MOONHACK 2020

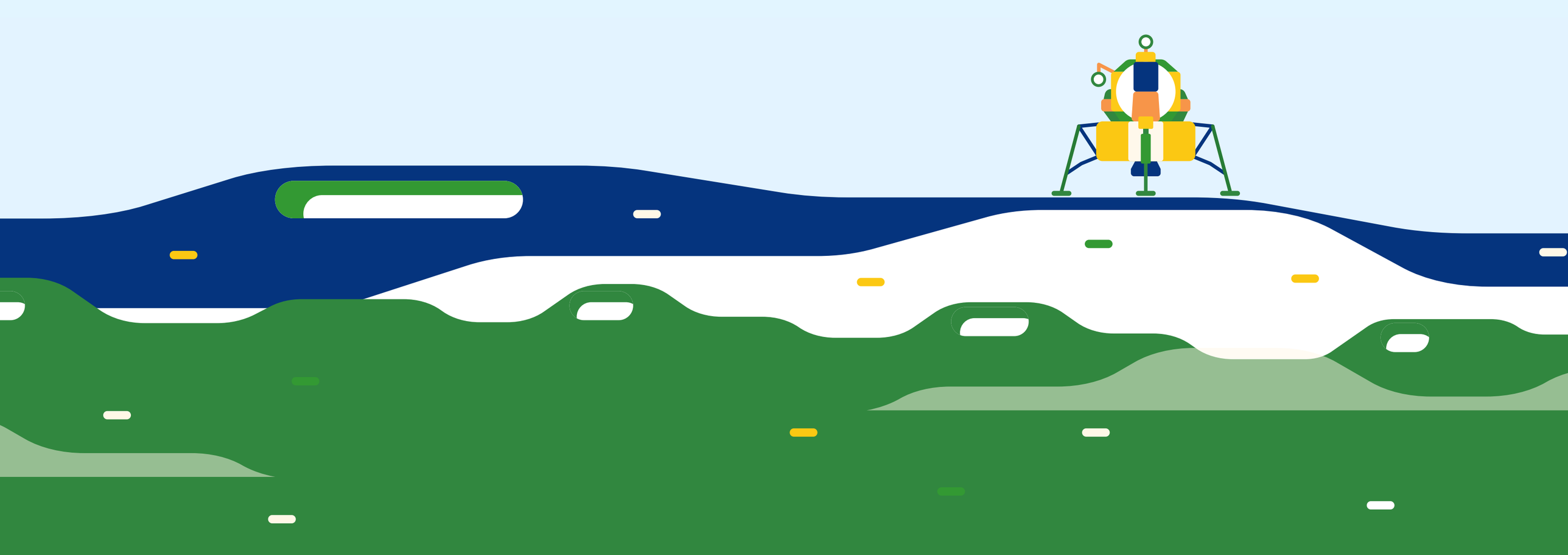
PYTHON WATER ON MARS

PORTUGUESE

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



Água em Marte

Projeto Moonhack Python

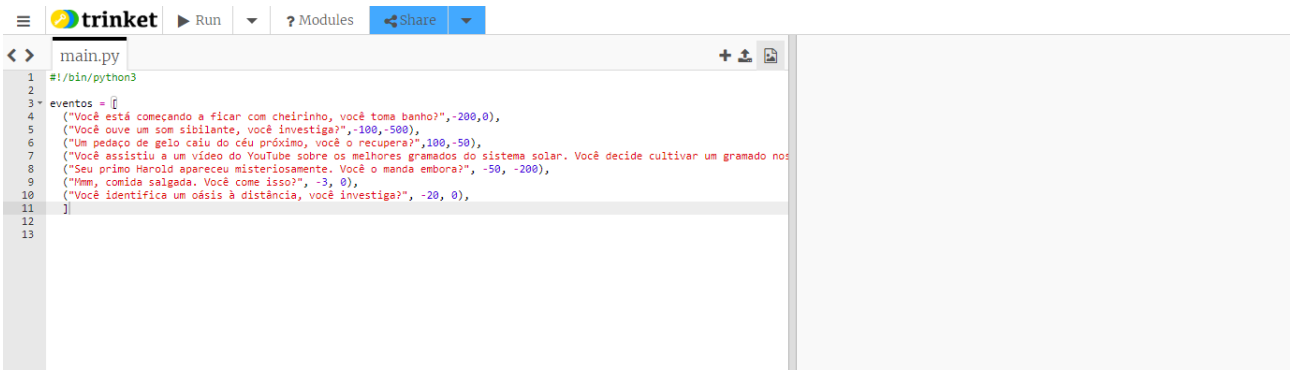
Introdução

A água é um recurso incrivelmente precioso que é crucial para a vida. A conservação da água é importante e só se tornará mais importante à medida que nossa espécie se aventurar no espaço. Hoje estaremos fazendo um jogo de sobrevivência baseado em texto, no qual o jogador deve tomar decisões conscientes da água para sobreviver o maior tempo possível sem ficar sem água em Marte.

Etapa 1: Manter o controle da água

Nesta primeira etapa, iniciaremos nosso jogo e configuramos uma variável que podemos usar para acompanhar a quantidade de água que temos.

- Abra o projeto inicial em http://bit.ly/pythonagua. Sua janela deve ficar assim:



À esquerda é a sua janela de código; você notará que já existe algum código aqui (voltaremos a isso mais tarde). À direita está sua janela de execução; é aqui que a saída do seu jogo será exibida. No topo está o botão de execução; você usará esse botão para executar seu código.

- Precisamos dar ao nosso jogador um pouco de água para começar. Vamos começar com um bom número redondo de 1000L Adicione o seguinte código destacado na parte inferior do seu programa:

```
10      ("Você identifica um o
11      ]
12
13  agua = 1000
```

- Queremos que nosso usuário continue jogando o jogo enquanto tiver água. Quando eles ficarem sem água, seu personagem terminará o jogo. Vamos criar nosso loop principal do jogo, que continuará enquanto o usuário tiver água:

```
13  agua = 1000
14
15  while agua > 0:
16
```

Observe os dois pontos (:) no final da nossa linha. Esse pequeno personagem pode parecer insignificante, mas se o esquecermos, o Python irá reclamar!

- Em seguida, vamos adicionar um pouco de água ao nosso ciclo principal do jogo. Essa será uma quantidade que é subtraída todos os dias, independentemente das opções do jogador, porque ele precisa beber. Adicione o seguinte código dentro do seu loop:

```
15 while agua > 0:
16     agua -= 5
17
```

  Observe os dois espaços na frente da linha 16. Chamamos isso de recuo. Isso diz ao python que a instrução nessa linha está dentro do loop. Em outras palavras, isso acontece toda vez que percorremos o loop principal do jogo (se você usou o Scratch, é semelhante aos blocos de loop laranja).

- Agora é hora de alguma saída. Digamos ao usuário que o jogo acabou e que morreu quando ficou sem água. Adicione o seguinte código fora do loop principal do jogo (lembre-se de se livrar do recuo!):

```
15 while agua > 0:
16     agua -= 5
17
18 print("Fim de jogo :(")
19
```

- Execute seu código. O que acontece?
- O Python apenas tentará executar nosso código o mais rápido possível. Sem nenhuma saída no loop principal, não temos como saber o que está acontecendo até chegarmos ao fim. Vamos adicionar alguma saída ao nosso loop principal.

```
15 while agua > 0:
16     agua -= 5
17     print("Um dia se passou.")
18
19 print("Fim de jogo :(")
20
```

- Execute seu código novamente. Agora você deve ver todos os dias que passam antes do final do jogo. O personagem do jogador morre.
- Vamos mudar a última linha de código que inserimos para nos dizer quanta água adicionamos. Podemos combinar o valor de nossas variáveis com o texto que estamos imprimindo usando o sinal +.

```
15 while agua > 0:
16   agua -= 5
17   print("Um dia se passou. Sua água " + agua + "L")
18
19 print("Fim de jogo :(")
20
```

- Execute seu código novamente. Você recebeu um erro?
- Uma grande parte da programação está tentando descobrir por que estamos recebendo os erros que estamos recebendo. Nesse caso, o erro está nos dizendo que “não podemos concatenar objetos 'str' e 'int'”. Ao programar, você frequentemente precisará descobrir o que essas mensagens de erro estranhas significam.
  Esse erro está nos dizendo que estamos tentando combinar duas coisas diferentes que o Python não sabe combinar, o texto (string, encurtado para str) e o número que estamos usando para acompanhar nossa água (número inteiro, encurtado para int). Para corrigir isso, podemos dizer ao Python para converter nosso número inteiro em uma string com o seguinte código:

```
15 while agua > 0:
16   agua -= 5
17   print("Um dia se passou. Sua água " + str(agua) + "L")
18
19 print("Fim de jogo :(")
20
```

- Execute seu código. Você deve obter uma saída dizendo a quantidade de água que estamos usando todos os dias, antes que o personagem do jogador fique sem água.

Desafio: Alterar o uso diário da água

Atualmente, estamos usando 5 litros de água por dia. Qual você acha que é o mínimo absoluto de que uma pessoa precisa para sobreviver? Você pode alterar o código para que o jogador use essa quantidade de água?

Etapa 2: mantendo o controle dos dias

Até o momento, acompanhamos a quantidade de água que um jogador usou, mas, para ver como eles foram bem, acompanharemos o número de dias que passaram.

- Primeiro, definiremos o valor inicial da variável do dia, começando no dia 1.

```
13  agua = 1000
14  dia = 1
15
16  while agua > 0:
```

- Em seguida, todos os dias que passarmos, queremos aumentar o valor do dia atual em um. Adicione o seguinte código ao loop principal do jogo:

```
16  while agua > 0:
17    agua -= 5
18    print("Um dia se passou. Sua água " + str(agua) + "L")
19    dia += 1
20
```

- Por fim, informaremos o jogador sobre o desempenho deles. Faremos isso substituindo a declaração de impressão final por uma que inclua o número de dias que se passaram.

```
19    dia += 1
20
21  print("Você durou " + str(dia) + " dias")
22
```

Observe que tivemos que usar str (dia) novamente para converter o número em texto.

- Execute seu código. Você deve obter uma saída para todos os dias e depois uma saída final que diz quantos dias o jogador durou.

Desafio: envie o número do dia todos os dias

Atualmente, estamos apenas usando a variável day no final. Você pode alterar o código para gerar o número do dia todos os dias?

Atualmente, a saída dirá:

Um dia se passou. Sua água 950L

Você pode mudar para dizer:

Dia: 10. Sua água 950L

Etapa 3: seu evento diário

Agora vamos adicionar uma jogabilidade. Todos os dias, vamos dar uma opção ao jogador, e essa escolha determinará quanta água eles perdem ou ganham.

- Você provavelmente notou que há um monte de código no começo que você não escreveu. Esta é uma lista de eventos que podem ocorrer em Marte. Vamos adicionar outra declaração de impressão ao nosso loop principal. Isso acessará nossa lista de eventos e escolherá aleatoriamente.

```
16 while agua > 0:
17     agua -= 5
18     print("Um dia se passou. Sua água " + str(agua) + "L")
19     print(random.choice(eventos))
20     dia += 1
```

Ah não! Outro erro Este diz “nome 'aleatório' não está definido”. Isso acontece porque o aleatório não faz parte do python por padrão, faz parte de um módulo. Precisamos importá-lo para poder usá-lo.

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

- Execute seu código. Todos os dias, você deve imprimir um evento aleatório.
- Precisamos usar o evento várias vezes no dia especificado. Se usássemos random.choice (events) sempre que desejássemos usar o evento, teríamos um evento diferente! Em vez disso, vamos armazenar esse evento em uma variável chamada evento, que podemos usar várias vezes sem ter que se preocupar.

```
19     print("Um dia se passou. Sua água " + str(agua) + "L")
20     evento = random.choice(eventos)
21     print(evento)
22     dia += 1
```

- Execute seu código novamente. Você deve obter um resultado semelhante ao tempo anterior.
- Você provavelmente percebeu que o evento em nossa produção é cercado por várias informações e símbolos extras. Isso ocorre porque armazenamos a pergunta e os valores para a quantidade de água a ganhar ou perder se o jogador responder sim ou não no mesmo lugar. Nós realmente só queremos fazer a pergunta ao jogador sem mostrar a ele informações extras. Para acessar a pergunta, podemos usar o que é chamado de notação de índice. O índice é apenas a posição em que o elemento está, começando em 0. Altere sua linha de impressão para o seguinte:

```
20     evento = random.choice(eventos)
21     print(evento[0])
```

- Execute seu código novamente. Isso é muito melhor, mas o jogador não pode responder à pergunta.
- Em vez de usar uma declaração de impressão, vamos usar uma declaração de entrada. Isso diz ao python para esperar que o usuário escreva alguma coisa.

```
20     evento = random.choice(eventos)
21     input(evento[0])
```

Agora, vamos armazenar essa entrada em uma variável, porque queremos usá-la mais tarde.

```
20    evento = random.choice(eventos)
21    resposta = input(evento[0])
22    dia += 1
```

- Execute seu código novamente. Seu jogo agora deve esperar o jogador responder todos os dias.

Etapa 4: usando a resposta do jogador

O jogador está respondendo à pergunta, agora precisamos usar a resposta para julgar o uso da água.

- O usuário deve responder sim ou não à pergunta. Se eles responderem "sim", devemos adicionar o primeiro número no evento:

```
21    resposta = input(evento[0])
22    if resposta == "sim":
23      agua += evento[1]
24    dia += 1
```

  Observe que, mesmo adicionando, muitos números nos eventos são negativos. Assim como na matemática, adicionar um número negativo é o mesmo que subtrair o positivo desse número.

- Se o jogador responder "não", devemos adicionar o segundo número no evento.

```
22    if resposta == "sim":
23      agua += evento[1]
24    elif resposta == "não":
25      agua += evento[2]
26    dia += 1
```

- Execute seu código. O jogador agora deve poder responder às perguntas. Você notará que a quantidade de água que o jogador possui mudará de acordo com suas decisões. O que acontece se o jogador responder algo diferente de sim ou não?

Etapa 5: Validar a entrada do usuário

Agora você tem um jogo de trabalho, mas o jogador pode responder da maneira que quiser. Queremos apenas aceitar "sim" e "não" como respostas.

- Existem apenas duas opções, sim e não, para que possamos solicitar ao usuário que responda da maneira que queremos. Modifique sua linha de resposta para o seguinte:

```
21    resposta = input(evento[0] + " (sim/não): ")
22    if resposta == "sim":
23      agua += evento[1]
```

- É melhor, mas o jogador ainda pode responder da maneira que quiser. Um jogador pode inserir outra coisa para enganar o jogo ou digitar incorretamente sua resposta. Vamos

adicionar uma declaração "else" para obter qualquer resposta que não seja sim ou não.

```
24    elif resposta == "não":
25      agua += evento[2]
26    else:
27      print("Por favor, responda sim ou não")
28    dia += 1
```

- Execute seu código. O que acontece se o jogador digitar algo diferente de sim ou não?

  Run your code. What happens if the player enters something other than yes or no?

- Agora, estamos pedindo ao usuário que corrija sua resposta, mas, de qualquer maneira, estamos fazendo a próxima pergunta! Precisamos continuar fazendo a mesma pergunta até que eles nos dêem uma resposta que queremos ouvir. Podemos fazer isso com um loop.

```
20    evento = random.choice(eventos)
21    while True:
22      resposta = input(evento[0] + " (sim/não): ")
23      if resposta == "sim":
24        agua += evento[1]
25      elif resposta == "não":
26        agua += evento[2]
27      else:
28        print("Por favor, responda sim ou não")
29    dia += 1
```

  Observe que recuamos a linha de resposta e a instrução if / else. Isso significa que tudo está no loop. Você pode fazer isso destacando as linhas que deseja recuar e pressionando a tecla "tab" no teclado.

- Execute seu código. O que acontece se você responder algo diferente de sim ou não? Deveria fazer a pergunta novamente. Mas o que acontece se você responder sim ou não? Ainda está fazendo a pergunta novamente! Caramba!

- O loop "while True" é um loop que sempre continuará em loop. Isso é bastante útil se você quiser ir para sempre, mas não queremos ir para sempre, só queremos ir até que o usuário responda sim ou não! Felizmente, temos um comando especial que pode nos interromper, mesmo que seja um comando que duraria para sempre. Esse comando é "break". Vamos adicionar um comando de interrupção quando o jogador responder sim ou não.

```
23      if resposta == "sim":
24        agua += evento[1]
25        break
26      elif resposta == "não":
27        agua += evento[2]
28        break
29      else:
```

- Execute seu código novamente. Agora você só deve receber a mesma pergunta novamente se não responder sim ou não.

  As coisas estão funcionando bem agora, mas há mais um pequeno problema que alguns jogadores podem enfrentar. O que acontece se, em vez de responder "sim", o jogador responder "Sim" com Y maiúsculo?

- Python faz distinção entre maiúsculas e minúsculas. Isso significa que, se você estiver comparando duas cadeias, elas pensarão que são cadeias diferentes apenas porque possuem letras maiúsculas diferentes. Felizmente, isso é fácil de corrigir. Python possui um comando que você pode usar em strings para torná-las todas em minúsculas. Altere sua linha de resposta para o seguinte:

```
22      resposta = input(evento[0] + " (sim/não): ").lower()
23      if resposta == "sim":
24        agua += evento[1]
```

- Execute seu código. Agora ele deve aceitar "Sim", "Sim", "Sim" e qualquer outra combinação de letras maiúsculas.

Parabéns!

Você terminou este projeto. Bem feito! Tente jogar o seu jogo e veja se você consegue sobreviver por mais tempo que seus amigos. Quer mais? Experimente os desafios abaixo ou dê uma olhada em nossos projetos do Scratch ou em projetos dos anos anteriores do Moonhack.

Desafio: adicione mais eventos

Existem alguns eventos diferentes, mas você pode adicionar mais?
Dica: se você tiver dificuldades, tente copiar e editar um dos eventos que já estão lá.

Desafio: jogo da Terra

Você pode modificar o jogo para ser sobre a Terra em vez de Marte? Pense nos tipos de eventos que podem acontecer e nas escolhas que um jogador pode ter que enfrentar.

Desafio avançado: reproduza novamente, Sam

Você pode dar ao jogador a opção de continuar jogando depois de perder o jogo?
Dica: você deseja que todo o seu jogo se repita.