对啦！');
      flag = false;
      break;
    }
    if (count == 4) {
      window.alert('有点慢，对啦！');
      flag = false;
      break;
    }
    if (count == 4) {
      window.alert('下次快点，对啦！');
      flag = false;
      break;
    }
  } else {
```

```
    if (number > target) {
      window.alert('大了，继续！');
    } else {
      window.alert('小了，继续！');
    }
  }
}
if (flag) {
  window.alert('笨蛋！5次都猜不对！');
}
```

将保存后的 js 代码导入到 html 文件中测试。



当然，也可以把猜数字小游戏的 Python 代码导出，同样可执行。

```
"C:\Program Files\Python\Python27\python.exe"
请输入你猜的数字：15
大了，继续！
请输入你猜的数字：7
大了，继续！
请输入你猜的数字：3
大了，继续！
请输入你猜的数字：2
大了，继续！
请输入你猜的数字：1
下次快点，对啦！

Process finished with exit code 0
```

注意：

◎ Python 代码的导出执行，当程序涉及输入且输入的是数字时，需要使用 int()，将输入的字符串型“数字”强制类型转换成整型。


```
# Blockly生成
year = text_prompt('请输入年份：')
# 修改加入强制类型转换后
year = int(text_prompt('请输入年份：'))
```

如果不进行强制类型转换，执行脚本时可能会报错，即使不报错，结果也可能不正确。但这一问题在 JavaScript 导出代码中不存在，兼容性良好。

```
TypeError: not all arguments converted during string formatting
```

◎ 对于 JavaScript 代码的测试，可以导入到 html 中，在浏览器中执行，测试效果与 Blockly 中效果相同。html 代码如下：

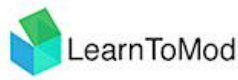
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Blockly闰年JS代码测试: </title>
</head>
<body>
    <script src = "GuessNumberGame.js">
    </script>
</body>
</html>
```

本条注意事项专门针对没有 JavaScript 基础，但是对 Blockly 代码生成工具又感兴趣，想亲手测试验证的读者。

关于 Blockly 作为代码生成工具的使用，我们这里只举了两个基础的，有代表性的例子，当然如果你学有余力或者对所生成的目标代码十分熟悉，可以自行尝试更加有趣、更加复杂的例子，如果你觉得这并不能满足你的需求，那么可以尝试自己动手定义你想要的块，生成代码的格式、类型和种类。

6.3.2 Blockly 的二次开发

随着 Blockly 逐渐的完善，它被越来越多的人所熟知，同时，凭借它可视化编程，良好的可扩展性等特点，很多的开发者利用 Blockly 进行二次开发,因此衍生出许多优秀的产品和工具。



前面曾提到过，Blockly 的初衷是针对开发人员设计的，它是一个针对有经验的开发人员的复杂库。从用户的角度来看，Blockly 是一种直观，可视化的构建代码的方法。从开发人员的角度来看，Blockly 本质上是一个包含正确语法、生成代码的文本框， Blockly 可以导出多种语言，例如 JavaScript，Python，PHP，Lua，Dart 等，下面是对 Blockly 进行二次开发的步骤：

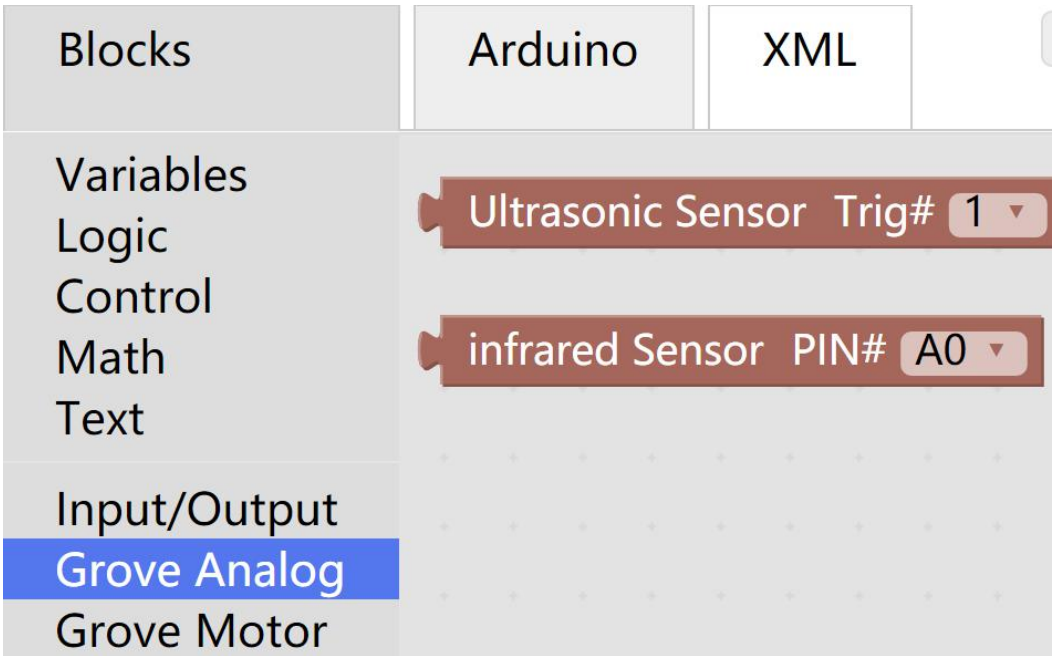
a.集成块编辑器。 Blockly 编辑器包括用于存储块类型的工具箱和用于排列块的工作空间。

b.创建应用程序的块。 一旦你的应用程序中有 Blockly，你就需要创建块供用户编码，然后将它们添加到您的 Blockly 工具箱。

c.构建应用程序的其余部分。 本身，Blockly 只是一种生成代码的方法，你的应用程序的核心在于如何处理该代码。

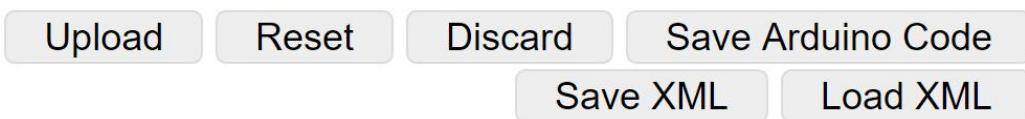
可能单纯的文字描述比较抽象，难以理解，以 FreDuino 为例，它是基于 Blockly 二次开发而成的一个远程硬件控制平台。

a. 集成块编辑器，在工具箱里增添了与硬件外设进行交互的编码块。



b. 创建应用程序块，实现了 Blockly 块与硬件控制代码之间的转换。

c. 在构建应用程序部分，通过与硬件外设建立通信，实现代码的上传，进而完成与硬件的交互。



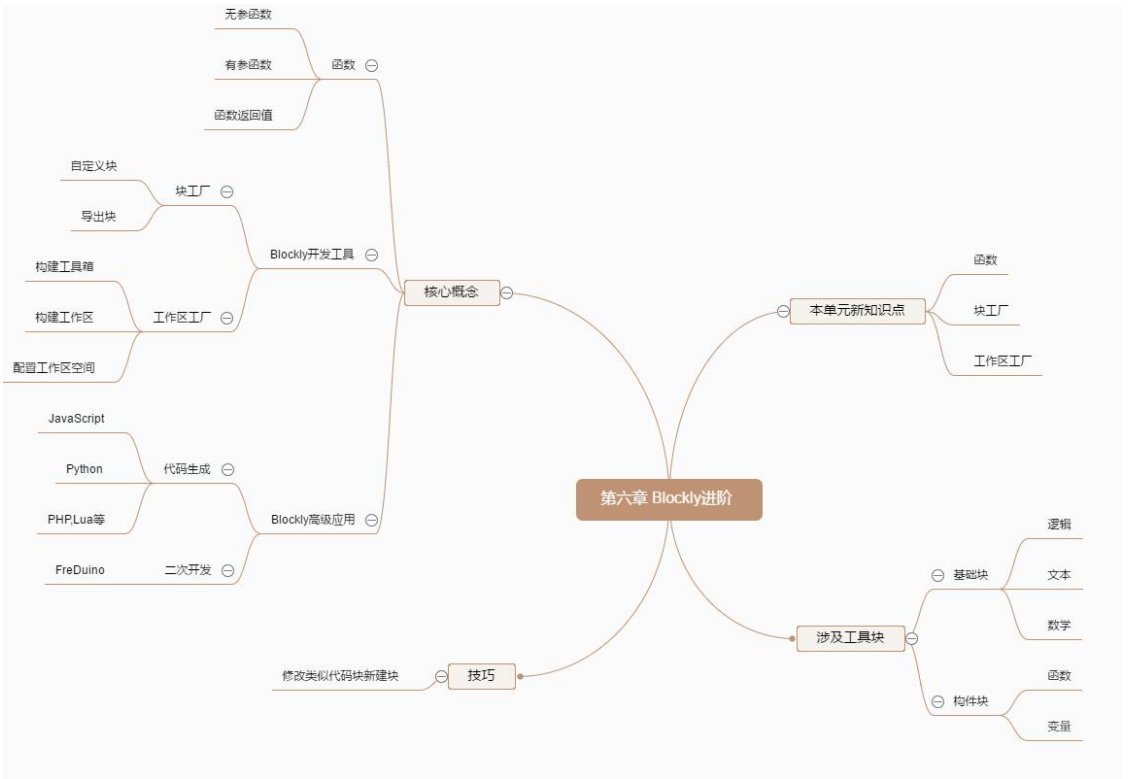
FreDuino 就是通过上述三个步骤而诞生的，实现了对硬件外设的远程可视化控制。

课后习题

1. 写一个判断素数的函数，在主函数输入一个整数，输出是否是素数。

2. 设计定义一个自己的工具块。

3. 编写函数，给出年、月、日，计算该日是该年的第 n 天，并尝试将其导出的代码在其他语言环境中调试运行。


第六章 Blockly进阶
核心概念
函数
无参函数
有参函数
函数返回值
Blockly开发工具
块工厂
自定义块
导出块
工作区工厂
构建工具箱
构建工作区
配置工作区空间
Blockly高级应用
代码生成
JavaScript
Python
PHP,Lua等
二次开发
FreDuino
技巧
修改类似代码块新建块
本单元新知识点
函数
块工厂
工作区工厂
涉及工具块
基础块
逻辑
文本
数学
构件块
函数
变量