MOONHACK 2020

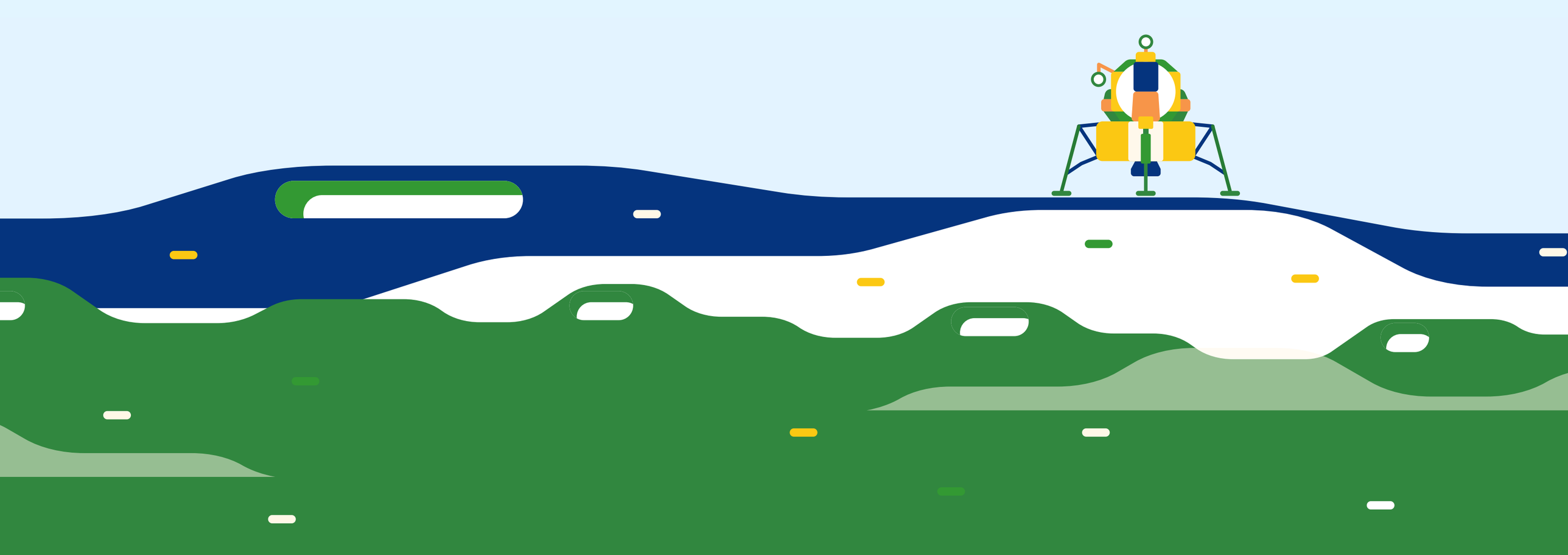
PYTHON WATER ON MARS

MANDARIN

BROUGHT TO YOU BY CODE CLUB AUSTRALIA
POWERED BY TELSTRA FOUNDATION

code club / AUSTRALIA | POWERED BY TELSTRA FOUNDATION

SUBMIT AND BE COUNTED AT
MOONHACK.COM



火星上的水 - Moonhack Python项目

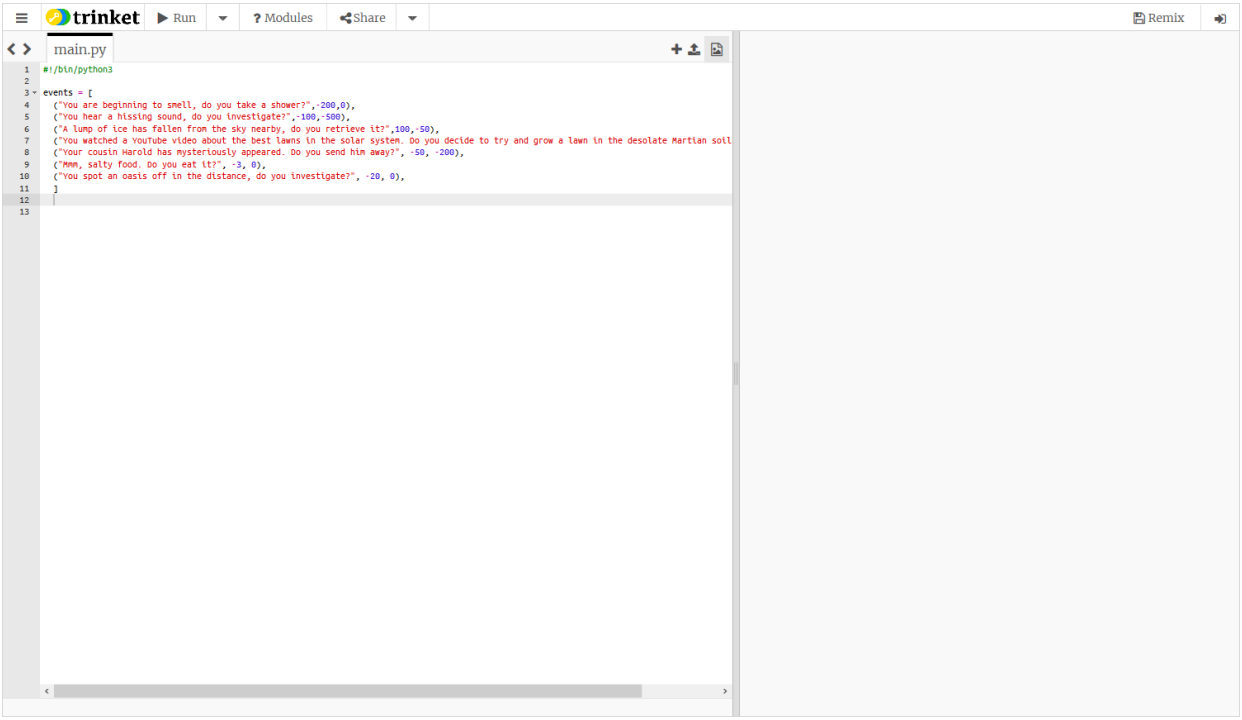
简介

水是生命赖以为生的宝贵资源。 节约用水很重要，并且在我们的物种进入太空之后会变得更加重要。 今天我们将会制作一个基于文本的生存游戏，在这个游戏中，如果想要在火星上生存尽可能长时间并且避免把水用光，那么玩家必须作出小心用水的决定。

步骤1：记录水量

现在我们即将开始游戏，首先我们需要设置一个变量用来记录我们有多少水。

- 在此链接打开项目模版bit.ly/pythonwater。 你会看到以下界面：



On the left is your code window; you will notice there is already some code here (we'll come back to that later). 右边的界面用来显示游戏的输出结果，叫做执行窗口。 运行代码的按钮在上面的菜单栏，你可以按此按钮来运行代码。

- We need to give our player some water to start off with. 我们的玩家需要一些水来开始游戏。让我们从整数：1000升开始吧！ 在你程序的最后，加入以下的高亮代码。

```
10    ("You spot an oasis
11    ]
12
13  water = 1000
```

- 只要有水，我们的玩家就可以一直玩游戏。 当水没有了，游戏就结束了。 让我们创建游戏的主循环 - 只要有水，游戏就可以继续：

```
13  water = 1000
14
15  while water > 0:
```

注意每行代码都是用冒号(:) 结尾。 这个冒号看起来很微不足道，但是如果我们忘记加，Python会报错。

- 接下来，在游戏的主循环里面，我们需要让玩家用水。 不管玩家怎么选择，他们每天都要减去一定数量的水，因为他们需要喝水。 把下面的代码加入你的循环里面：

```
15 while water > 0:
16   water -= 5
```

注意在第16行的前面有两个空格，我们把它叫做**缩进**。 这告诉Python那一行指令是在循环里面。 换句话说，它会在每次执行游戏主循环的时候发生（如果你用过Scratch，它就类似于橙色循环方块）。

- 现在是时候写一些输出了。 让我们在玩家用完水的时候告诉玩家游戏结束。 在你的游戏主循环之外添加以下代码（请记得删除缩进！）：

```
15 while water > 0:
16   water -= 5
17
18 print ("Game Over :(")
```

- 运行你的代码。 会发生什么呢?
- Python会尽快运行我们的代码。 因为在主循环中没有任何输出，在结束之前我们都无法知道发生了什么。 让我们在主循环中添加一些输出。

```
15 while water > 0:
16   water -= 5
17   print("A day has passed.")
18
19 print ("Game Over :(")
```

- 再次运行你的代码。 现在你应该可以看到游戏结束前过去日子的输出。
- 让我们改一下刚刚添加的那一行代码，来告诉我们添加了多少水。 我们可以使用+号将变量的值与要打印的文本结合起来。

```
15 while water > 0:
16   water -= 5
17   print("A day has passed. Your Water " + water + "L")
18
19 print ("Game Over :(")
```

- 再次运行你的代码。 有没有报错?
- 编程的很大一部分是试图弄清楚为什么我们会遇到错误。 这一次的错误告诉我们“cannot concatenate 'str' and 'int' objects”。 在编程时，你经常需要弄清楚这些奇怪的错误消息的含义。

这个错误告诉我们，我们正在尝试结合两种不同的东西，一个是文本（字符串，缩写为str），另一个是用于记录水的数字（整数，缩写为int），然而Python不知道如何结合。 为了解决这个问题，我们可以告诉Python使用以下代码将整数转换为字符串：

```
15 while water > 0:
16   water -= 5
17   print("A day has passed. Your Water " + str(water) + "L")
18
19 print ("Game Over :(")
```

- 运行你的代码。 现在你应该可以看到一个输出，告诉我们在玩家的角色用尽水之前，每天我们要用多少水。

挑战：改变日常用水习惯

目前，我们每天使用5升水。 您认为一个人的最低生存限度是什么呢？ 您可以更改代码使得玩家使用定量的水吗？

步骤2：追踪天数

到目前为止，我们一直在跟踪玩家使用的水量，但是为了查看他们的使用情况，我们将记录已经过去的天数。

- 首先，我们将天数变量的初始值设置为1。

```
13 water = 1000
14 day = 1
15
16 while water > 0:
```

- 接下来，每经过一天，我们希望递增天数的变量。 将以下代码添加到你的游戏主循环中：

```
16 while water > 0:
17   water -= 5
18   print("A day has passed. Your Water " + str(water) + "L")
19   day += 1
20
21 print ("Game Over :(")
```

- 最后，我们将让玩家知道他们的战果。 为此，我们将最终的打印语句替换为包含已经度过的天数的语句。

```
19      day += 1
20
21  print("you lasted " + str(day) + " days")
```

请注意，我们必须再次使用str(day)将数字转换为文本。

- 运行你的代码。 你应该每天获得一个输出，然后有一个最终输出，该输出表明玩家持续了多少天。

挑战：输出每天的天数

目前，我们只是在最后使用day变量。 你可以改变代码来输出每天的天数吗？

当前的输出是：

```
A day has passed. Your Water 950L
```

你可不可以改成这样：

```
Day: 10. Your Water 950L
```

步骤3：你的日常事件

现在，我们将添加一些游戏性。 每天，我们都会给玩家一个选择，这个选择将决定他们损失或获得的水量。

- 你可能已经注意到，开始的两行不是我们写的代码。 这儿有一个火星上可能发生的事件的列表。 接下来在主循环中添加另一个打印语句。 这将访问我们的事件列表并随机选择一个。

```
18    print("A day has passed. Your Water " + str(water) + "L")
19    print(random.choice(events))
20    day += 1
```

哦，不！ 另一个错误： “name 'random' is not defined”。 这是因为random不是python默认的。它是**模块**的一部分。 我们需要导入它才能使用它。

```
1  #!/bin/python3
2  import random
3
4  events = [
```

- 运行你的代码。 每天，你都应该打印出一个随机事件。
- 我们需要在指定日期多次使用该事件。 如果我们每次都使用random.choice (events)获取事件，我们都会得到一个不同的事件！ 相反，让我们将该事件存储在一个名为e

vent的变量中，我们可以多次使用它而不必担心。

```
19    print("A day has passed. Your Water " + str(water) + "L")
20    event = random.choice(events)
21    print(event)
22    day += 1
```

- 再次运行你的代码。 你应该得到与上次相似的结果。
- 您可能已经注意到伴随着输出的事件，输出中有大量额外的信息和符号。 这是因为当玩家回答是或否后，我们将存储有关获得或失去的水量的问题和值。 我们其实只想问玩家一个问题，而不想向他们显示额外的信息。 要访问该问题，我们可以使用所谓的**索引**符号。 索引只是元素所处的位置。位置从0开始。 将你的打印行更改为以下内容：

```
20    event = random.choice(events)
21    print(event[0])
```

- 再次运行你的代码。 已经好了很多，但是玩家还无法回答这个问题。
- 让我们使用input输入语句，而不是使用print打印语句。 这告诉python期望用户输入一些东西。

```
20    event = random.choice(events)
21    input(event[0])
22    day += 1
```

现在，让我们将输入存储在变量中，因为我们稍后将要使用它。

```
20    event = random.choice(events)
21    response = input(event[0])
22    day += 1
```

- 再次运行您的代码。 现在，您的游戏应等待玩家每天响应。

步骤4：使用玩家的答案

玩家正在回答这个问题，现在我们需要使用他们的回答来判断他们的用水量。

- 玩家应该回答是或者否。 如果他们回答“yes”，我们应该在事件中添加第一个数字：

```
21    response = input(event[0])
22    if response == "yes":
23        water += event[1]
24    day += 1
```

请注意，即使我们在做添加数字，事件中的许多数字都是负数。 就像算术一样，加一个负数和减去该数的正数是相同效果的。

- 如果玩家回答“no”，我们应该在事件中添加第二个数字。

```
22  if response == "yes":
23      water += event[1]
24  elif response == "no":
25      water += event[2]
26  day += 1
```

- 运行你的代码。 现在玩家们应该可以回答问题了。 你会注意到，玩家拥有的水量将会根据他们的决定而改变。 如果玩家回答“yes”或“no”之外的其他内容，会发生什么？

步骤5：验证用户输入

您现在有一个正常运行的游戏，但是玩家可以按照他们想要的任何方式回答。 我们只想接受“yes”和“no”作为答案。

- 只有两个选项，yes和no，因此我们可以提示用户按照我们希望他们回答的方式回答。将您的响应行修改为以下内容：

```
20  event = random.choice(events)
21  response = input(event[0] + " (yes/no): ")
22  if response == "yes":
```

- 好了很多，但是玩家仍然可以按照他们想要的任何方式回答。 玩家可以输入其他内容来破解游戏，或者输入错误的答案。 让我们添加一个"else"语句来捕获“yes”或“no”以外的任何响应。

```
24  elif response == "no":
25      water += event[2]
26  else:
27      print("Please answer yes or no!")
28  day += 1
```

- 运行你的代码。 如果玩家回答“yes”或“no”之外的其他内容，会发生什么？
- 现在，我们要求用户更正他们的回答，但是无论如何我们需要继续问他们下一个问题！我们需要不断问他们同样的问题，直到他们给我们想要的答案为止。 我们可以使用循环来做到这一点。

```
20    event = random.choice(events)
21    while True:
22        response = input(event[0] + " (yes/no): ")
23        if response == "yes":
24            water += event[1]
25        elif response == "no":
26            water += event[2]
27        else:
28            print("Please answer yes or no!")
29    day += 1
```

请注意，我们已经缩进了响应行和if/else语句。 这意味着一切都在循环中。 您可以通过高亮要缩进的行，并按键盘上的"tab"键来执行此操作。

- 运行你的代码。 如果您回答"yes"或"no"*以外*的值会怎样？ 它应该再次问您一个问题。 但是，如果您*确实*回答了是或否后怎样呢？ 它应该会再次问您一个问题。 天啊！

- “while True”循环是一个始终保持循环的循环。 如果您想永远执行下去，这个功能会非常有用。但是我们不想永远执行下去，我们只想循环询问，直到用户回答yes或no！ 幸运的是，我们有一条特殊的命令可以使我们跳出循环，即使这个循环会永久执行。 这个命令是“ break”。 让我们在玩家回答是或否时添加一个break命令。

```
22        response = input(event[0] + " (yes/no): ")
23        if response == "yes":
24            water += event[1]
25            break
26        elif response == "no":
27            water += event[2]
28            break
29        else:
```

- 再次运行你的代码。 现在，如果您不回答yes或no，则应该只再次遇到相同的问题。现在事情可以正常工作了，但有一些玩家可能还是会遇到一些小问题。 如果玩家用大写字母Y回答“Yes”而不是回答“yes”，会发生什么情况？

- Python是区分大小写的 这意味着，如果您比较两个字符串，Python会因为它们的大小写不同而认为它们是不同的字符串。 幸运的是，这很容易解决。 Python有一个命

令，您可以应用到字符串上以使字符全部变为小写。 将您的响应行修改为以下内容：

```
21    while True:
22        response = input(event[0] + " (yes/no): ").lower()
23        if response == "yes":
```

- 运行你的代码。 它现在应该接受“Yes”，“YES”，“yeS”以及其他任何大小写字母组合。

恭喜你！

你已经完成了这个项目。 太棒了！ 试一下玩玩你写的游戏，看看能不能比你的朋友生存更长的时间。 想要更多功能？ 请尝试以下挑战，或者看看我们的Scratch项目或Moonhack前几年的项目。

挑战：添加更多事件

游戏已经有了几个不同的事件，但是你可以添加更多吗？

提示：如果遇到困难，请尝试复制和编辑现有的事件之一。

挑战：地球游戏

你可以将游戏修改为关于地球而不是火星吗？ 考虑可能发生的事件的种类，以及玩家可能必须面对的选择。

高级挑战：Sam，再玩一次

你能给玩家一个可以在输掉比赛后继续玩的选择吗？

提示：你也许想要把整个游戏放在一个循环里面。