python in pieces

A 2Simple Secondary Product

LESSON SOLUTIONS

Level 3

www.2simple.com

Contents

Lesson 1- Lists

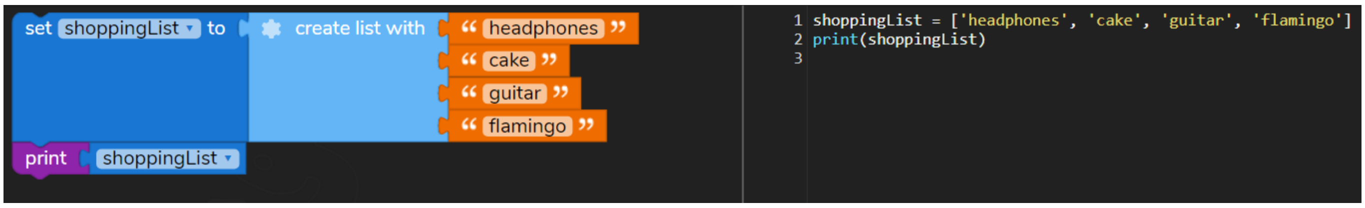
Stage 1- Create a list

I need to buy some gifts for my family - I need some headphones, some cake, a guitar and an inflatable flamingo.
Let's put these names in a list. In Python, a list is an ordered collection of items. In our case we will have a list of four strings.

Tasks

- Make a list called 'shoppingList' which contains the following strings: "headphones", "cake", "guitar", "flamingo". You can use the 'create list with' block, or in Python you can write 'shoppingList=["headphones", ...etc]'.
- Print the list and check that you can see the value of the variable in the debug panel (bottom-right) and the printed list in the output panel (bottom left)

Solution



```python
shoppingList = ['headphones', 'cake', 'guitar', 'flamingo']
print(shoppingList)
```

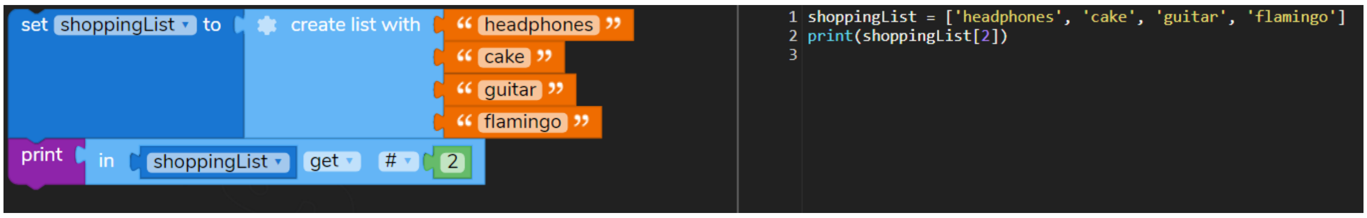
Stage 2- Printing items

The items in a Python list can be accessed in order, but starting at zero rather than one. For example
- shoppingList[0] refers to the first item in the list.
- shoppingList[1] is the second item.

Tasks

- Print the name of the third item in the shopping list using the shoppingList[i] notation. Remember that the index starts from 0 not 1. What is the correct value of the index 'i'?.

Solution



```python
shoppingList = ['headphones', 'cake', 'guitar', 'flamingo']
print(shoppingList[2])
```

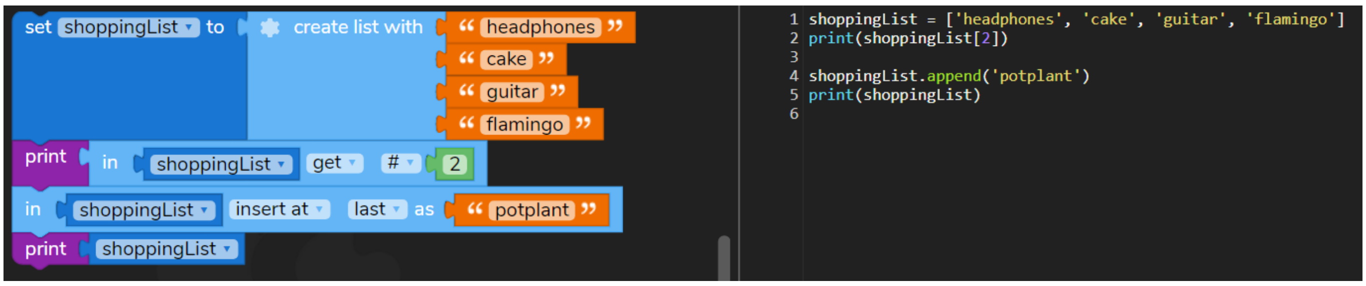
Stage 3- Adding an item

I forgot that my mother wants a pot-plant.
In Python, you can append to the end of a list by using the 'append' method.

Tasks

- Add "potplant" to the end of the list. You can use the 'insert at last' block or the 'shoppingList.append' method.
- Print the list to the output panel. Check that it contains five items now

Solution


set shoppingList to create list with "headphones" "cake" "guitar" "flamingo"
print in shoppingList get # 2
in shoppingList insert at last as "potplant"
print shoppingList

```
shoppingList = ['headphones', 'cake', 'guitar', 'flamingo']
print(shoppingList[2])

shoppingList.append('potplant')
print(shoppingList)

```

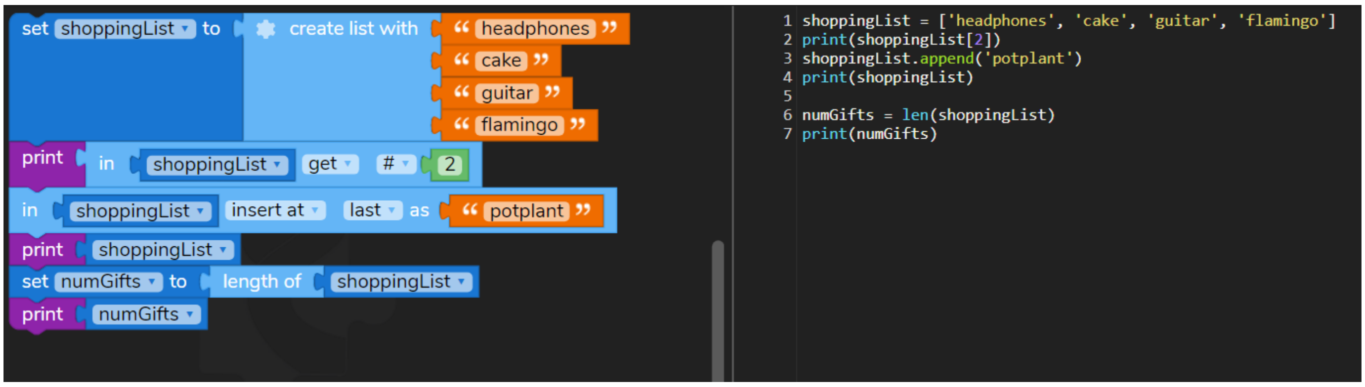
Stage 4- Find the length of the list

In Python, we can use the 'len' function to find the length of a string or a list.

Tasks

- Create a variable called 'numGifts' and assign it a value of 'len(shoppingList)'. Use the 'length of' block or the 'len()' function.
- Print the value of 'numGifts' to the output panel

Solution


set shoppingList to create list with "headphones" "cake" "guitar" "flamingo"
print in shoppingList get # 2
in shoppingList insert at last as "potplant"
print shoppingList
set numGifts to length of shoppingList
print numGifts

```
shoppingList = ['headphones', 'cake', 'guitar', 'flamingo']
print(shoppingList[2])
shoppingList.append('potplant')
print(shoppingList)

numGifts = len(shoppingList)
print(numGifts)
```

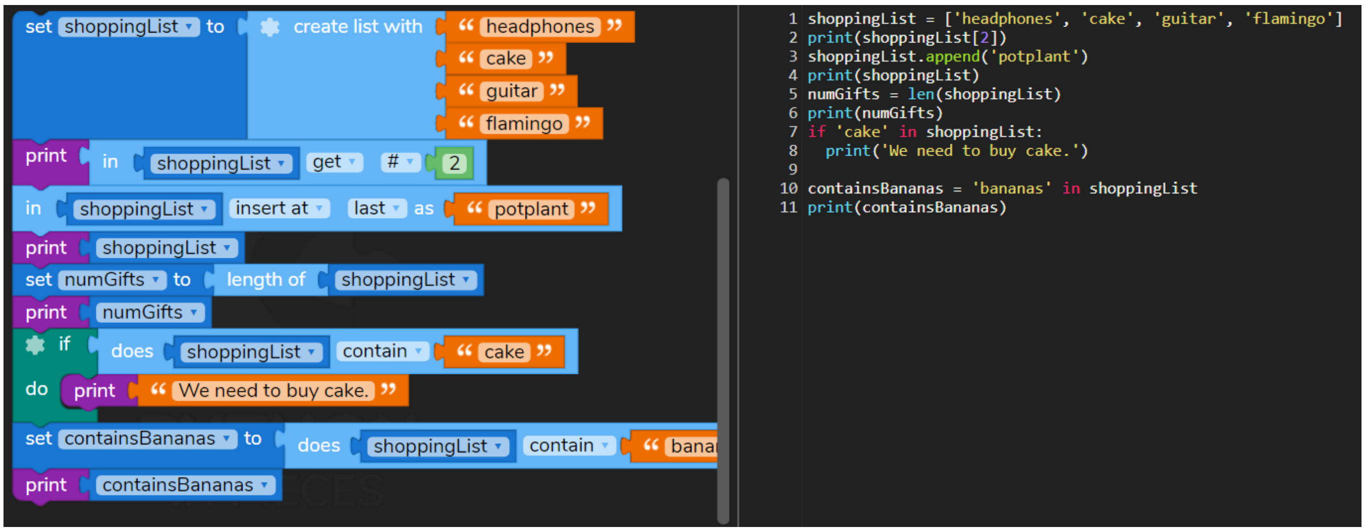
Stage 5- Query a list

Did I remember to put ‘cake’ in the list?
- What about ‘bananas’?
- To answer these kinds of questions we can use the ‘in’ keyword, which tells you if a list contains an item.

Tasks

- Use an ‘if’ statement to print the text “We need to buy cake” if the list contains ‘cake’. You can use the ‘does list contain’ block or you can write ‘if “cake” in shoppingList’.
- Make a variable called ‘containsBananas’. Set its value equal to the value of the statement ‘ “bananas” in shoppingList’. Print the value of ‘containsBananas’ - this should print ‘False’ to the output panel because the list does not contain ‘bananas’.

Solution



```
set shoppingList to create list with “headphones” “cake” “guitar” “flamingo”
print in shoppingList get # 2
in shoppingList insert at last as “potplant”
print shoppingList
set numGifts to length of shoppingList
print numGifts
if does shoppingList contain “cake”
do print “We need to buy cake.”
set containsBananas to does shoppingList contain “bana
print containsBananas
```

```python
shoppingList = ['headphones', 'cake', 'guitar', 'flamingo']
print(shoppingList[2])
shoppingList.append('potplant')
print(shoppingList)
numGifts = len(shoppingList)
print(numGifts)
if 'cake' in shoppingList:
  print('We need to buy cake.')

containsBananas = 'bananas' in shoppingList
print(containsBananas)
```

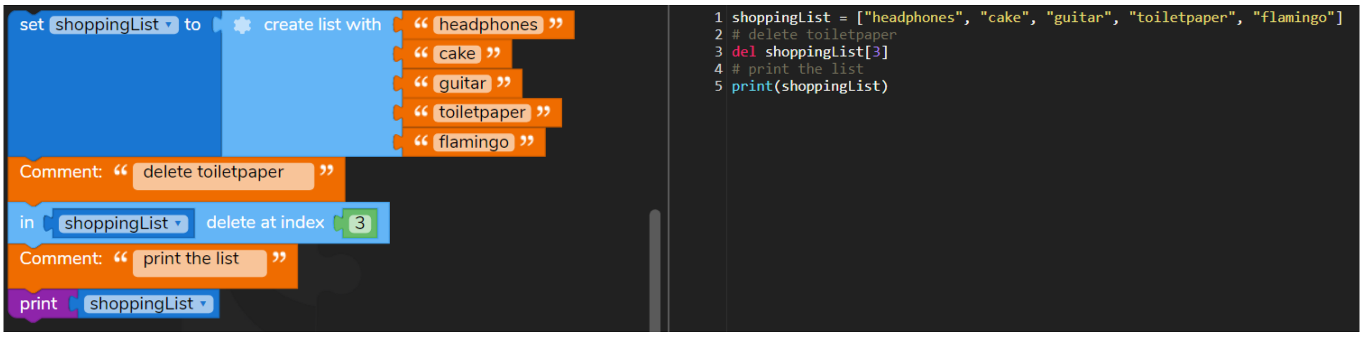
Stage 6- Remove an item

To remove from a list we can use Python's 'del' keyword. You have to specify which index to delete.
For example 'del shoppingList[1]' will delete the second item in the list because the index starts from zero. 'del shoppingList[i]' will delete the item at index i.

Tasks

- Someone has put 'toiletpaper' in my list by mistake. What index is it? Use the 'del' keyword to delete it and then print the list. Use 'del shoppingList[i]' and remember that the index starts from zero.

Solution



```
set shoppingList to create list with "headphones" "cake" "guitar" "toiletpaper" "flamingo"
Comment: "delete toiletpaper"
in shoppingList delete at index 3
Comment: "print the list"
print shoppingList
```

```
shoppingList = ["headphones", "cake", "guitar", "toiletpaper", "flamingo"]
# delete toiletpaper
del shoppingList[3]
# print the list
print(shoppingList)
```

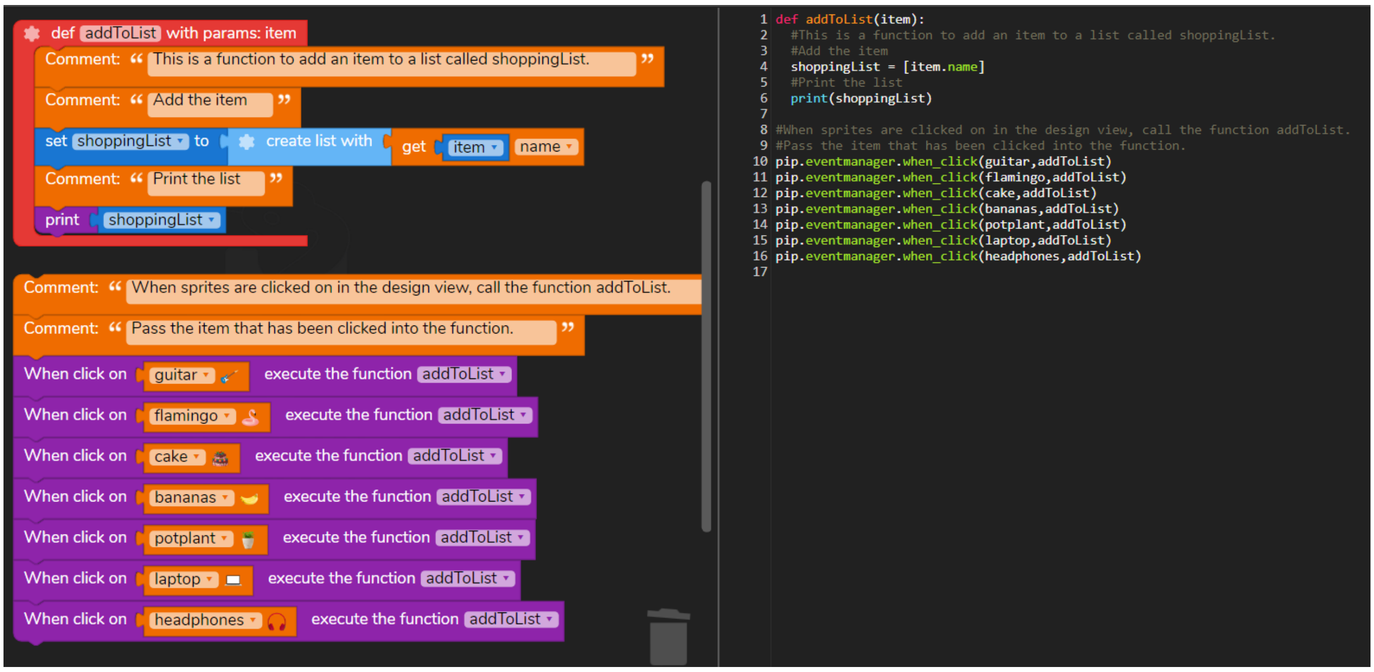
Stage 7- Making it interactive

Look at design view, it represents a shop with different items that you can buy. Look at the code, what do you think it does?
Let's add some comments to explain. In Python you can use a hash sign (#) to make any line into a comment, or you can surround some text with three double-quote symbols (" " ").

Tasks

- Add a few comments to explain to someone else what you think the code is meant to do. Use a hash (#) or three double-quote symbols (" " ")

Solution



```python
def addToList(item):
  #This is a function to add an item to a list called shoppingList.
  #Add the item
  shoppingList = [item.name]
  #Print the list
  print(shoppingList)

#When sprites are clicked on in the design view, call the function addToList.
#Pass the item that has been clicked into the function.
pip.eventmanager.when_click(guitar,addToList)
pip.eventmanager.when_click(flamingo,addToList)
pip.eventmanager.when_click(cake,addToList)
pip.eventmanager.when_click(bananas,addToList)
pip.eventmanager.when_click(potplant,addToList)
pip.eventmanager.when_click(laptop,addToList)
pip.eventmanager.when_click(headphones,addToList)
```

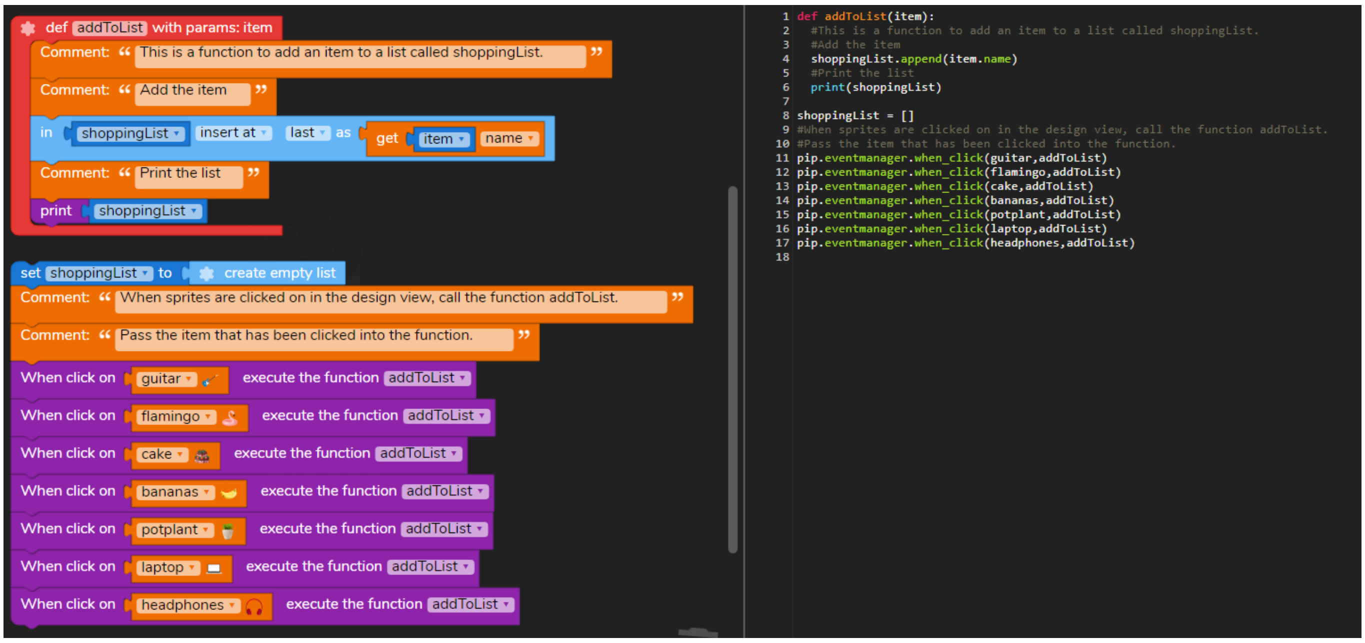
Stage 8- Append to the lists - debug

The code doesn't work properly at the moment. Click on some items and see what is printed out. You should see that the shopping list only ever contains one item - the last one you clicked on. The problem is that we are re-assigning the variable 'shoppingList' rather than appending to the list. Let's fix it.

Tasks

- Define the variable called 'shoppingList' at the top of your program, outside of the 'addToList' function. Initialize 'shoppingList' it to an empty list. You can use the 'create empty list' block or you can write: 'shoppingList = [ ]'
- Change the code inside the function so that 'item.name' is appended to the list rather than the list being re-assigned.
- Check that the code works now. Click on a few items so that your shopping list isn't empty.

Solution



```
def addToList with params: item
  Comment: " This is a function to add an item to a list called shoppingList. "
  Comment: " Add the item "
  in shoppingList insert at last as get item name
  Comment: " Print the list "
  print shoppingList

set shoppingList to create empty list
Comment: " When sprites are clicked on in the design view, call the function addToList. "
Comment: " Pass the item that has been clicked into the function. "
When click on guitar execute the function addToList
When click on flamingo execute the function addToList
When click on cake execute the function addToList
When click on bananas execute the function addToList
When click on potplant execute the function addToList
When click on laptop execute the function addToList
When click on headphones execute the function addToList
```

```python
def addToList(item):
  #This is a function to add an item to a list called shoppingList.
  #Add the item
  shoppingList.append(item.name)
  #Print the list
  print(shoppingList)

shoppingList = []
#When sprites are clicked on in the design view, call the function addToList.
#Pass the item that has been clicked into the function.
pip.eventmanager.when_click(guitar,addToList)
pip.eventmanager.when_click(flamingo,addToList)
pip.eventmanager.when_click(cake,addToList)
pip.eventmanager.when_click(bananas,addToList)
pip.eventmanager.when_click(potplant,addToList)
pip.eventmanager.when_click(laptop,addToList)
pip.eventmanager.when_click(headphones,addToList)
```

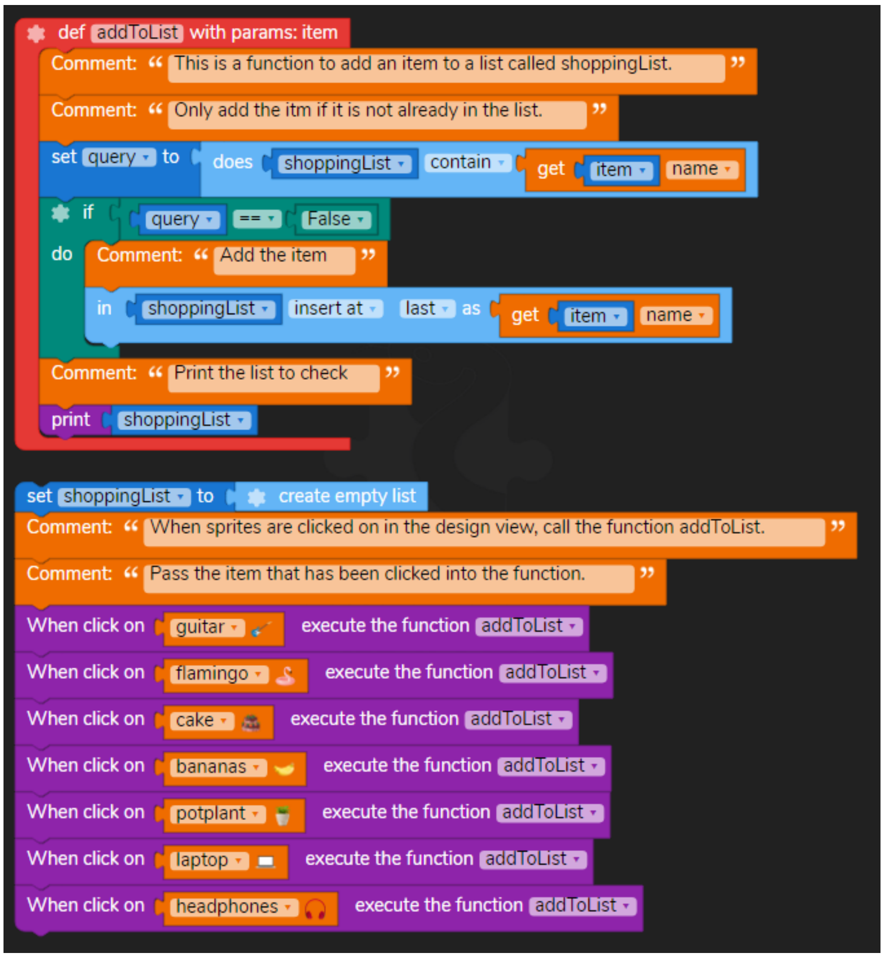
Stage 9- Only one of each item

At the moment you can add the same item's name multiple times to the list. Let's make it so that if an item's name is already in the list you cannot add it again.

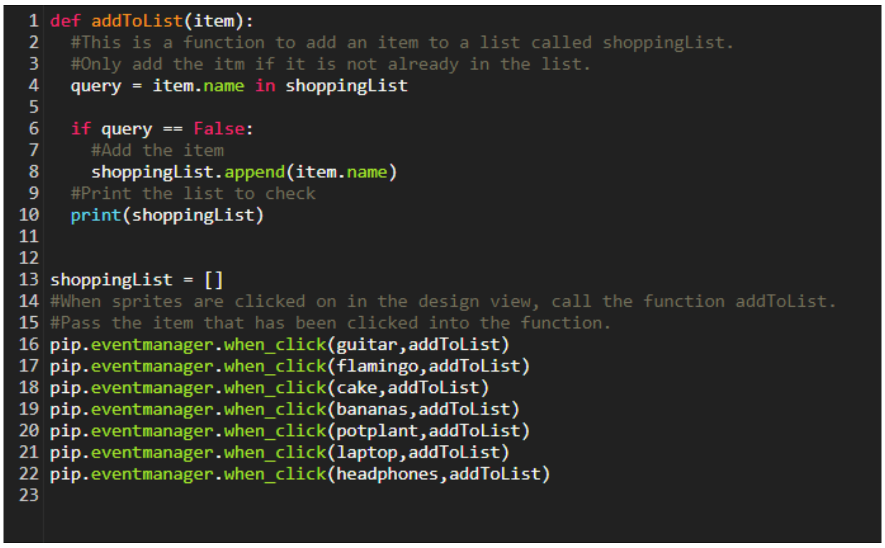
Tasks

- Before appending to the list, use an 'if' statement and the 'in' keyword to check if the shoppingList contains the item's name. Only add it if it is not in the list already.
- Check your program by clicking on the flamingo three times. At the end, your shopping list should contain just one flamingo.

Solution



```
def addToList with params: item
    Comment: “ This is a function to add an item to a list called shoppingList. ”
    Comment: “ Only add the itm if it is not already in the list. ”
    set query to does shoppingList contain get item name
    if query == False
    do  Comment: “ Add the item ”
        in shoppingList insert at last as get item name
    Comment: “ Print the list to check ”
    print shoppingList

set shoppingList to create empty list
Comment: “ When sprites are clicked on in the design view, call the function addToList. ”
Comment: “ Pass the item that has been clicked into the function. ”
When click on guitar execute the function addToList
When click on flamingo execute the function addToList
When click on cake execute the function addToList
When click on bananas execute the function addToList
When click on potplant execute the function addToList
When click on laptop execute the function addToList
When click on headphones execute the function addToList
```



```python
def addToList(item):
  #This is a function to add an item to a list called shoppingList.
  #Only add the itm if it is not already in the list.
  query = item.name in shoppingList

  if query == False:
    #Add the item
    shoppingList.append(item.name)
  #Print the list to check
  print(shoppingList)


shoppingList = []
#When sprites are clicked on in the design view, call the function addToList.
#Pass the item that has been clicked into the function.
pip.eventmanager.when_click(guitar,addToList)
pip.eventmanager.when_click(flamingo,addToList)
pip.eventmanager.when_click(cake,addToList)
pip.eventmanager.when_click(bananas,addToList)
pip.eventmanager.when_click(potplant,addToList)
pip.eventmanager.when_click(laptop,addToList)
pip.eventmanager.when_click(headphones,addToList)
```

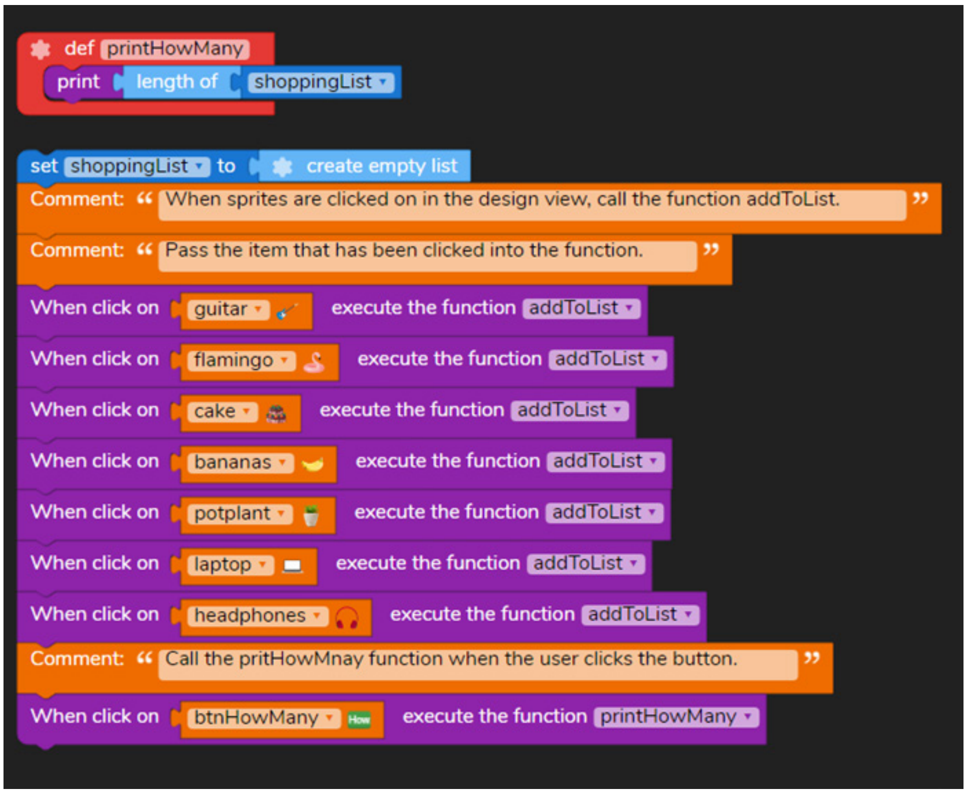
Stage 10- Count the number of items

How many items are in the list? Let's add a button which tells us.

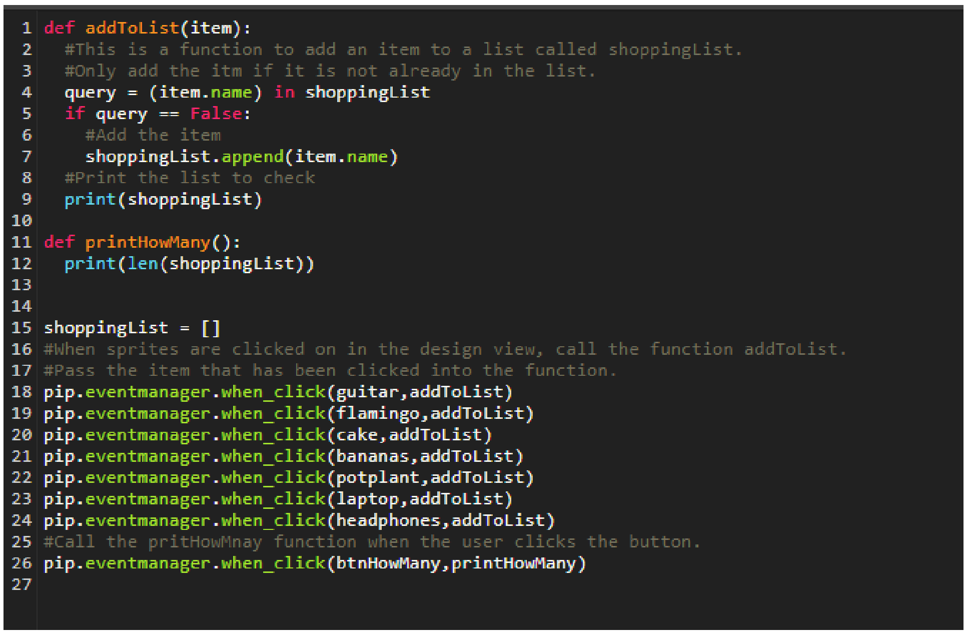
Tasks

- Add two buttons. One should say "How many" and it should be called it 'btnHowMany'. The other button should say "Remove" and should be called it 'btnRemove'.
- Make a function called 'printHowMany'. Add code to execute the function when 'btnHowMany' is clicked.
- Inside the 'printHowMany' function print the length of 'shoppingList' to the output panel.
- Test your code by adding three elements to the shoppingList and clicking the "Print length" button. It should say '3' in the output panel.

Solution



```
def printHowMany
  print length of shoppingList

set shoppingList to create empty list
Comment: “ When sprites are clicked on in the design view, call the function addToList. ”
Comment: “ Pass the item that has been clicked into the function. ”
When click on guitar execute the function addToList
When click on flamingo execute the function addToList
When click on cake execute the function addToList
When click on bananas execute the function addToList
When click on potplant execute the function addToList
When click on laptop execute the function addToList
When click on headphones execute the function addToList
Comment: “ Call the pritHowMnay function when the user clicks the button. ”
When click on btnHowMany execute the function printHowMany
```



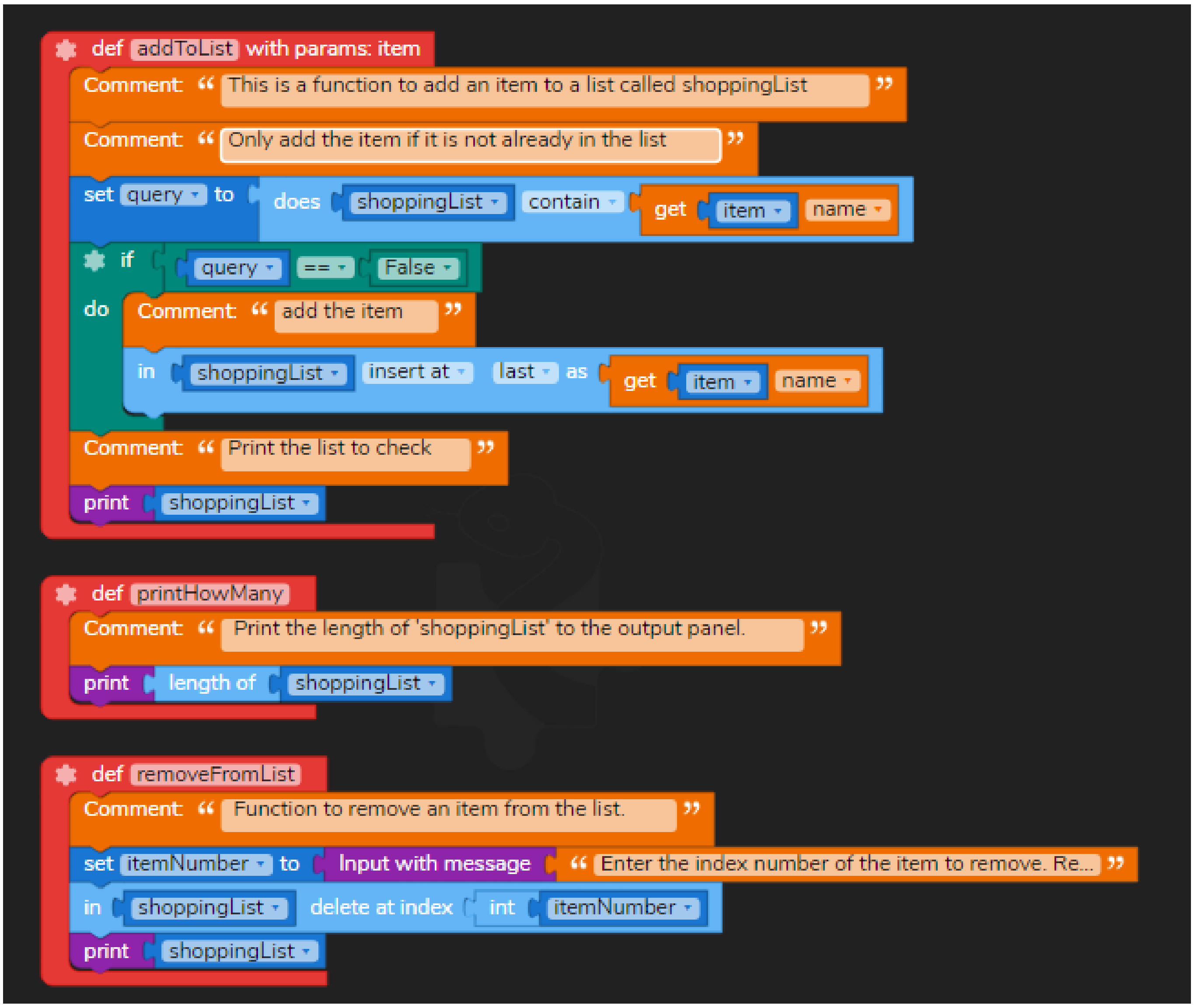
```python
def addToList(item):
  #This is a function to add an item to a list called shoppingList.
  #Only add the itm if it is not already in the list.
  query = (item.name) in shoppingList
  if query == False:
    #Add the item
    shoppingList.append(item.name)
  #Print the list to check
  print(shoppingList)

def printHowMany():
  print(len(shoppingList))


shoppingList = []
#When sprites are clicked on in the design view, call the function addToList.
#Pass the item that has been clicked into the function.
pip.eventmanager.when_click(guitar,addToList)
pip.eventmanager.when_click(flamingo,addToList)
pip.eventmanager.when_click(cake,addToList)
pip.eventmanager.when_click(bananas,addToList)
pip.eventmanager.when_click(potplant,addToList)
pip.eventmanager.when_click(laptop,addToList)
pip.eventmanager.when_click(headphones,addToList)
#Call the pritHowMnay function when the user clicks the button.
pip.eventmanager.when_click(btnHowMany,printHowMany)
```

Stage 11- Remove an item

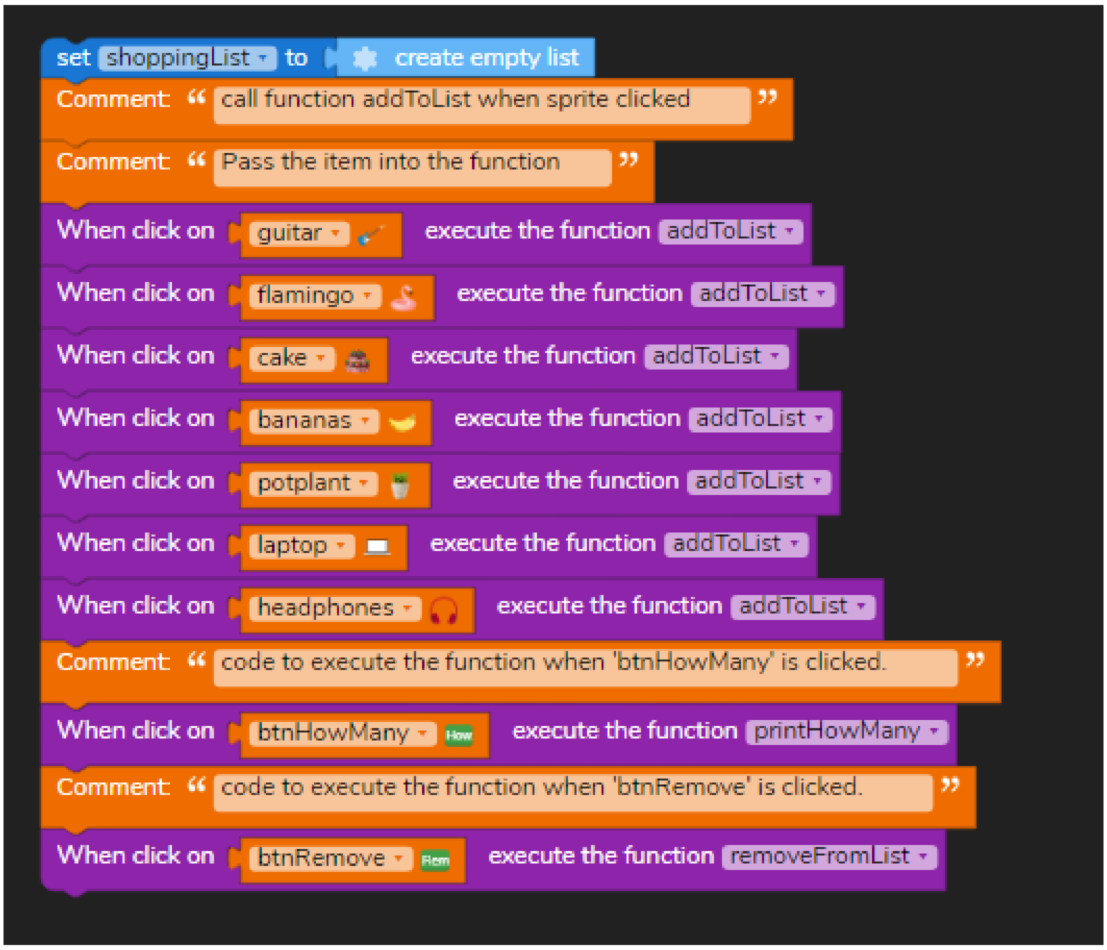
Tasks

- Make a function called ‘removeFromList’. Add code to execute the function when ‘btnRemove’ is clicked.
- Inside the ‘removeFromList’ function, present them with an input box and ask them to select which index to remove. Remind the user that the index starts from zero.
- Remove the item at that index and print the revised shopping list. Hint - the value returned from the ‘input’ function is a string. For example it might be “2”. You will need to convert the string into an integer (eg. 2).
- Test your code by adding some items to the list and then deleting the item at index zero until your list is empty.

Solution



```
def addToList with params: item
    Comment: “ This is a function to add an item to a list called shoppingList ”
    Comment: “ Only add the item if it is not already in the list ”
    set query to does shoppingList contain get item name
    if query == False
    do  Comment: “ add the item ”
        in shoppingList insert at last as get item name
    Comment: “ Print the list to check ”
    print shoppingList

def printHowMany
    Comment: “ Print the length of 'shoppingList' to the output panel. ”
    print length of shoppingList

def removeFromList
    Comment: “ Function to remove an item from the list. ”
    set itemNumber to Input with message “ Enter the index number of the item to remove. Re... ”
    in shoppingList delete at index int itemNumber
    print shoppingList
```



```
set shoppingList to create empty list
Comment: "call function addToList when sprite clicked"
Comment: "Pass the item into the function"
When click on guitar execute the function addToList
When click on flamingo execute the function addToList
When click on cake execute the function addToList
When click on bananas execute the function addToList
When click on potplant execute the function addToList
When click on laptop execute the function addToList
When click on headphones execute the function addToList
Comment: "code to execute the function when 'btnHowMany' is clicked."
When click on btnHowMany execute the function printHowMany
Comment: "code to execute the function when 'btnRemove' is clicked."
When click on btnRemove execute the function removeFromList
```

```python
def addToList(item):
  # function to add an item to the list called shoppingList
  # Only add the item if it is not already on the list
  if item.name in shoppingList:
    print(shoppingList)
  else:
    # add the item
    shoppingList.append(item.name)
    #print the list to check
    print(shoppingList)

def removeFromList():
  #Function to remove an item from the list
  itemNumber = input("Enter the index number of the item to remove.")
  del shoppingList[int(itemNumber)]
  print(shoppingList)

def printHowMany():
  #Print the length of shoppingList
  print(len(shoppingList))


shoppingList = []
#click events, when you click and item it is added to the list
pip.eventmanager.when_click(guitar,addToList)
pip.eventmanager.when_click(flamingo,addToList)
pip.eventmanager.when_click(cake,addToList)
pip.eventmanager.when_click(bananas,addToList)
pip.eventmanager.when_click(potplant,addToList)
pip.eventmanager.when_click(laptop,addToList)
pip.eventmanager.when_click(headphones,addToList)
pip.eventmanager.when_click(btnHowMany,printHowMany)
pip.eventmanager.when_click(btnRemove,removeFromList)
```

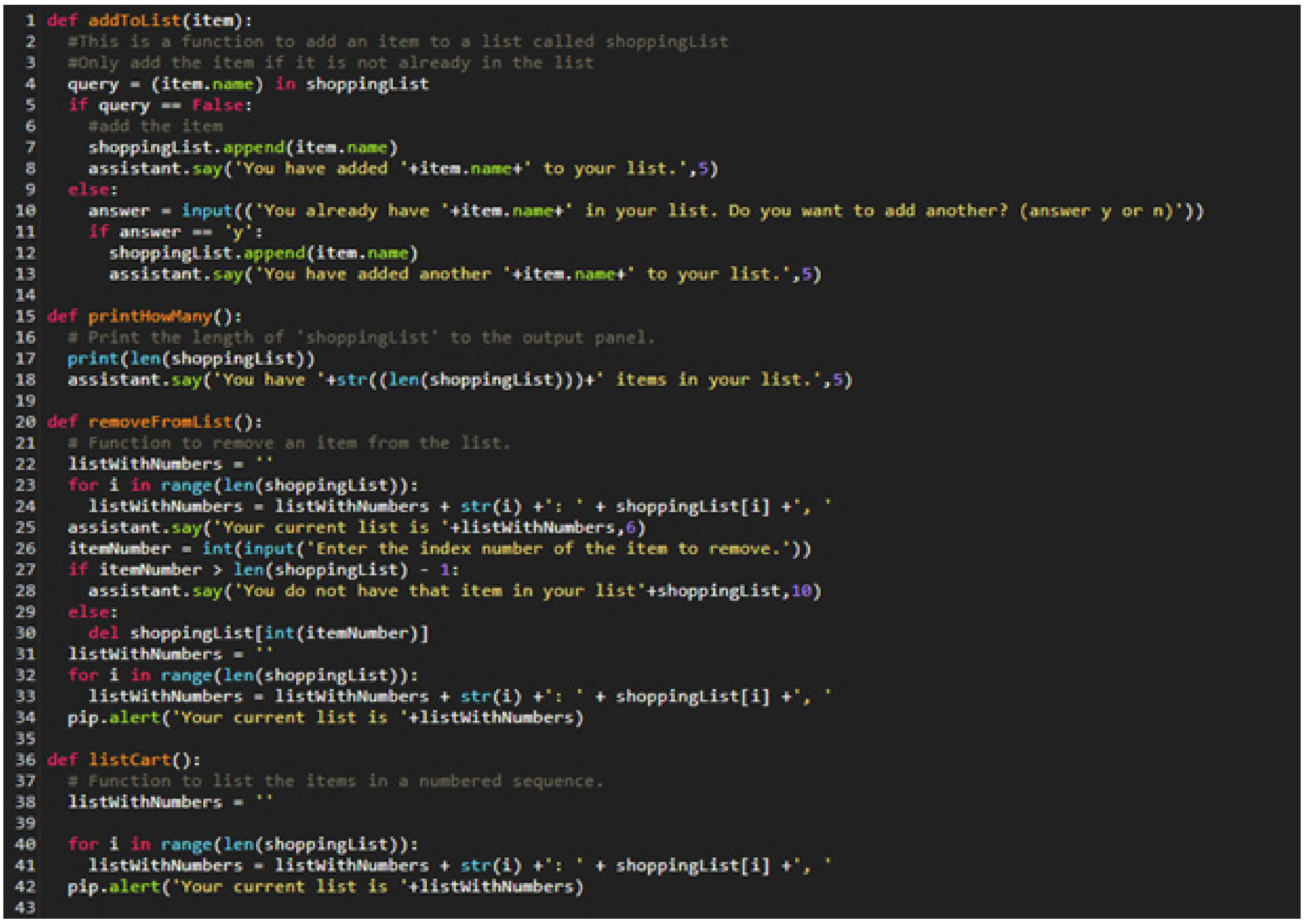
Challenge

Try out the following challenges.

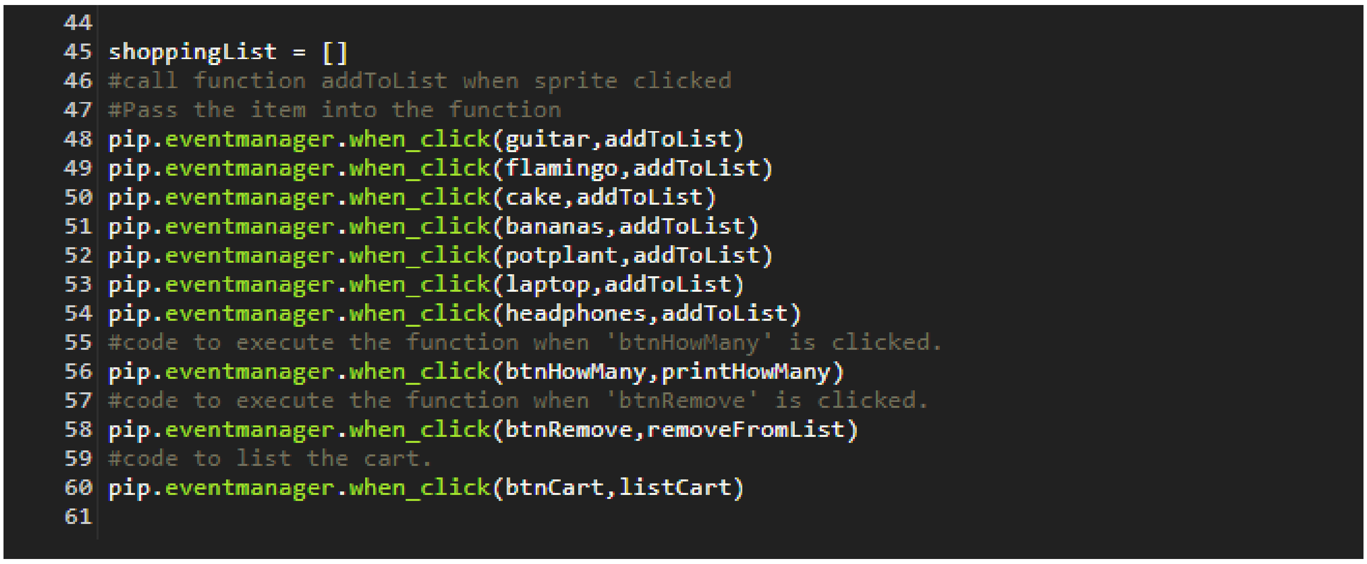
Suggestions

- Instead of printing to the console, add a shop assistant character who tells the user the answers.
- Instead of stopping users adding multiple items, instead ask them if they are sure that's what they want to do.
- Stop users deleting items that are not there - for example if a shopper has 3 items then they should not be able to delete the item at index 5.
- To help the user, when you ask for the item to be deleted, print the item numbers alongside the item names. Use a for loop to do this - for example 0: flamingo, 1: cake, 2: laptop.
- Instead of asking the user which index they want to delete, ask them which name they want to delete and then remove that name.
- Maintain several lists at once - shoppers should be asked their name and the code should maintain a different list for each person.

Possible Solution



```
def addToList(item):
  #This is a function to add an item to a list called shoppingList
  #Only add the item if it is not already in the list
  query = (item.name) in shoppingList
  if query == False:
    #add the item
    shoppingList.append(item.name)
    assistant.say('You have added '+item.name+' to your list.',5)
  else:
    answer = input(('You already have '+item.name+' in your list. Do you want to add another? (answer y or n)'))
    if answer == 'y':
      shoppingList.append(item.name)
      assistant.say('You have added another '+item.name+' to your list.',5)

def printHowMany():
  # Print the length of 'shoppingList' to the output panel.
  print(len(shoppingList))
  assistant.say('You have '+str((len(shoppingList)))+' items in your list.',5)

def removeFromList():
  # Function to remove an item from the list.
  listWithNumbers = ''
  for i in range(len(shoppingList)):
    listWithNumbers = listWithNumbers + str(i) +': ' + shoppingList[i] +', '
  assistant.say('Your current list is '+listWithNumbers,6)
  itemNumber = int(input('Enter the index number of the item to remove.'))
  if itemNumber > len(shoppingList) - 1:
    assistant.say('You do not have that item in your list'+shoppingList,10)
  else:
    del shoppingList[int(itemNumber)]
  listWithNumbers = ''
  for i in range(len(shoppingList)):
    listWithNumbers = listWithNumbers + str(i) +': ' + shoppingList[i] +', '
  pip.alert('Your current list is '+listWithNumbers)

def listCart():
  # Function to list the items in a numbered sequence.
  listWithNumbers = ''

  for i in range(len(shoppingList)):
    listWithNumbers = listWithNumbers + str(i) +': ' + shoppingList[i] +', '
  pip.alert('Your current list is '+listWithNumbers)

```



```python

shoppingList = []
#call function addToList when sprite clicked
#Pass the item into the function
pip.eventmanager.when_click(guitar,addToList)
pip.eventmanager.when_click(flamingo,addToList)
pip.eventmanager.when_click(cake,addToList)
pip.eventmanager.when_click(bananas,addToList)
pip.eventmanager.when_click(potplant,addToList)
pip.eventmanager.when_click(laptop,addToList)
pip.eventmanager.when_click(headphones,addToList)
#code to execute the function when 'btnHowMany' is clicked.
pip.eventmanager.when_click(btnHowMany,printHowMany)
#code to execute the function when 'btnRemove' is clicked.
pip.eventmanager.when_click(btnRemove,removeFromList)
#code to list the cart.
pip.eventmanager.when_click(btnCart,listCart)

```

Lesson 2- Nested Selection

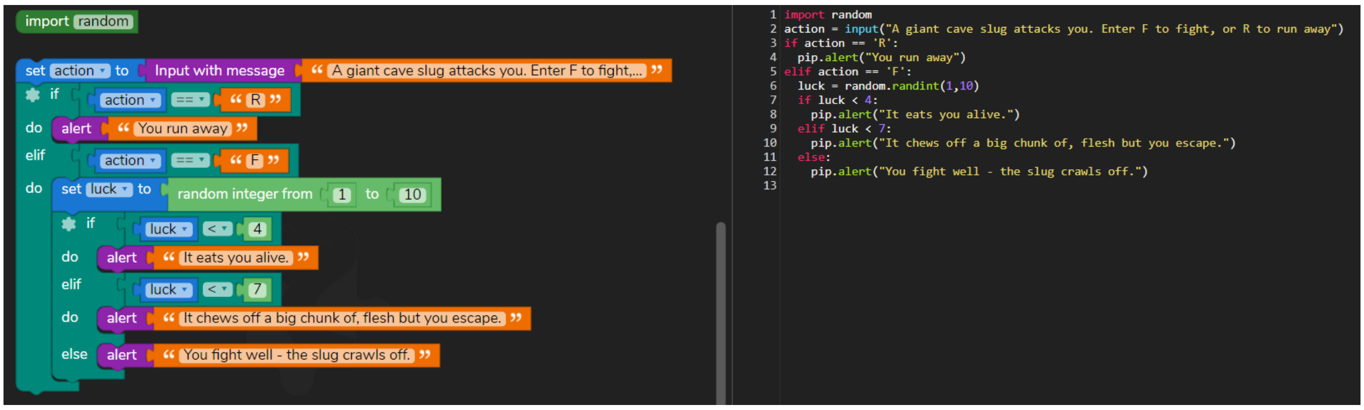
Stage 1- if, elif, else statements

When you put an 'if' statement inside another 'if' statement they are called 'nested'.
You can create complicated branching code in this way. In fact we use nested if/else decision making in real life all the time. But real life is boring.
And watch out, a giant cave slug has just attacked you.

Tasks

- After the 'if' statement, add an 'elif' statement that checks if action is 'F'. If so, make a variable called 'luck' and assign it a random value between 1 and 10. Use 'random.randint'.
- Use a nested 'if' statement inside the 'elif' block to check if 'luck' is less than 4. If so, alert the text 'It eats you alive'. Use 'pip.alert' to do this.
- Add an 'elif' statement to check if luck is less than 7. If so, alert the text 'It chews off a big chunk of flesh, but you escape'.
- Add an 'else' statement which alerts the text 'You fight well - the slug crawls off'.
- Test your code a few times. Fight the slug and check that you get random outcomes.

Solution



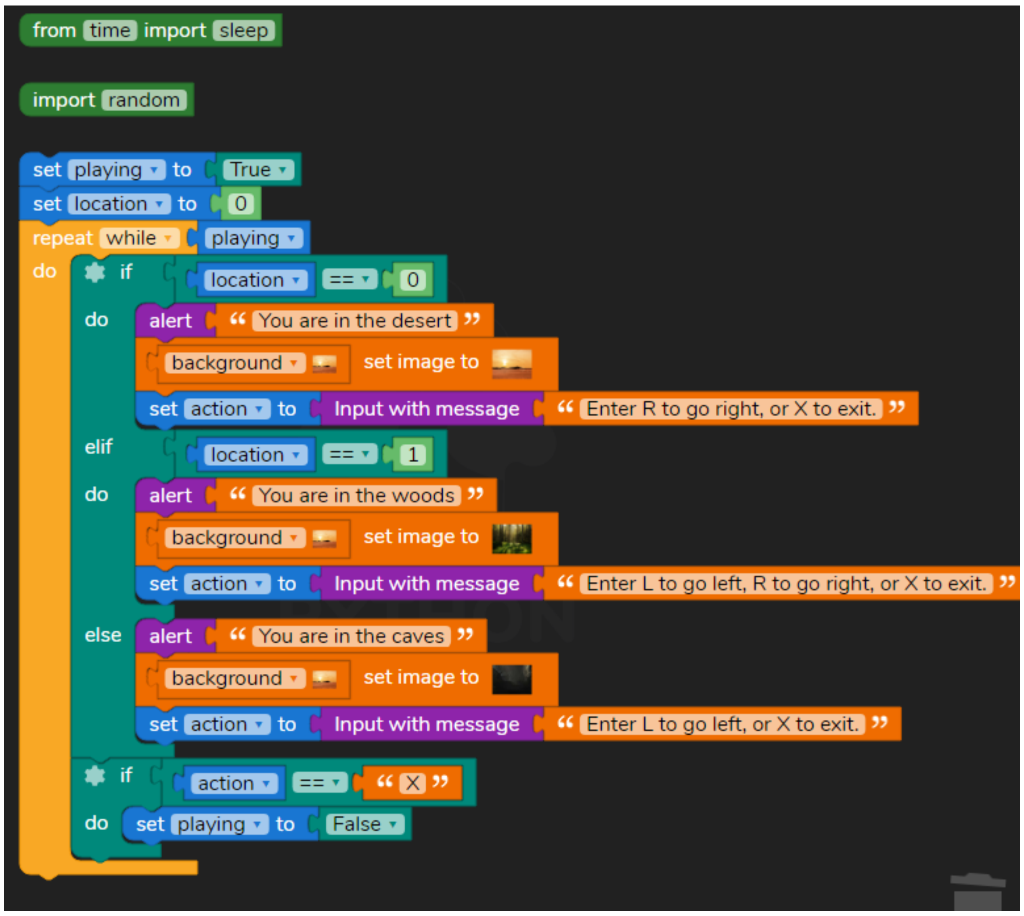
```python
import random
action = input("A giant cave slug attacks you. Enter F to fight, or R to run away")
if action == 'R':
  pip.alert("You run away")
elif action == 'F':
  luck = random.randint(1,10)
  if luck < 4:
    pip.alert("It eats you alive.")
  elif luck < 7:
    pip.alert("It chews off a big chunk of, flesh but you escape.")
  else:
    pip.alert("You fight well - the slug crawls off.")
```

Let's start making an adventure game. There will be three locations in our game. We will call them 0, 1 and 2. They will actually be a desert, the woods and a cave. You start in location 0 (the desert). The user can enter 'L' to go left or 'R' to go right.

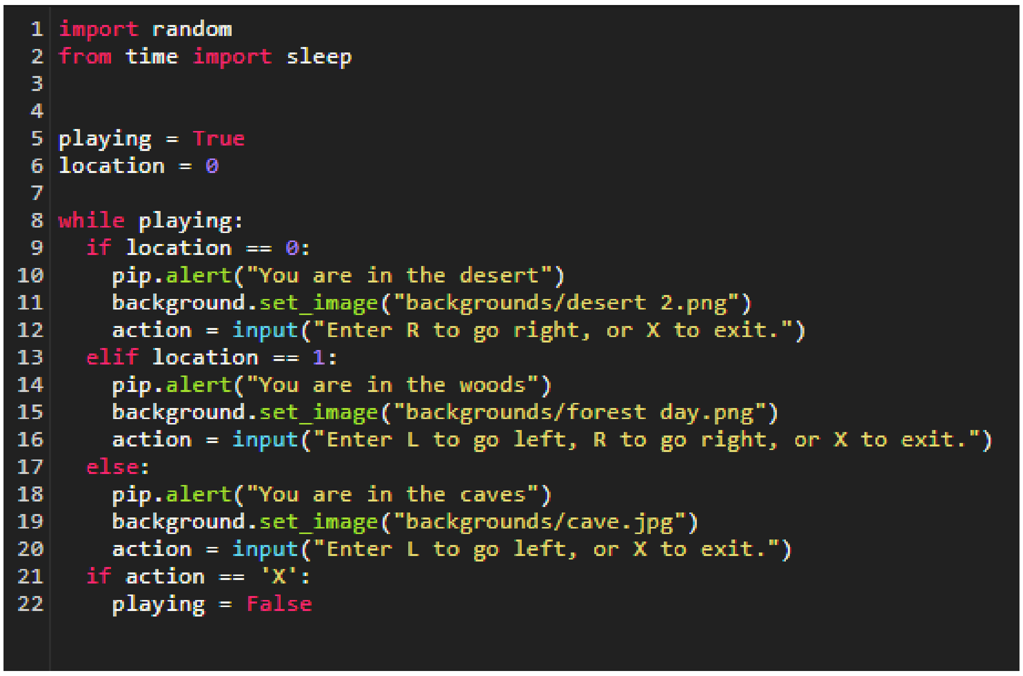
Tasks

- Underneath the 'import' statements, make a variable called 'playing' and initialize it to True, and another called 'location', initialized to 0.
- Make a 'while' loop which runs while playing is True. In Python you can write 'while playing:'
- Next, use an 'if' statement to check if the player is at location 0.
- If so, use 'pip.alert' to say 'You are in the desert', use 'background.set_image' to set the background image to a desert and then use an 'input' statement to ask the user 'Enter R to go right, or X to exit'. Assign their response to a variable called 'action'.
- Use an 'elif' statement to check if location equals 1. If so, alert 'You are in the woods', set the background image to woods and then input 'Enter L to go left, R to go right, or X to exit'. Assign their response to 'action'.
- Add an 'else' statement - this code will be executed if the player is at location 2. Alert the text 'You are in the caves', set the background image to the cave and then input 'Enter L to go left, or X to exit'. Assign their response to 'action'.
- At the end of your while loop, check if 'action' equals 'X' and if so set playing to False. Play the game and check it tells you that you're in the desert. Enter X to exit.

Solution



```
from time import sleep

import random

set playing to True
set location to 0
repeat while playing
do  if location == 0
    do  alert "You are in the desert"
        background set image to
        set action to Input with message "Enter R to go right, or X to exit."
    elif location == 1
    do  alert "You are in the woods"
        background set image to
        set action to Input with message "Enter L to go left, R to go right, or X to exit."
    else alert "You are in the caves"
        background set image to
        set action to Input with message "Enter L to go left, or X to exit."
    if action == "X"
    do  set playing to False
```



```python
import random
from time import sleep


playing = True
location = 0

while playing:
  if location == 0:
    pip.alert("You are in the desert")
    background.set_image("backgrounds/desert 2.png")
    action = input("Enter R to go right, or X to exit.")
  elif location == 1:
    pip.alert("You are in the woods")
    background.set_image("backgrounds/forest day.png")
    action = input("Enter L to go left, R to go right, or X to exit.")
  else:
    pip.alert("You are in the caves")
    background.set_image("backgrounds/cave.jpg")
    action = input("Enter L to go left, or X to exit.")
  if action == 'X':
    playing = False
```

Stage 3- Add functions

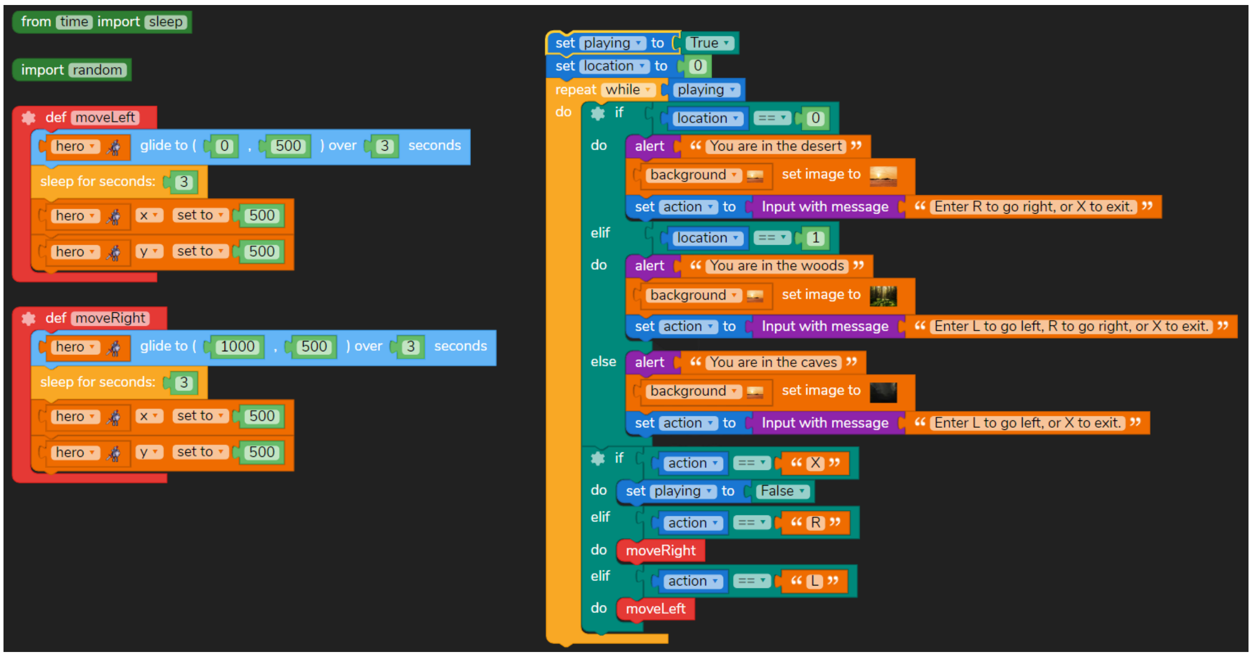
When the user enters action ‘L’ we need to move left, and when they enter ‘R’ we need to move right. We should make functions for these because they will need to contain quite a lot of code. Let’s add some visual elements to make it more interesting

Tasks

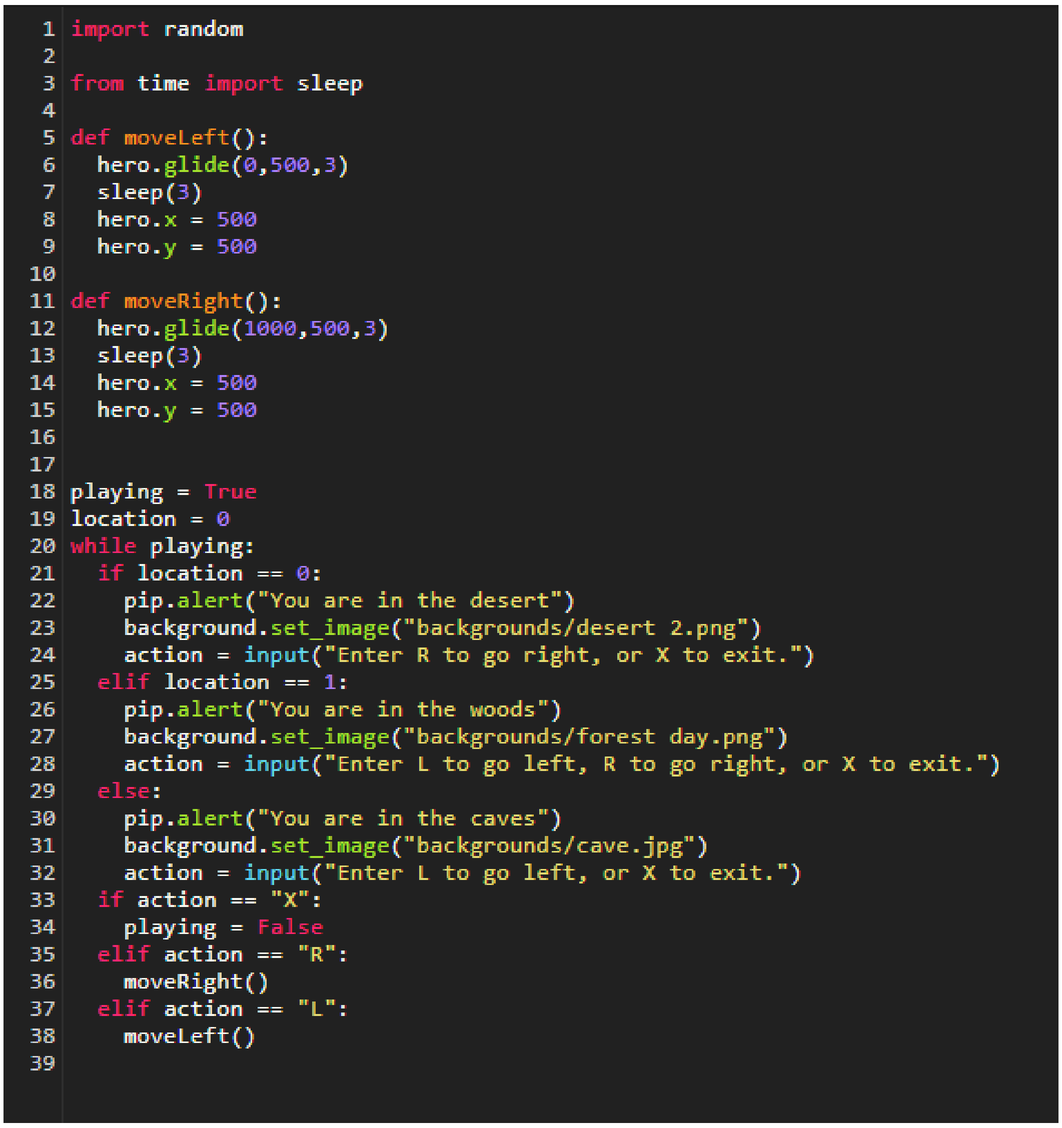
- Before your while loop, define two functions called ‘moveLeft’ and ‘moveRight’. They don’t need any parameters.
- Inside the ‘moveLeft’ function make ‘hero’ glide to the point (0, 500) over 3 seconds. You can use the ‘glide to’ block or write ‘hero.glide(0, 500, 3)’. Make the program sleep for 3 seconds and then set hero.x = 500 and hero.y = 500.
- Inside ‘moveRight’ make ‘hero’ glide to (1000, 500) over 3 seconds. Then, make the program sleep for 3 seconds and reposition the hero.
- After checking if action is ‘X’, use ‘elif’ to check if ‘action’ equals ‘L’. If so execute the function ‘moveLeft’. Use another ‘elif’ to check if action equals ‘R’ and if so execute the function ‘moveRight’.
- Test the program - make the hero move right a few times and then enter ‘X’ to exit. Don’t worry that you always stay in the desert - we haven’t written that code yet.

Solutions on following page...

Solution



```
from time import sleep

import random

def moveLeft
  hero glide to ( 0 , 500 ) over 3 seconds
  sleep for seconds: 3
  hero x set to 500
  hero y set to 500

def moveRight
  hero glide to ( 1000 , 500 ) over 3 seconds
  sleep for seconds: 3
  hero x set to 500
  hero y set to 500

set playing to True
set location to 0
repeat while playing
do  if location == 0
    do   alert “ You are in the desert ”
         background set image to
         set action to Input with message “ Enter R to go right, or X to exit. ”
    elif location == 1
    do   alert “ You are in the woods ”
         background set image to
         set action to Input with message “ Enter L to go left, R to go right, or X to exit. ”
    else alert “ You are in the caves ”
         background set image to
         set action to Input with message “ Enter L to go left, or X to exit. ”
    if action == “ X ”
    do   set playing to False
    elif action == “ R ”
    do   moveRight
    elif action == “ L ”
    do   moveLeft
```



```python
import random

from time import sleep

def moveLeft():
  hero.glide(0,500,3)
  sleep(3)
  hero.x = 500
  hero.y = 500

def moveRight():
  hero.glide(1000,500,3)
  sleep(3)
  hero.x = 500
  hero.y = 500


playing = True
location = 0
while playing:
  if location == 0:
    pip.alert("You are in the desert")
    background.set_image("backgrounds/desert 2.png")
    action = input("Enter R to go right, or X to exit.")
  elif location == 1:
    pip.alert("You are in the woods")
    background.set_image("backgrounds/forest day.png")
    action = input("Enter L to go left, R to go right, or X to exit.")
  else:
    pip.alert("You are in the caves")
    background.set_image("backgrounds/cave.jpg")
    action = input("Enter L to go left, or X to exit.")
  if action == "X":
    playing = False
  elif action == "R":
    moveRight()
  elif action == "L":
    moveLeft()

```

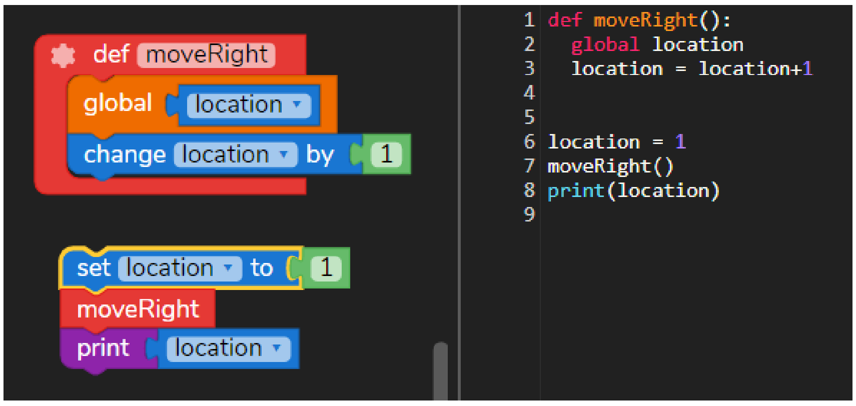
Stage 4- Global variables - debug

Before we go much further we need to recap how variables work in Python. You can have a variables with the same name inside and outside a function:
This prints the word ‘Global value’ not ‘CHANGED VALUE’.
This is good - it means that other programmers can use variables called ‘val’ inside their functions, safe in the knowledge that they aren’t messing up the value used outside by someone else.
In our program we actually do want change the value of ‘location’ inside our moveLeft and moveRight functions, so we need to tell Python to let us change it.
This is called declaring a variable ‘global’, and you need to do it whenever you want to change a ‘global’ variable inside a function

Tasks

- Run the code. Check that you get an error. Python won’t let you change the variable unless you mark it as ‘global’. Fix it by declaring ‘location’ as global. Check that it prints 2.

Solution



```python
def moveRight():
  global location
  location = location+1


location = 1
moveRight()
print(location)
```

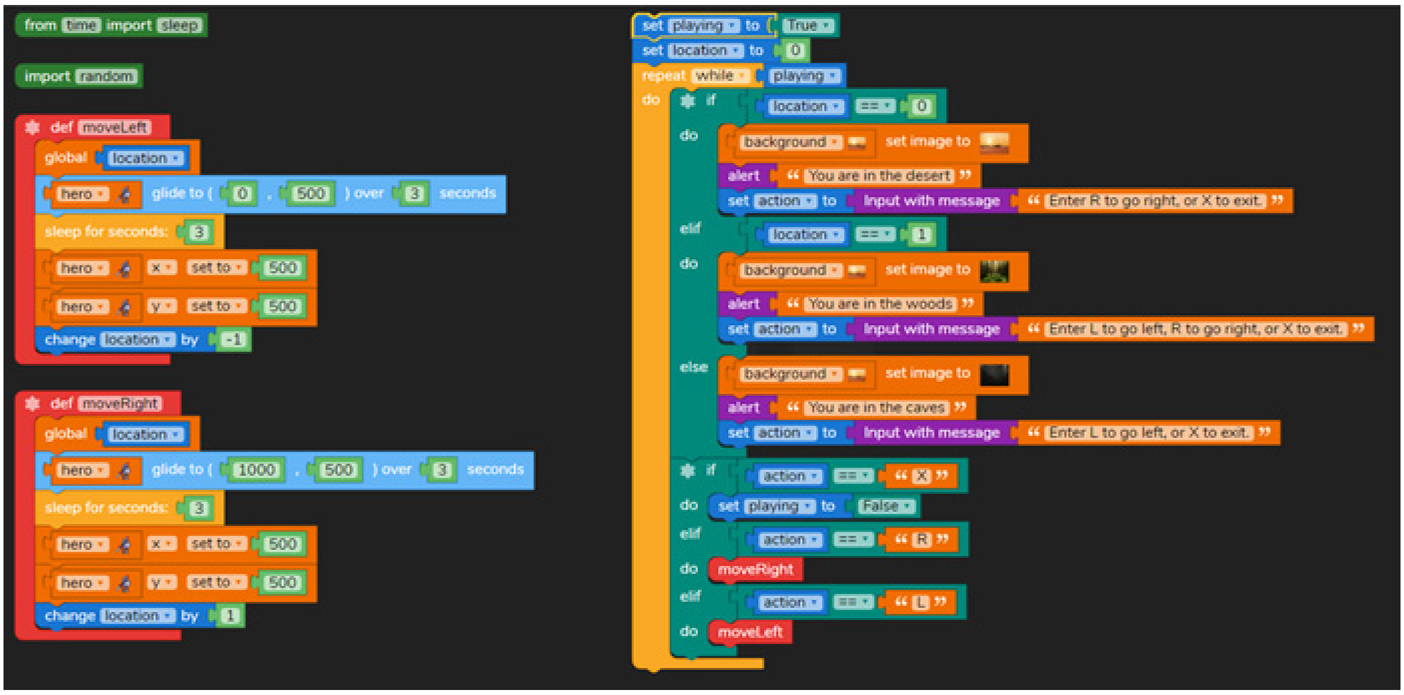
Stage 5- Moving the hero

We need to change the value of the variable ‘location’ inside our functions.

Tasks

- In the ‘moveLeft’ function declare ‘location’ as global. After the 3 second sleep, change ‘location’ to ‘location’ minus one.
- In the ‘moveRight’ function declare ‘location’ as global. After the 3 second sleep, change ‘location’ to ‘location’ plus one.
- Test your code. Go right and check that you reach the woods. In the debug panel you should see that location is 1. Use X to exit.

Solution on following page...



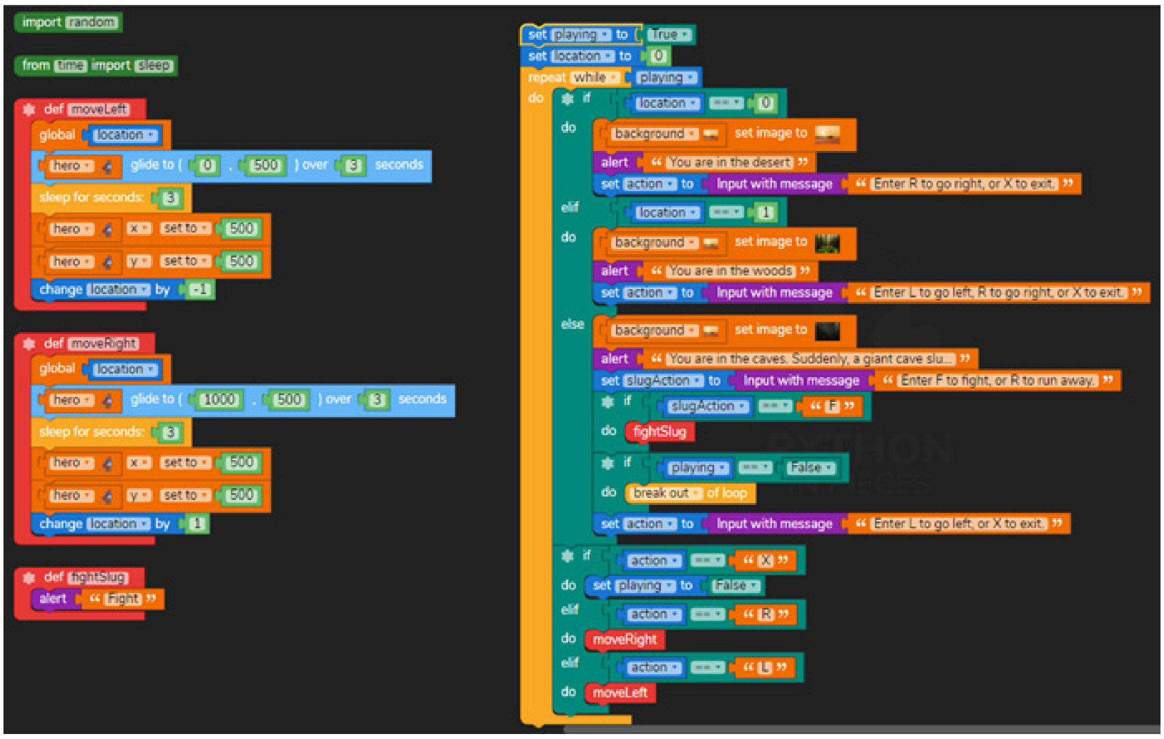
```
from time import sleep

import random

def moveLeft
  global location
  hero glide to ( 0 , 500 ) over 3 seconds
  sleep for seconds: 3
  hero x set to 500
  hero y set to 500
  change location by -1

def moveRight
  global location
  hero glide to ( 1000 , 500 ) over 3 seconds
  sleep for seconds: 3
  hero x set to 500
  hero y set to 500
  change location by 1

set playing to True
set location to 0
repeat while playing
do  if location == 0
    do  background set image to
        alert "You are in the desert"
        set action to Input with message "Enter R to go right, or X to exit."
    elif location == 1
    do  background set image to
        alert "You are in the woods"
        set action to Input with message "Enter L to go left, R to go right, or X to exit."
    else
        background set image to
        alert "You are in the caves"
        set action to Input with message "Enter L to go left, or X to exit."
    if action == "X"
    do  set playing to False
    elif action == "R"
    do  moveRight
    elif action == "L"
    do  moveLeft
```

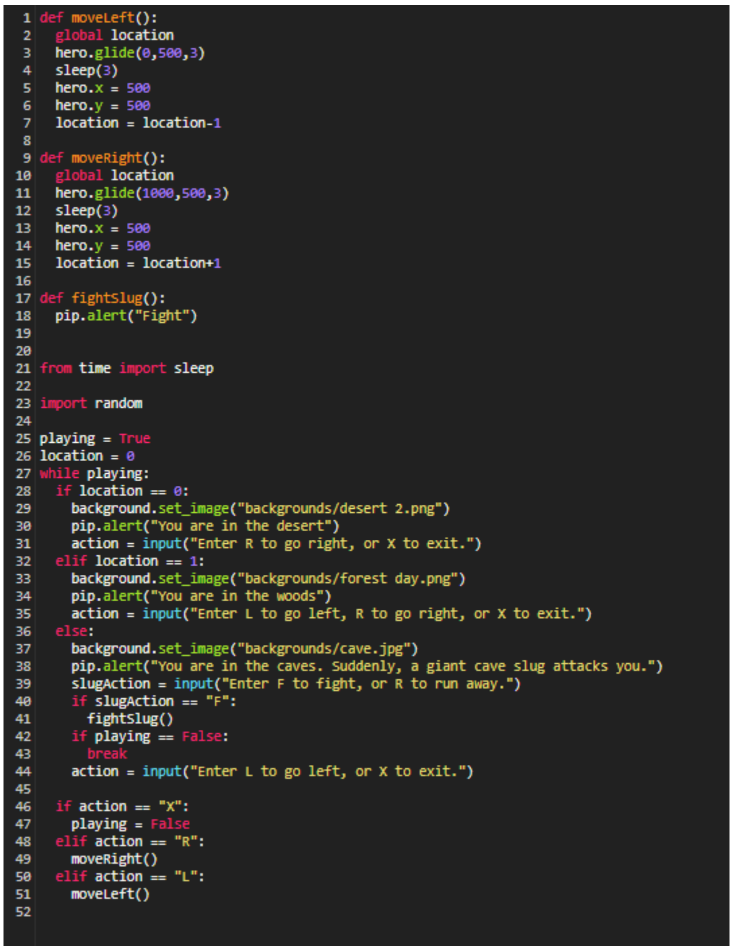
```
  global location
  hero.glide(0,500,3)
  sleep(3)
  hero.x = 500
  hero.y = 500
  location = location-1

def moveRight():
  global location
  hero.glide(1000,500,3)
  sleep(3)
  hero.x = 500
  hero.y = 500
  location = location+1


import random

from time import sleep

playing = True
location = 0
while playing:
  if location == 0:
    background.set_image("backgrounds/desert 2.png")
    pip.alert("You are in the desert")
    action = input("Enter R to go right, or X to exit.")
  elif location == 1:
    background.set_image("backgrounds/forest day.png")
    pip.alert("You are in the woods")
    action = input("Enter L to go left, R to go right, or X to exit.")
  else:
    background.set_image("backgrounds/cave.jpg")
    pip.alert("You are in the caves")
    action = input("Enter L to go left, or X to exit.")
  if action == "X":
    playing = False
  elif action == "R":
    moveRight()
  elif action == "L":
    moveLeft()
```

Stage 6- Nested logic

Different things can happen in our game depending on where the user is. For example, the caves are full of giant slugs. Nested ‘if’ statements are useful but sometimes it is better to separate out some logic into a function otherwise your code can get long and complex. Often, a combination of nested ‘if’ statements and functions is best.
Let’s make a function to handle the slug fighting.

Solution





```
def moveLeft():
  global location
  hero.glide(0,500,3)
  sleep(3)
  hero.x = 500
  hero.y = 500
  location = location-1

def moveRight():
  global location
  hero.glide(1000,500,3)
  sleep(3)
  hero.x = 500
  hero.y = 500
  location = location+1

def fightSlug():
  pip.alert("Fight")


from time import sleep

import random

playing = True
location = 0
while playing:
  if location == 0:
    background.set_image("backgrounds/desert 2.png")
    pip.alert("You are in the desert")
    action = input("Enter R to go right, or X to exit.")
  elif location == 1:
    background.set_image("backgrounds/forest day.png")
    pip.alert("You are in the woods")
    action = input("Enter L to go left, R to go right, or X to exit.")
  else:
    background.set_image("backgrounds/cave.jpg")
    pip.alert("You are in the caves. Suddenly, a giant cave slug attacks you.")
    slugAction = input("Enter F to fight, or R to run away.")
    if slugAction == "F":
      fightSlug()
    if playing == False:
      break
    action = input("Enter L to go left, or X to exit.")

  if action == "X":
    playing = False
  elif action == "R":
    moveRight()
  elif action == "L":
    moveLeft()

```

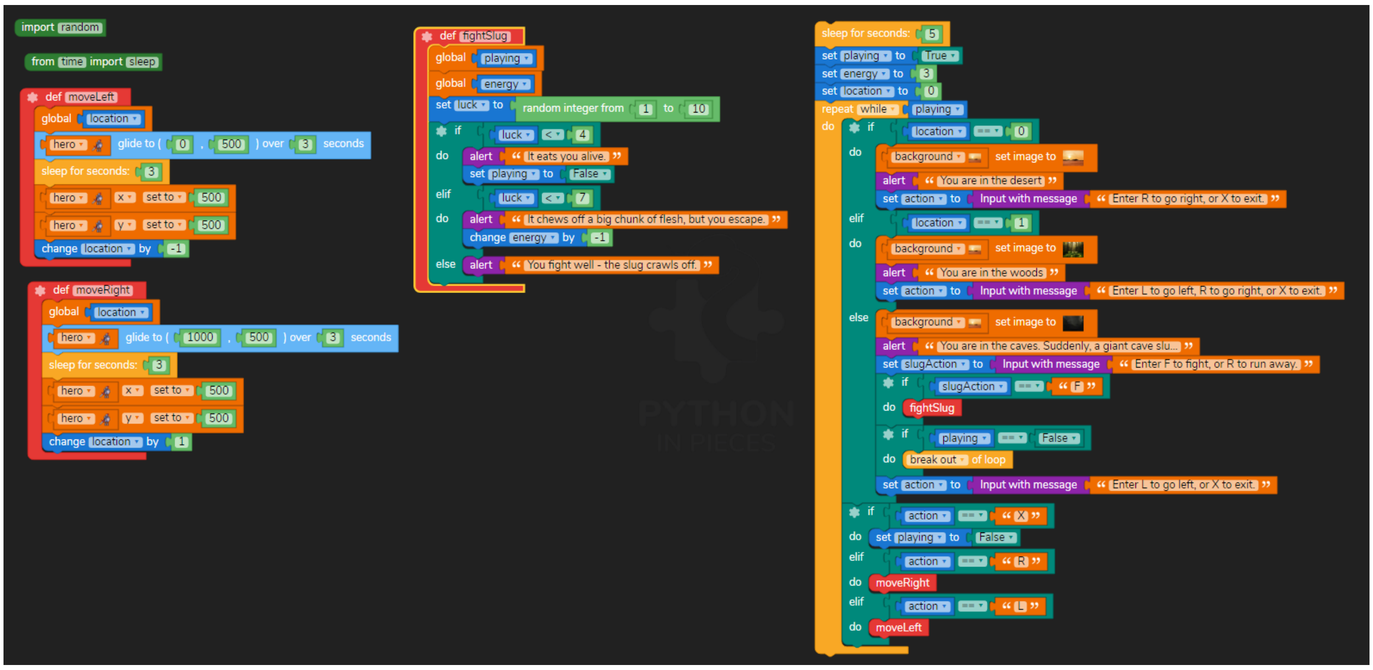
Stage 7- Fight the slug

Let’s fill in the details of the slug fighting function. We will use a variable called ‘energy’ and each time you lose a fight your energy should decrease.

Tasks

- After setting ‘location = 0’, make another variable called ‘energy’ and initialize it to 3.
- Inside the ‘fightSlug’ function, delete the alert, make a variable called ‘luck’ and assign it a random value between 1 and 10. Use ‘random.randint’.”
- Use an ‘if’ statement to check if luck is less than 4. If so, alert the text ‘It eats you alive’ and set ‘playing’ equal to False. Make sure you declare ‘playing’ as global first.
- Add an ‘elif’ statement to check if luck is less than 7. If so, alert the text ‘It chews off a big chunk of flesh, but you escape’ and decrease ‘energy’ by 1. Make sure you declare ‘energy’ as global first.
- Add an ‘else’ statement which alerts the text ‘You fight well - the slug crawls off’.
- Test your game by fighting the slug a few times. Check that the value of ‘energy’ shown in the debug panel is correct.

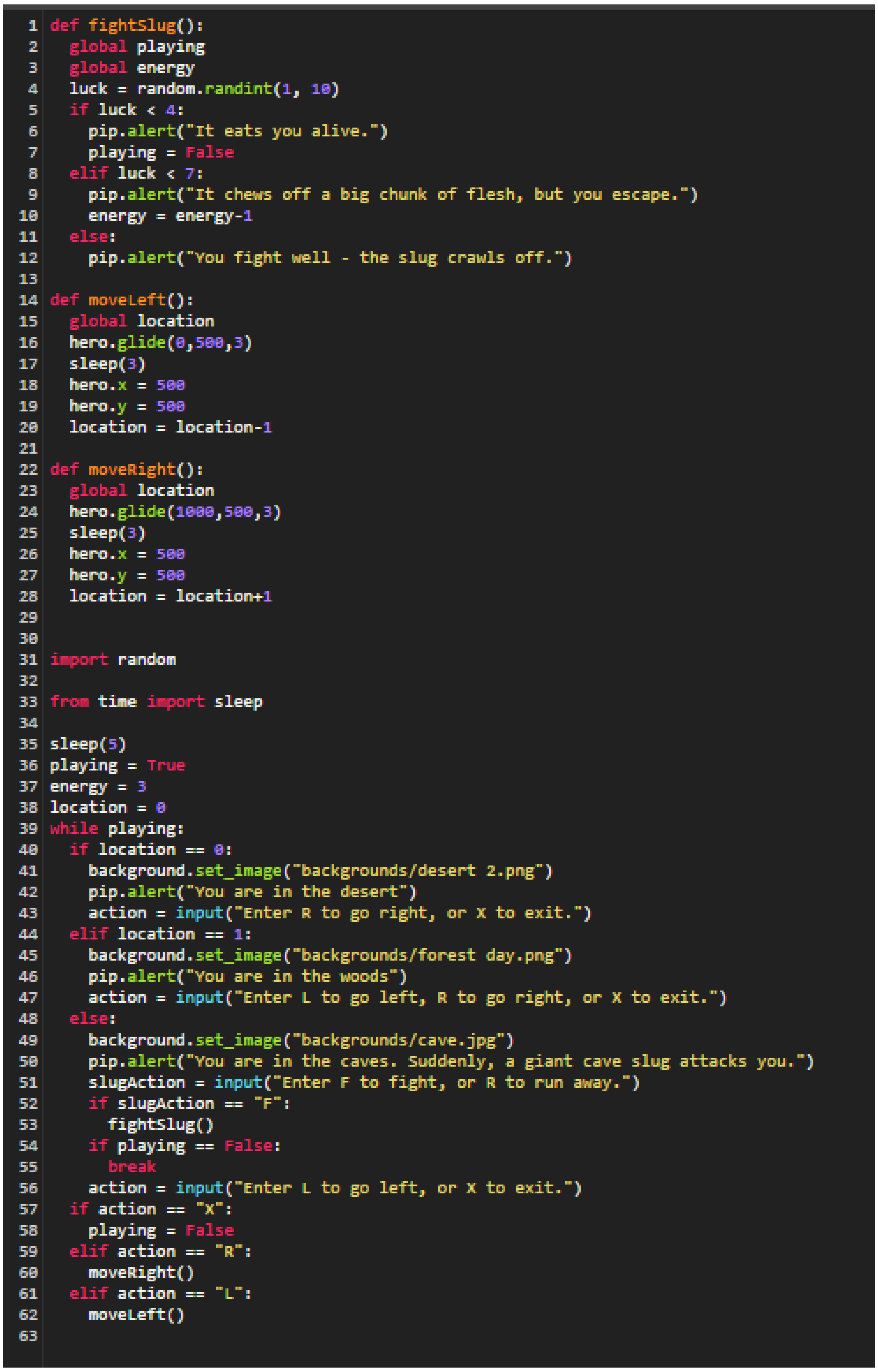
Solution



```
import random

from time import sleep

def moveLeft
    global location
    hero glide to ( 0 , 500 ) over 3 seconds
    sleep for seconds: 3
    hero x set to 500
    hero y set to 500
    change location by -1

def moveRight
    global location
    hero glide to ( 1000 , 500 ) over 3 seconds
    sleep for seconds: 3
    hero x set to 500
    hero y set to 500
    change location by 1

def fightSlug
    global playing
    global energy
    set luck to random integer from 1 to 10
    if luck < 4
    do  alert “It eats you alive.”
        set playing to False
    elif luck < 7
    do  alert “It chews off a big chunk of flesh, but you escape.”
        change energy by -1
    else alert “You fight well - the slug crawls off.”

sleep for seconds: 5
set playing to True
set energy to 3
set location to 0
repeat while playing
do  if location == 0
    do  background set image to
        alert “You are in the desert”
        set action to Input with message “Enter R to go right, or X to exit.”
    elif location == 1
    do  background set image to
        alert “You are in the woods”
        set action to Input with message “Enter L to go left, R to go right, or X to exit.”
    else
        background set image to
        alert “You are in the caves. Suddenly, a giant cave slu...”
        set slugAction to Input with message “Enter F to fight, or R to run away.”
        if slugAction == “F”
        do  fightSlug
        if playing == False
        do  break out of loop
        set action to Input with message “Enter L to go left, or X to exit.”
    if action == “X”
    do  set playing to False
    elif action == “R”
    do  moveRight
    elif action == “L”
    do  moveLeft
```

Solution continued on following page...

Solution



```python
def fightSlug():
  global playing
  global energy
  luck = random.randint(1, 10)
  if luck < 4:
    pip.alert("It eats you alive.")
    playing = False
  elif luck < 7:
    pip.alert("It chews off a big chunk of flesh, but you escape.")
    energy = energy-1
  else:
    pip.alert("You fight well - the slug crawls off.")

def moveLeft():
  global location
  hero.glide(0,500,3)
  sleep(3)
  hero.x = 500
  hero.y = 500
  location = location-1

def moveRight():
  global location
  hero.glide(1000,500,3)
  sleep(3)
  hero.x = 500
  hero.y = 500
  location = location+1


import random

from time import sleep

sleep(5)
playing = True
energy = 3
location = 0
while playing:
  if location == 0:
    background.set_image("backgrounds/desert 2.png")
    pip.alert("You are in the desert")
    action = input("Enter R to go right, or X to exit.")
  elif location == 1:
    background.set_image("backgrounds/forest day.png")
    pip.alert("You are in the woods")
    action = input("Enter L to go left, R to go right, or X to exit.")
  else:
    background.set_image("backgrounds/cave.jpg")
    pip.alert("You are in the caves. Suddenly, a giant cave slug attacks you.")
    slugAction = input("Enter F to fight, or R to run away.")
    if slugAction == "F":
      fightSlug()
    if playing == False:
      break
    action = input("Enter L to go left, or X to exit.")
  if action == "X":
    playing = False
  elif action == "R":
    moveRight()
  elif action == "L":
    moveLeft()

```

Challenge

Try out the following challenges.

Suggestions

- Check when the user's energy drops to zero and if so set playing = False and make sure the game ends.
- Tell the user their energy level when they reach a new location and tell them when the game is over.
- At the moment there is nothing stopping you going left from the desert, or right from the cave. location will become -1 or 3. Add code to stop this.
- Add another location - use your imagination.
- Make it so that the hero can move in four directions not just two.
- Add some different events that can happen in the different locations. This could be other monsters that you can fight or anything else that involves 'if' and 'elif' statements.
- Add a monster that asks a riddle - if the user gets it right they can escape.
- Add collectable objects which help you fight the slug, or that boost your energy level.

Lesson 3 – Password Generator

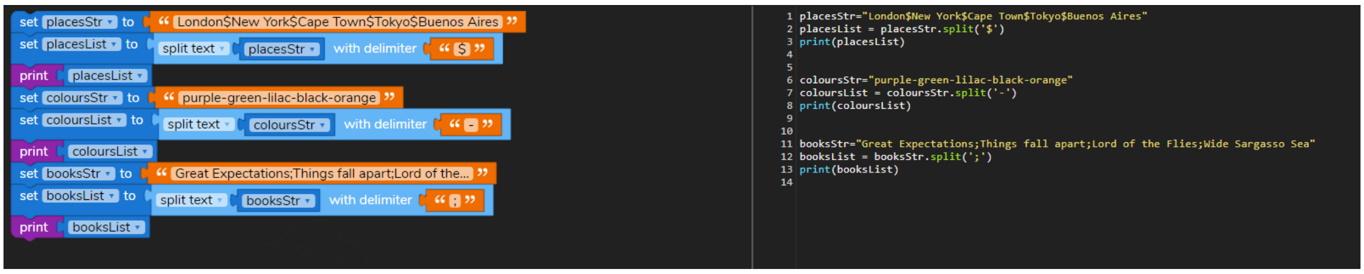
Stage 1- The ‘split’ method

In Python, you can split any string using the ‘split’ method. You need to tell the method what ‘separator’ to use. For example, here is a string consisting of four names separated by the ‘&’ symbol.

Tasks

- Make a list called ‘namesList’ which contains the four names in the string ‘namesStr’. Choose the right delimiter to use. Print ‘namesList’
- Make a list called ‘placesList’ which contains the four places in the string ‘placesStr’. Choose the right delimiter to use. Print ‘placesList’
- Make a list called ‘coloursList’ which contains the four colours in the string ‘coloursStr’. Choose the right delimiter to use. Print ‘coloursList’
- Make a list called ‘booksList’ which contains the four books in the string ‘booksStr’. Choose the right delimiter to use. Print ‘booksList’

Solution



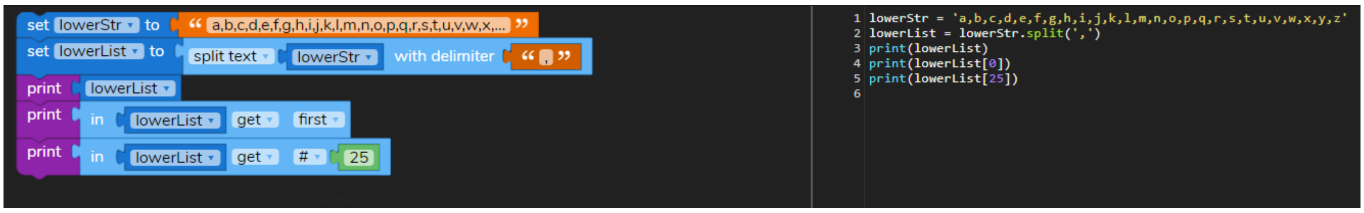
```
set placesStr to "London$New York$Cape Town$Tokyo$Buenos Aires"
set placesList to split text placesStr with delimiter "$"
print placesList
set coloursStr to "purple-green-lilac-black-orange"
set coloursList to split text coloursStr with delimiter "-"
print coloursList
set booksStr to "Great Expectations;Things fall apart;Lord of the..."
set booksList to split text booksStr with delimiter ";"
print booksList
```

```python
placesStr="London$New York$Cape Town$Tokyo$Buenos Aires"
placesList = placesStr.split('$')
print(placesList)


coloursStr="purple-green-lilac-black-orange"
coloursList = coloursStr.split('-')
print(coloursList)


booksStr="Great Expectations;Things fall apart;Lord of the Flies;Wide Sargasso Sea"
booksList = booksStr.split(';')
print(booksList)
```

Stage 2- Splitting the alphabet

Now we are ready to start our password generator.
The code for this stage contains a string variable called ‘lowerStr’, equal to the lower case letters of the alphabet, separated by commas.
lowerStr = “a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z”
We need to make a list from this so that we can easily choose random letters for our password. Our delimiter is a comma.

Solution



```
set lowerStr to "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,..."
set lowerList to split text lowerStr with delimiter ","
print lowerList
print in lowerList get first
print in lowerList get # 25
```

```python
lowerStr = 'a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z'
lowerList = lowerStr.split(',')
print(lowerList)
print(lowerList[0])
print(lowerList[25])
```

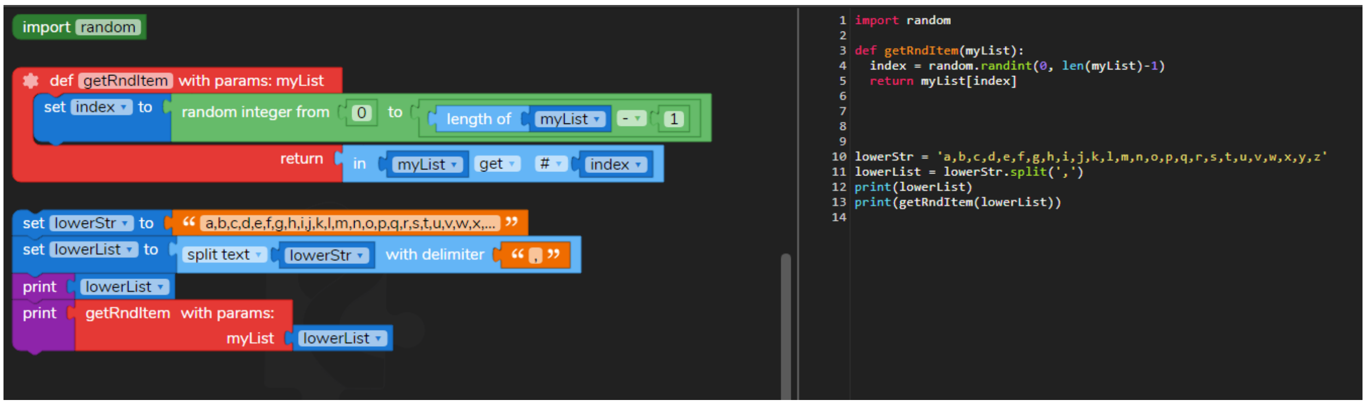
Stage 3- Choosing a random letter

To pick a random item from a list of 26 items we need to choose a random index between 0 and 25.In general, to pick a random item from a list we need to choose a random index between 0 and the length minus 1.

Tasks

- Import the 'random' module. This should go at the top of your program.
- Create a function called 'getRndItem'. This function should take one parameter called 'myList'. This function will return a random item from whatever list you pass in to it.
- Inside the function, create a variable called 'index' and use 'random.randint' to assign it a random value between 0 and the length of 'myList' minus one.
- Make the function return the item at that index in myList.
- Check it works. At the end of your program, print(getRndItem(lowerList)) and run your program a few times. Check that it prints random letters.

Solution



```
import random

def getRndItem(myList):
  index = random.randint(0, len(myList)-1)
  return myList[index]




lowerStr = 'a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z'
lowerList = lowerStr.split(',')
print(lowerList)
print(getRndItem(lowerList))
```

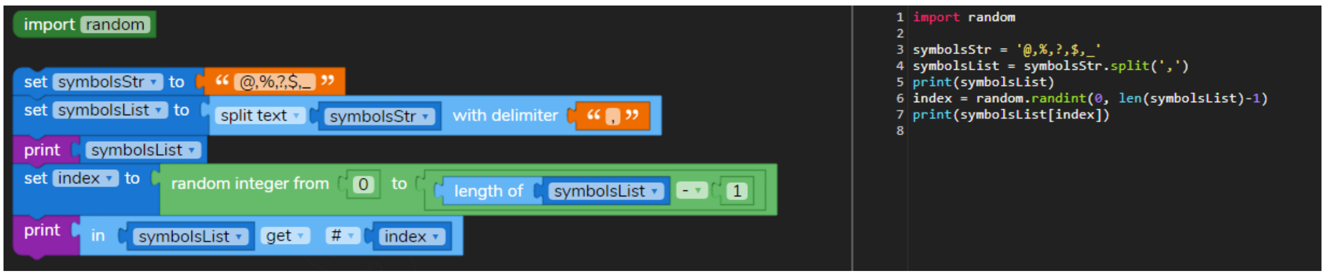
Stage 4- Adding symbols - debug

Someone has been trying to get a random symbol from a string of symbols, but it's not working. Try to run the code. It doesn't work at the moment, it just outputs all five symbols all the time. Let's fix it.

Tasks

- There is something wrong with the definition of 'symbolsStr'. Fix this first. Hint - the string is split using comma as the delimiter.
- There are five symbols, so we can pick from 0 to 4. At the moment if you keep running the program it will eventually fail with an error 'list index out of range'. Why? Fix this by making sure it selects an index between 0 and 4.
- Run your program a few times and check the 'index' variable is always between 0 and 4 and that it prints random symbols.

Solution



```
import random

symbolsStr = '@,%,?,$,_'
symbolsList = symbolsStr.split(',')
print(symbolsList)
index = random.randint(0, len(symbolsList)-1)
print(symbolsList[index])
```

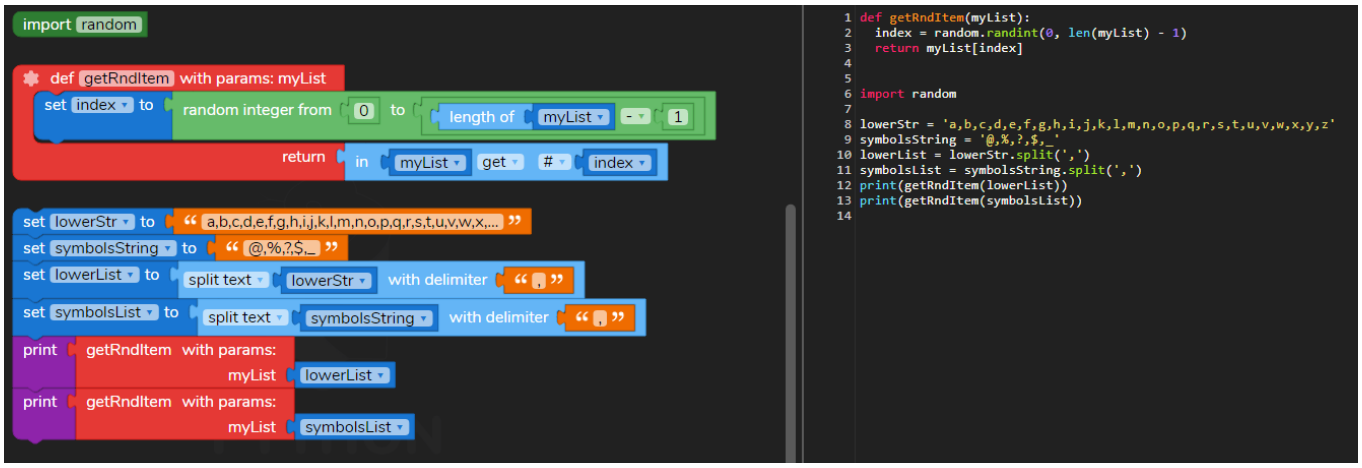
Stage 5- Adding symbols

Let's add symbols to our program and use our function to get a random symbol.

Tasks

- At the top of your program, after defining 'lowerStr', define another string called 'symbolsStr' and assign it the value "@,%,?,$,_'. Make a list called 'symbolsList', split 'symbolsStr' using comma as the delimiter and assign it to 'symbolsList'.
- Use your function 'getRndItem' to print a random symbol. Check that each time you run the program you get a random symbol.

Solution


import random
def getRndItem with params: myList
set index to random integer from 0 to length of myList - 1
return in myList get # index
set lowerStr to "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,..."
set symbolsString to "@,%,?,$,_"
set lowerList to split text lowerStr with delimiter ","
set symbolsList to split text symbolsString with delimiter ","
print getRndItem with params: myList lowerList
print getRndItem with params: myList symbolsList

```python
def getRndItem(myList):
  index = random.randint(0, len(myList) - 1)
  return myList[index]


import random

lowerStr = 'a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z'
symbolsString = '@,%,?,$,_'
lowerList = lowerStr.split(',')
symbolsList = symbolsString.split(',')
print(getRndItem(lowerList))
print(getRndItem(symbolsList))
```

Stage 6- 4 digit password

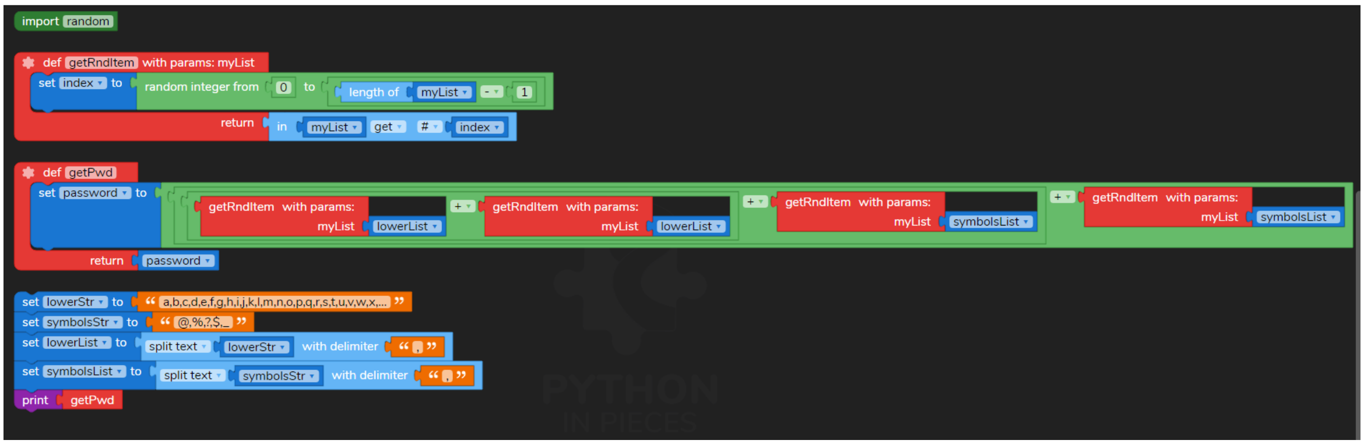
Let's make a 4 digit random password, consisting of two lowercase letters followed by two symbols.

Tasks

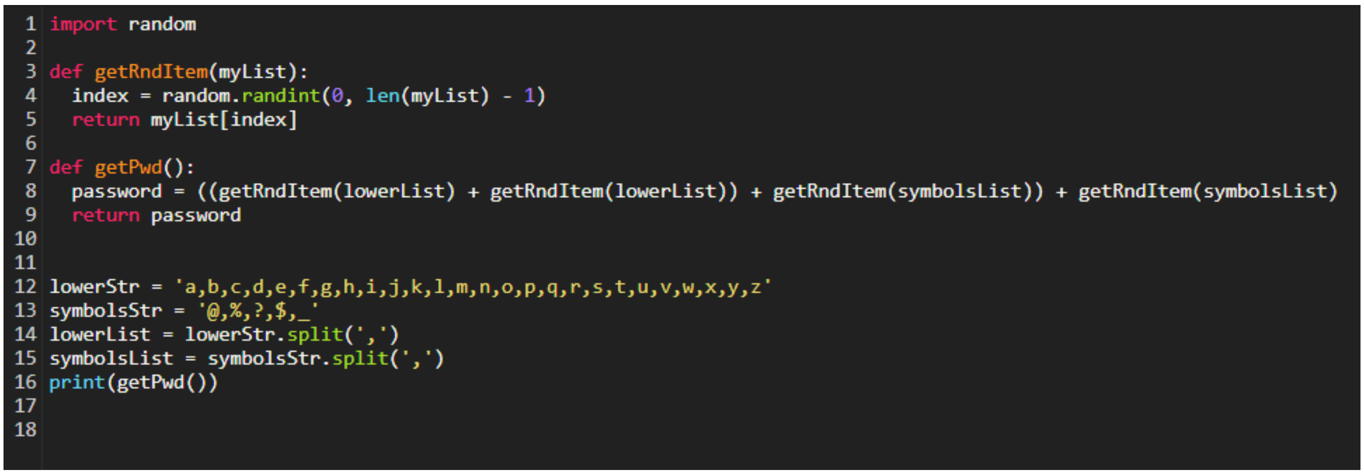
- Make a function called 'getPwd'. It should take no input parameters.
- Inside, create a variable called 'password'. Use the '+' operator to join together two random letters followed by two random symbols. Return 'password'.
- At the bottom of your program print(getPwd()) and check that it prints a random four-character password. Run it a few times to check.

Solutions on following page...

Solution



```
import random

def getRndItem with params: myList
  set index to random integer from 0 to length of myList - 1
  return in myList get # index

def getPwd
  set password to getRndItem with params: myList lowerList + getRndItem with params: myList lowerList + getRndItem with params: myList symbolsList + getRndItem with params: myList symbolsList
  return password

set lowerStr to “a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,...”
set symbolsStr to “@,%,?,$,_”
set lowerList to split text lowerStr with delimiter “,”
set symbolsList to split text symbolsStr with delimiter “,”
print getPwd
```



```
import random

def getRndItem(myList):
  index = random.randint(0, len(myList) - 1)
  return myList[index]

def getPwd():
  password = ((getRndItem(lowerList) + getRndItem(lowerList)) + getRndItem(symbolsList)) + getRndItem(symbolsList)
  return password


lowerStr = 'a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z'
symbolsStr = '@,%,?,$,_'
lowerList = lowerStr.split(',')
symbolsList = symbolsStr.split(',')
print(getPwd())
```

Challenge

Try out the following challenges.

Suggestions

- Create a counter that stores how many times the password has been checked so far. Display it on the screen.
- Change the code so that it creates passwords which also contain some random numbers or uppercase letters, or both. These will be even harder to break.
- Make a new function that will output a list of ‘n’ random passwords (where ‘n’ is a number you pass into the function).
- Make a new function into which you can pass four numbers n_low, n_upp, n_sym, n_dig. Your function should create a random password consisting of ‘n_low’ lowercase letters, ‘n_upp’ uppercase letters, ‘n_sym’ symbols and ‘n_dig’ digits 0-9.
- Add a ‘Start’ button to the stage. The password hacking should start when you click the button. Add a ‘Stop’ button to stop the hacking.

Lesson 4 – Binary and Decimal

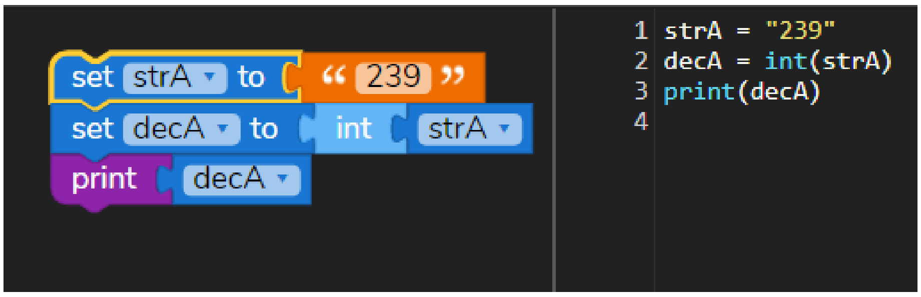
Stage 1- Recap – decimals and ‘int’

So far all the numbers we have used have been in ‘base 10’. Numbers in base 10 are also called ‘decimal’ or ‘denary’ numbers. The place values are the powers of 10. Reading right to left they are 1, 10, 100, 1000... Each one is allowed to hold a digit fom 0 to 9.
Reading from right to left, we can work out what 239 means. It means 9 lots of 1, 3 lots of 10 and 2 lots of 100.
The ‘int’ function converts from a string to a decimal number using base 10

Tasks

- ‘strA’ is a string. Convert it to a number called ‘decA’ and print ‘decA’. Use the ‘int’ function.

Solution



```
strA = "239"
decA = int(strA)
print(decA)
```

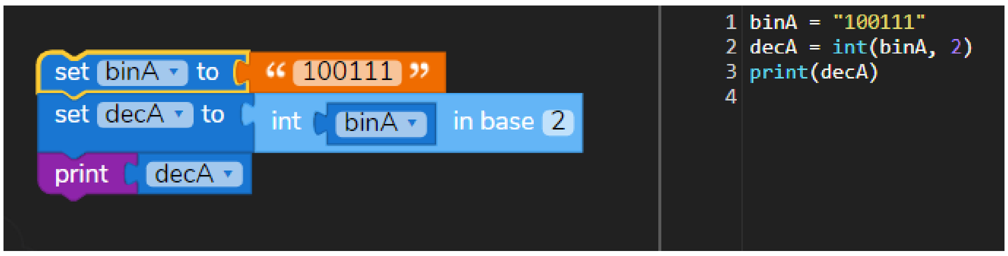
Stage 2- What is a binary number?

Numbers do not have to be represented in base 10. Computer hardware stores numbers in base 2 (binary). The place values are the powers of 2 and each is allowed to hold a digit 0 or 1. For example, the number 100111 should be thought of like this:
Reading from right to left, it consists of 1 lot of 1, 1 lot of 2, 1 lot of 4, 0 lots of 8, 0 lots of 16 and 1 lot of 32. Add them up and you get 39. This is the decimal value of the binary string “100111”.
In Python you can use the ‘int’ function with an extra parameter at the end to indicate what base to use.

Tasks

- ‘binA’ is a binary string. Convert it to a decimal called ‘decA’. Use the ‘int in base 2’ block or the ‘int(binA, 2)’ function. Print ‘decA’.

Solution



```
binA = "100111"
decA = int(binA, 2)
print(decA)
```

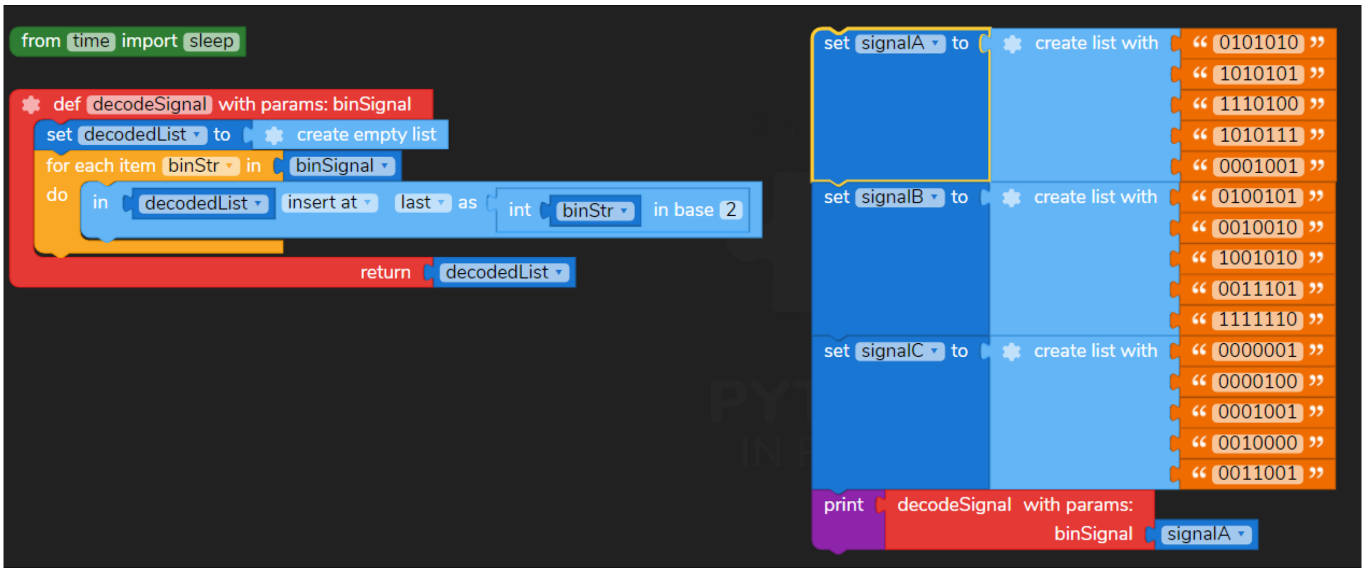
Stage 3- Multiple binary numbers

If we have a list of binary numbers, we can loop through the list and convert each one to a decimal.
Then we can return a new list containing the decimal numbers.

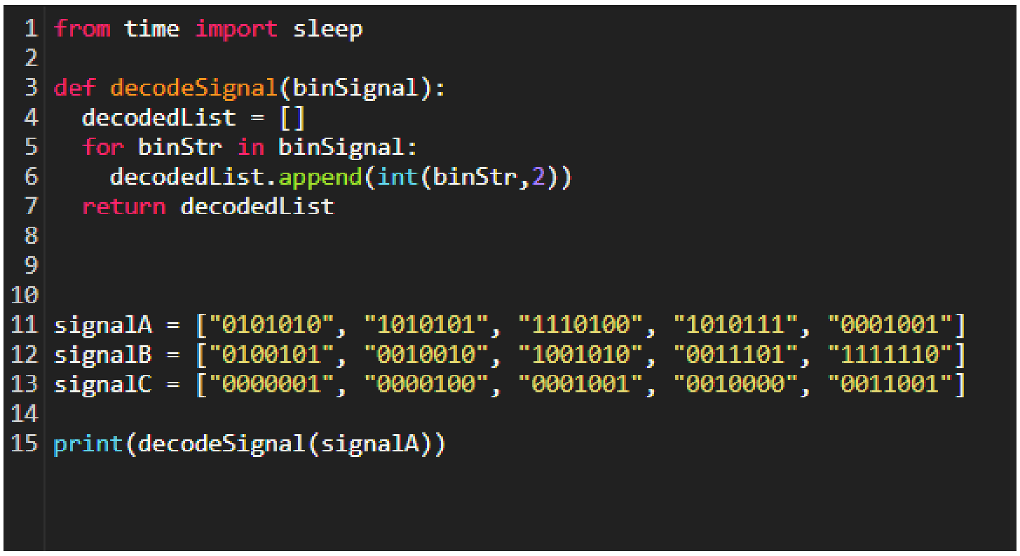
Tasks

- Make a function called 'decodeSignal'. It should take one parameter, called 'binSignal'. This will be a list of binary strings.
- Inside the function make an empty list called 'decodedList'.
- Use a 'for binStr in binSignal' loop to loop through each of the strings in the list 'binSignal'.
- Inside the loop, convert 'binStr' to a decimal number called 'dec' using the 'int' function in base 2.
- Append each decimal number to 'decodedList'
- At the end of your function, return 'decodedList'.
- Test your function by printing 'decodeSignal(signalA) - check that it prints a list of decimal numbers'

Solution



```
from time import sleep

def decodeSignal with params: binSignal
  set decodedList to create empty list
  for each item binStr in binSignal
  do  in decodedList insert at last as int binStr in base 2
  return decodedList

set signalA to create list with "0101010" "1010101" "1110100" "1010111" "0001001"
set signalB to create list with "0100101" "0010010" "1001010" "0011101" "1111110"
set signalC to create list with "0000001" "0000100" "0001001" "0010000" "0011001"
print decodeSignal with params: binSignal signalA
```



```
from time import sleep

def decodeSignal(binSignal):
  decodedList = []
  for binStr in binSignal:
    decodedList.append(int(binStr,2))
  return decodedList



signalA = ["0101010", "1010101", "1110100", "1010111", "0001001"]
signalB = ["0100101", "0010010", "1001010", "0011101", "1111110"]
signalC = ["0000001", "0000100", "0001001", "0010000", "0011001"]

print(decodeSignal(signalA))
```

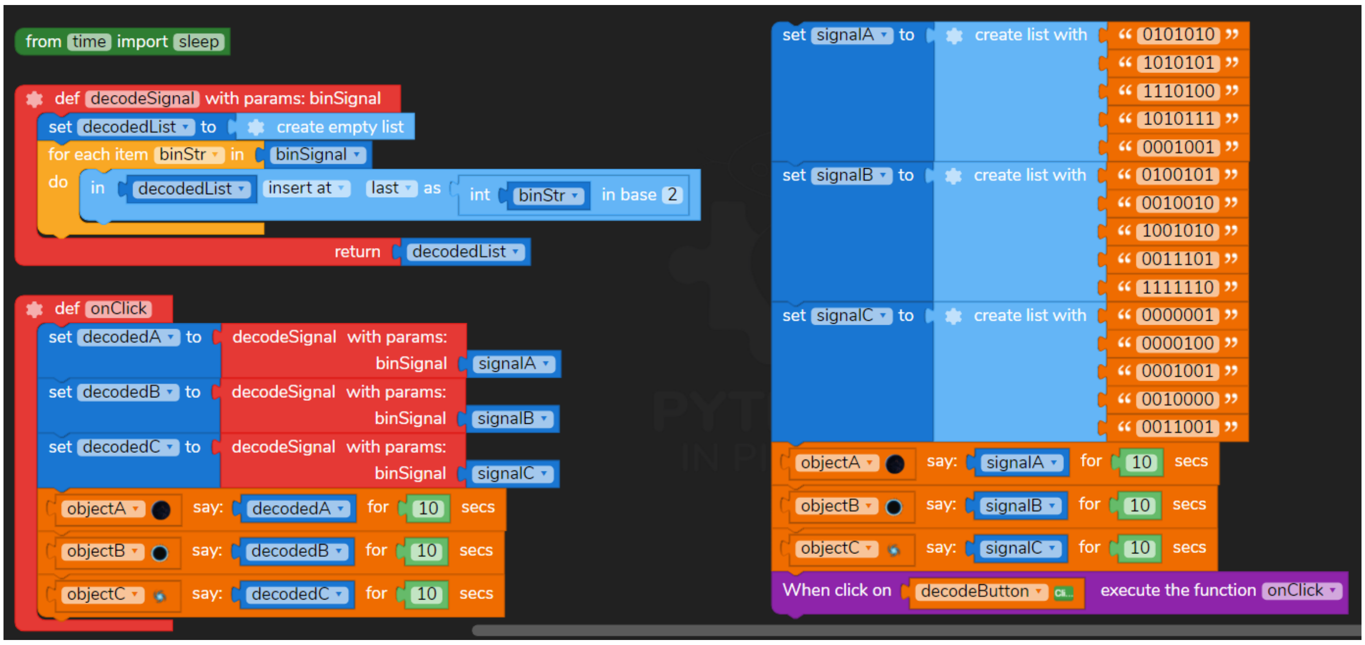
Stage 4- Alien messages

In 1974 a binary message called the 'Arecibo message' was sent from Earth towards a star cluster 21000 light years away, in an attempt to communicate with extra-terrestrials. It consisted of 0's and 1's which generate a picture of certain numbers, scientific formulae as well as a picture of a human and the Solar System.
What if they replied? They might reply with another number sequence to let us know they are intelligent. But space is full of random radio signals from all kinds of interstellar objects.

Tasks

- In design mode, find the three interstellar objects called 'objectA', 'objectB' and 'objectC'. Use 'objectA.say(signalA, 10)' to make objectA say signalA, and similar for objectB and objectC.
- Make a function called 'onClick'. This function needs no parameters.
- Inside, create a variable 'decodedA = decodeSignal(signalA)' and similar for signalB and signalC.
- Use the '.say' method to make each object show the decoded versions of each signal.
- Add a 'when click' listener to the button called 'decodeButton'. When the user clicks the button it should execute the function 'onClick'. To do this you can use the 'When click on' block or write 'pip.eventmanager.when_click(decodeButton, onClick)'.
- Run your program. Two of the number sequences look random, but one doesn't and could be from intelligent life - which one is it?

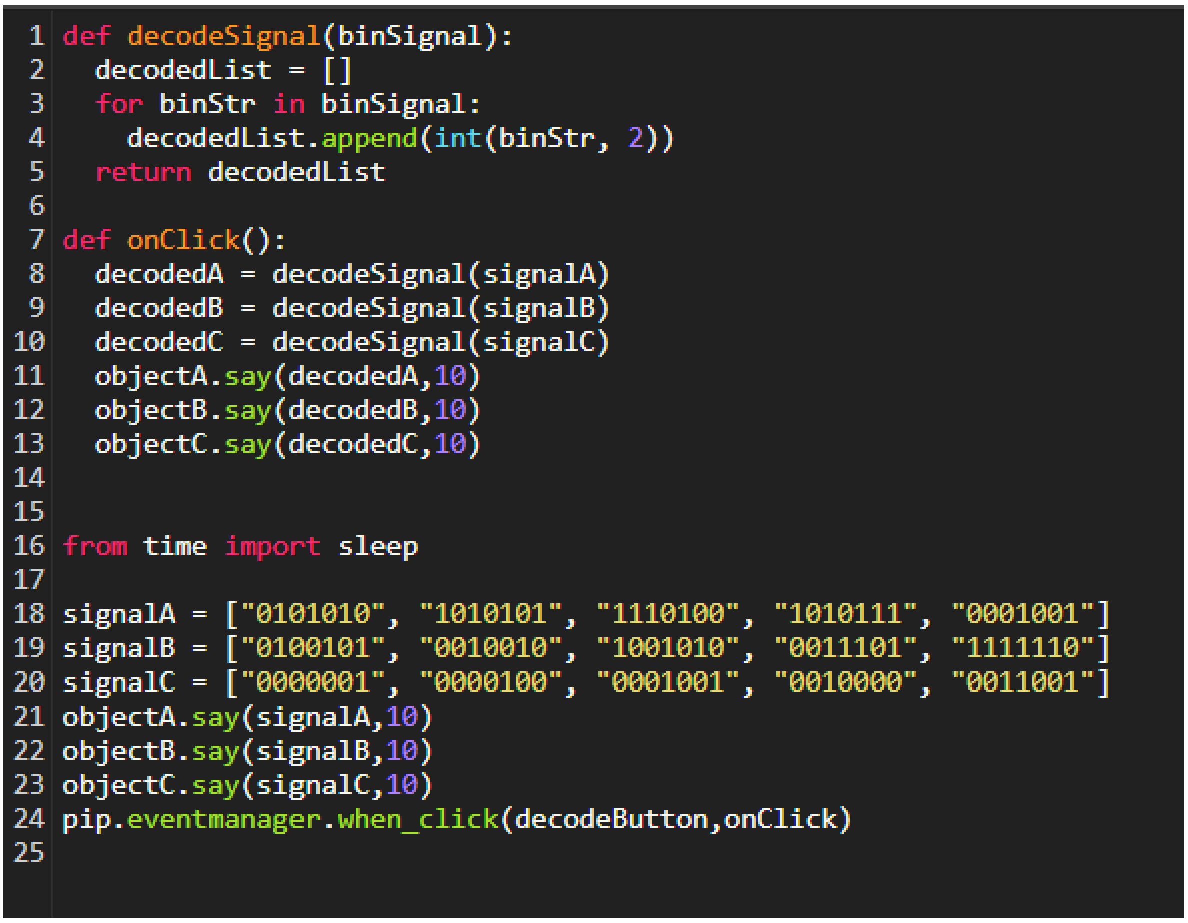
Solution



```
from time import sleep

def decodeSignal with params: binSignal
    set decodedList to create empty list
    for each item binStr in binSignal
    do  in decodedList insert at last as int binStr in base 2
    return decodedList

def onClick
    set decodedA to decodeSignal with params: binSignal signalA
    set decodedB to decodeSignal with params: binSignal signalB
    set decodedC to decodeSignal with params: binSignal signalC
    objectA say: decodedA for 10 secs
    objectB say: decodedB for 10 secs
    objectC say: decodedC for 10 secs

set signalA to create list with "0101010" "1010101" "1110100" "1010111" "0001001"
set signalB to create list with "0100101" "0010010" "1001010" "0011101" "1111110"
set signalC to create list with "0000001" "0000100" "0001001" "0010000" "0011001"
objectA say: signalA for 10 secs
objectB say: signalB for 10 secs
objectC say: signalC for 10 secs
When click on decodeButton execute the function onClick
```

Solutions continued on following page...

Solution



```python
def decodeSignal(binSignal):
  decodedList = []
  for binStr in binSignal:
    decodedList.append(int(binStr, 2))
  return decodedList

def onClick():
  decodedA = decodeSignal(signalA)
  decodedB = decodeSignal(signalB)
  decodedC = decodeSignal(signalC)
  objectA.say(decodedA,10)
  objectB.say(decodedB,10)
  objectC.say(decodedC,10)


from time import sleep

signalA = ["0101010", "1010101", "1110100", "1010111", "0001001"]
signalB = ["0100101", "0010010", "1001010", "0011101", "1111110"]
signalC = ["0000001", "0000100", "0001001", "0010000", "0011001"]
objectA.say(signalA,10)
objectB.say(signalB,10)
objectC.say(signalC,10)
pip.eventmanager.when_click(decodeButton,onClick)

```

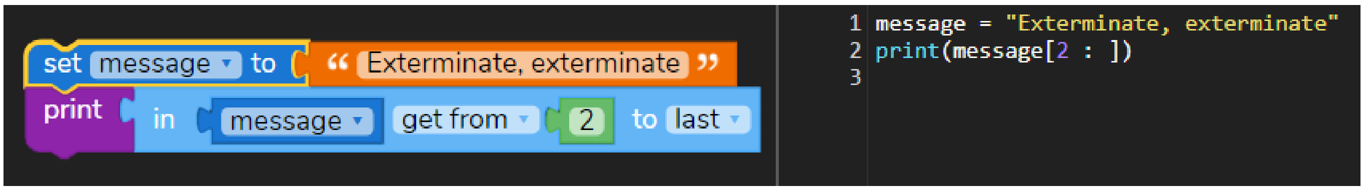
Stage5- Substrings

To access a character in a string you can use square brackets. The index you use starts at zero. For example the first letter ‘k’ in the string “Take me to your leader” is at index 2.
To access all of the characters from index 2 onwards you can use ‘string slicing’:

Tasks

- Print the string “terminate, exterminate” using string slicing. What is the correct index?

Solution


set message to “Exterminate, exterminate”
print in message get from 2 to last

```python
message = "Exterminate, exterminate"
print(message[2 : ])

```

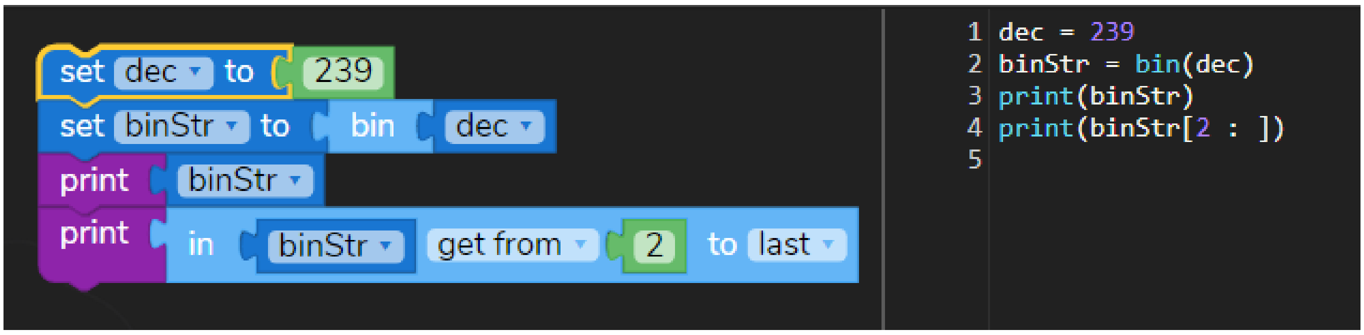
Stage 6- Decimal to binary

There is a function called ‘bin’ in Python to convert from decimal to a binary string.
Python always puts the two characters ‘0b’ at the front to tell you that it is in binary. We can remove them using string slicing to get the string that we want, containing just 0’s and 1’s

Tasks

- Use the ‘bin’ function to convert ‘dec’ to a string called ‘binStr’. Print the value of ‘binStr’. Check that you see the ‘0b’ characters at the start
- Use string slicing to print ‘binStr’ without the ‘0b’ characters.

Solution



```
dec = 239
binStr = bin(dec)
print(binStr)
print(binStr[2 : ])
```

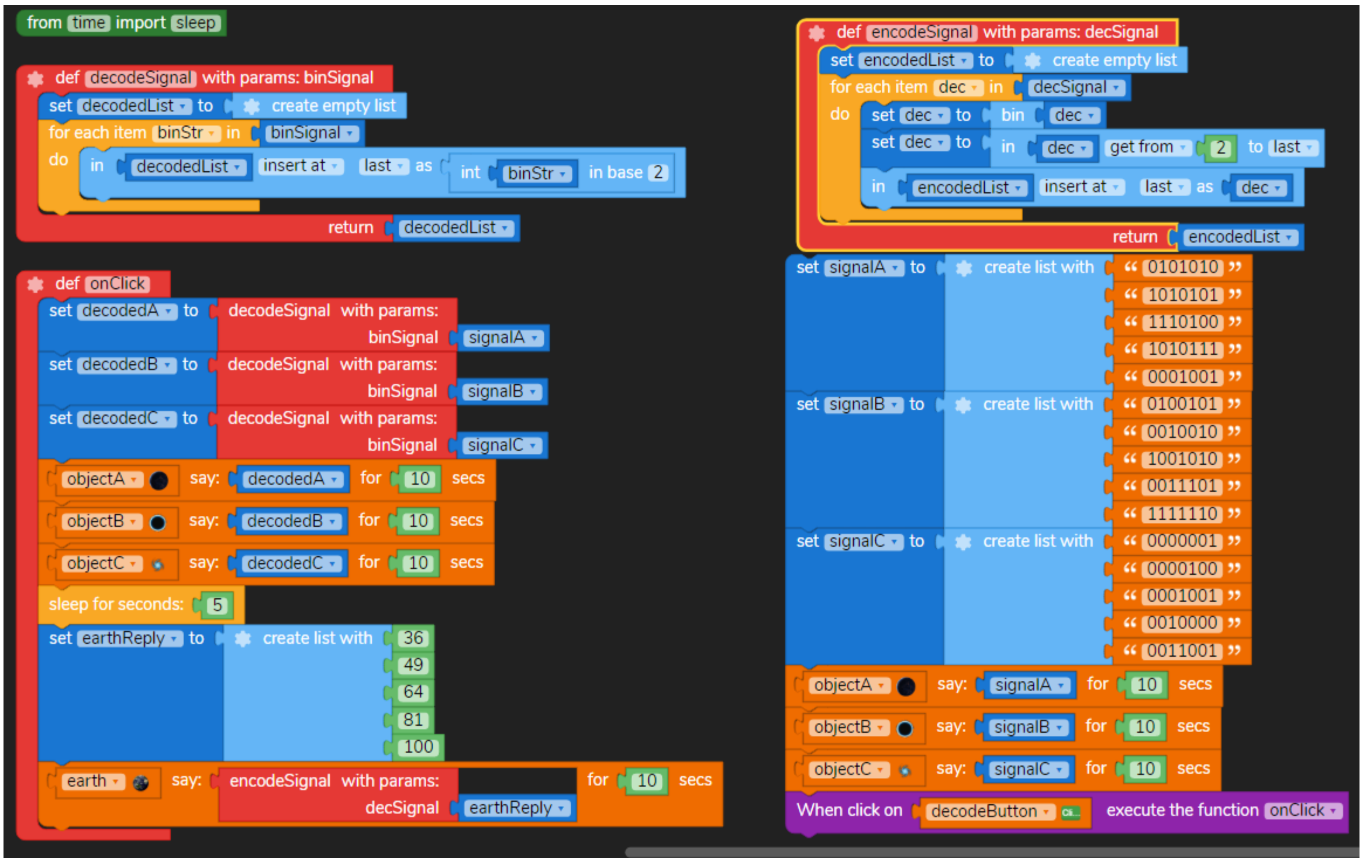
Stage 7- Conversion

We can use the ‘bin’ function to make a function that encodes a signal to be sent back to the aliens.

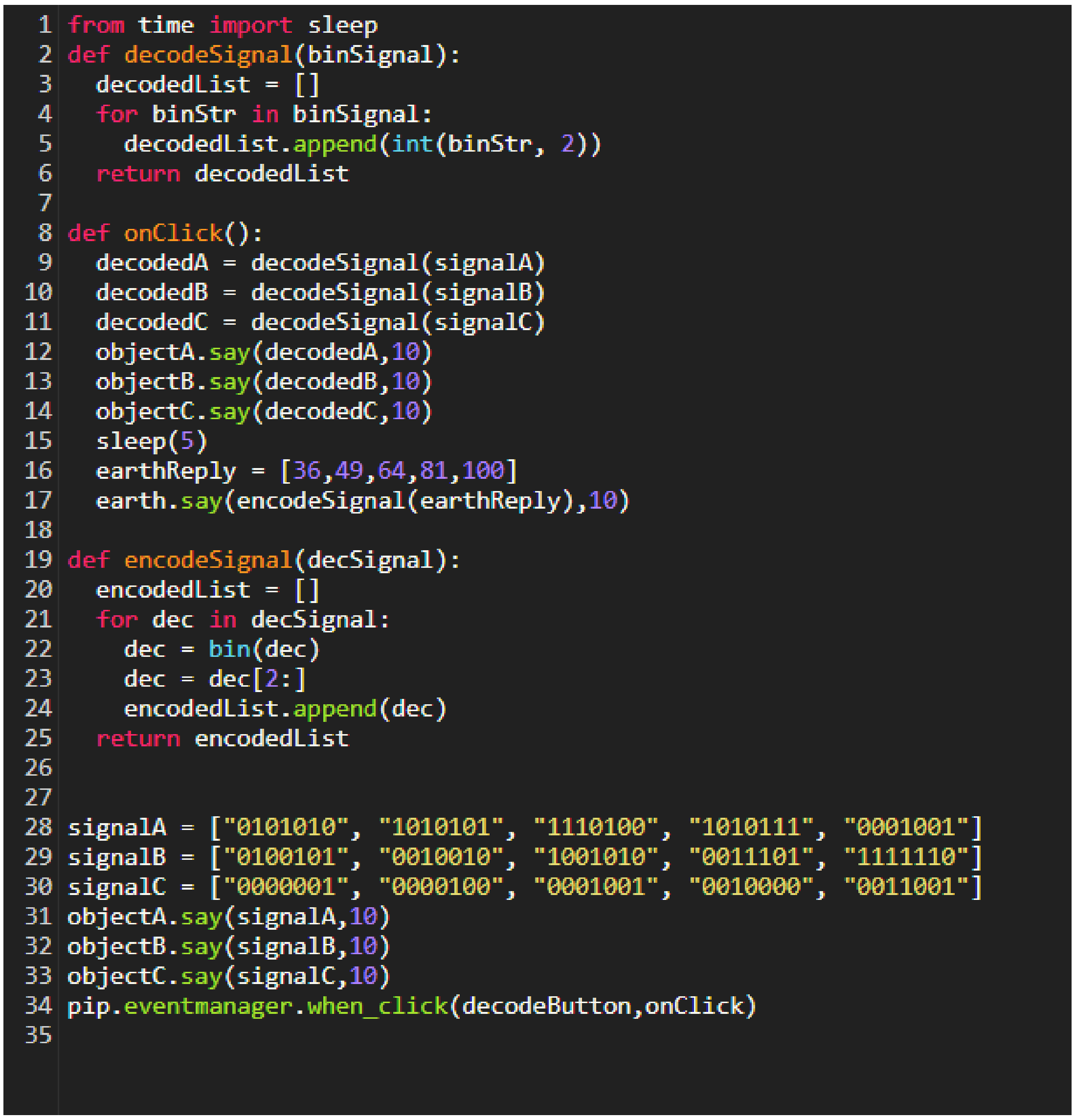
Tasks

- Make a function called encodeSignal. It should take one parameter called ‘decSignal’
- Inside, create an empty list called ‘encodedList’
- Use a ‘for dec in decSignal’ loop to loop through each of the decimal numbers in the list ‘decSignal’.
- Inside the loop, convert ‘dec’ to a binary string without the ‘0b’ characters. Use the ‘bin’ function and string slicing.
- Append each string to ‘encodedList’
- At the end of your function, return ‘encodedList’.
- Test your function by sending a reply from Earth to the alien galaxy. What number pattern did the aliens use? At the end of the ‘onClick’ function, add a ‘sleep(5)’ command. Make a list that contains the next five numbers in the sequence, encode it in binary using the ‘encodeSignal’ function and make ‘earth’ say it.

Solutions on following page...



```
from time import sleep

def decodeSignal with params: binSignal
  set decodedList to create empty list
  for each item binStr in binSignal
  do  in decodedList insert at last as int binStr in base 2
  return decodedList

def onClick
  set decodedA to decodeSignal with params: binSignal signalA
  set decodedB to decodeSignal with params: binSignal signalB
  set decodedC to decodeSignal with params: binSignal signalC
  objectA say: decodedA for 10 secs
  objectB say: decodedB for 10 secs
  objectC say: decodedC for 10 secs
  sleep for seconds: 5
  set earthReply to create list with 36 49 64 81 100
  earth say: encodeSignal with params: decSignal earthReply for 10 secs

def encodeSignal with params: decSignal
  set encodedList to create empty list
  for each item dec in decSignal
  do  set dec to bin dec
      set dec to in dec get from 2 to last
      in encodedList insert at last as dec
  return encodedList

set signalA to create list with "0101010" "1010101" "1110100" "1010111" "0001001"
set signalB to create list with "0100101" "0010010" "1001010" "0011101" "1111110"
set signalC to create list with "0000001" "0000100" "0001001" "0010000" "0011001"
objectA say: signalA for 10 secs
objectB say: signalB for 10 secs
objectC say: signalC for 10 secs
When click on decodeButton execute the function onClick
```



```
from time import sleep
def decodeSignal(binSignal):
  decodedList = []
  for binStr in binSignal:
    decodedList.append(int(binStr, 2))
  return decodedList

def onClick():
  decodedA = decodeSignal(signalA)
  decodedB = decodeSignal(signalB)
  decodedC = decodeSignal(signalC)
  objectA.say(decodedA,10)
  objectB.say(decodedB,10)
  objectC.say(decodedC,10)
  sleep(5)
  earthReply = [36,49,64,81,100]
  earth.say(encodeSignal(earthReply),10)

def encodeSignal(decSignal):
  encodedList = []
  for dec in decSignal:
    dec = bin(dec)
    dec = dec[2:]
    encodedList.append(dec)
  return encodedList


signalA = ["0101010", "1010101", "1110100", "1010111", "0001001"]
signalB = ["0100101", "0010010", "1001010", "0011101", "1111110"]
signalC = ["0000001", "0000100", "0001001", "0010000", "0011001"]
objectA.say(signalA,10)
objectB.say(signalB,10)
objectC.say(signalC,10)
pip.eventmanager.when_click(decodeButton,onClick)

```

Challenge

Try out the following challenges.

Suggestions

- Make it so that the user can enter their own sequence of numbers to be sent to the aliens.
- Do some internet research and find the Arecibo message. Try and write it as a long list of 1's and 0's, separated by commas. It starts off: [“00000010101010000000000”, “00101000001010000000100”...]
- See if you can encode it as a sequence of decimal numbers.

Lesson 5 – Morse code

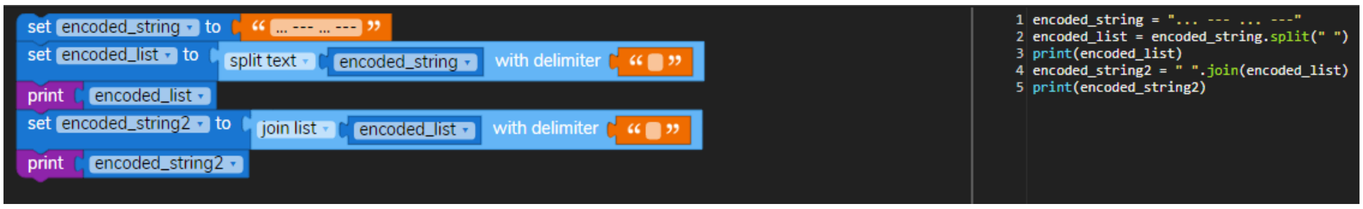
Stage 1- String split and join

If you have a string, you can split it by specifying a ‘delimiter’ and you will get a list containing the individual items. The reverse process is called ‘join’. Given a delimiter and a list, the join method will join the items together.
“&”.join([“Andy”, “Bala”, “Chris”, “Denise”])
Notice that you do not write list.join(delimiter), you write delimiter.join(list). This is because ‘join’ is a method of the String class, not the List class

Tasks

- Use ‘split’ to split ‘encoded_string’ into a list called ‘encoded_list’ using space as the delimiter. Print the list.
- Use ‘join’ to join ‘encoded_list’ into a string called ‘encoded_string2’. Use space as the delimiter. Print it and check that you get back the original string.

Solution



```
set encoded_string to “ ... --- ... --- ”
set encoded_list to split text encoded_string with delimiter “ ”
print encoded_list
set encoded_string2 to join list encoded_list with delimiter “ ”
print encoded_string2
```

```python
encoded_string = "... --- ... ---"
encoded_list = encoded_string.split(" ")
print(encoded_list)
encoded_string2 = " ".join(encoded_list)
print(encoded_string2)
```

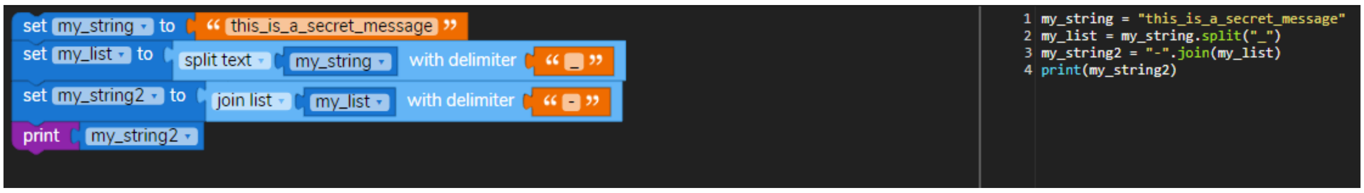
Stage 2- Split and join - debug

Someone wants to split the string “this_is_a_secret_message” using underscore (_) as the delimiter and then join the list together with dashes (-).
Their program doesn’t work, see if you can fix it.

Tasks

- Fix the code. It should print “this-is-a-secret-message”. Remember the syntax is ‘delimiter.join(list)’.

Solution



```
set my_string to “ this_is_a_secret_message ”
set my_list to split text my_string with delimiter “ _ ”
set my_string2 to join list my_list with delimiter “ - ”
print my_string2
```

```python
my_string = "this_is_a_secret_message"
my_list = my_string.split("_")
my_string2 = "-".join(my_list)
print(my_string2)
```

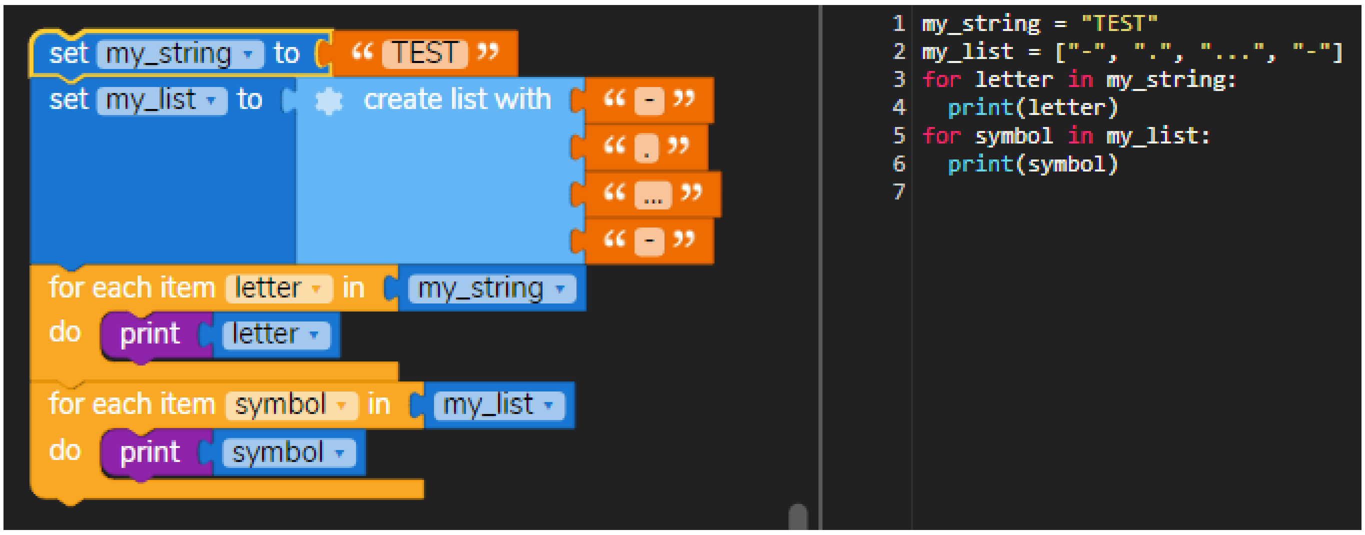
Stage 3- 'for-in' loops

In Python you can use a 'for-in' loop to go through each letter in a string, or each item in a list. The code is the same.

Tasks

- Use a 'for letter in my_string' loop to print each letter in the string.
- Use a 'for symbol in my_list' loop to print each symbol in the list.

Solution


set my_string to "TEST"
set my_list to create list with "-" "." "..." "-"
for each item letter in my_string
do print letter
for each item symbol in my_list
do print symbol

```
my_string = "TEST"
my_list = ["-", ".", "...", "-"]
for letter in my_string:
  print(letter)
for symbol in my_list:
  print(symbol)
```

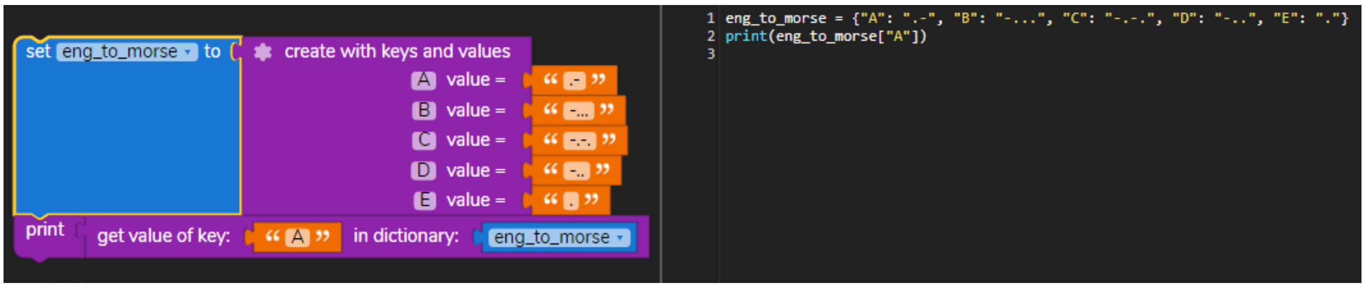
Stage 4- Dictionaries

We will also need to find a way to get the Morse code symbol for a letter. To do this we need to use a 'dictionary'. Here is a dictionary, it stores some letters and their Morse equivalents:
The English letters are called the 'keys' of the dictionary and the Morse versions are called the 'values'. You can access a value using square brackets:
eng_to_morse["C"]

Tasks

- Print "A" in Morse code.

Solution


set eng_to_morse to create with keys and values
A value = ".-"
B value = "-..."
C value = "-.-."
D value = "-.."
E value = "."
print get value of key: "A" in dictionary: eng_to_morse

```
eng_to_morse = {"A": ".-", "B": "-...", "C": "-.-.", "D": "-..", "E": "."}
print(eng_to_morse["A"])
```

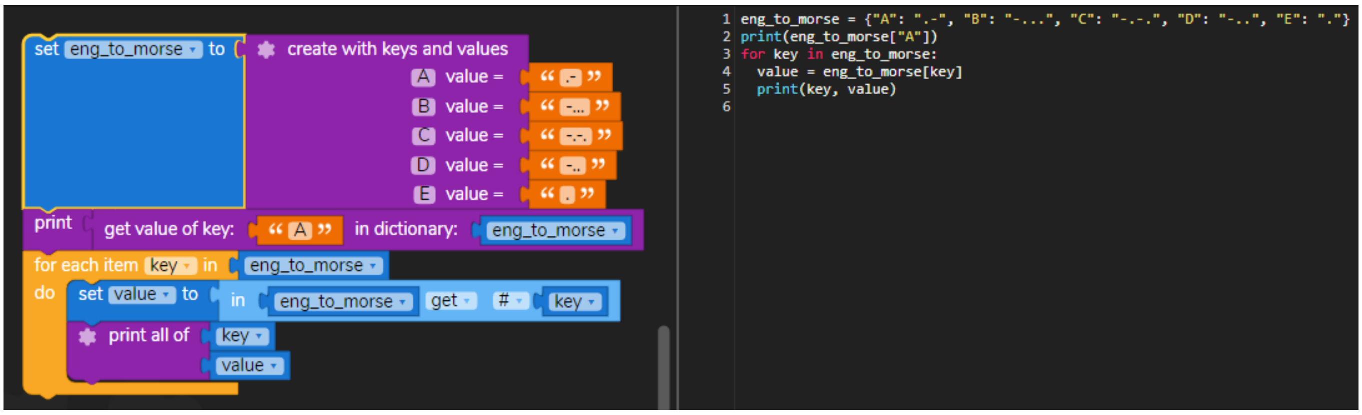
Stage 5- Looping through a dictionary

You can use a for-in loop to loop through a dictionary as well.
It will give you the keys of the dictionary (in our example this is the English letters)

Tasks

- Use 'for key in eng_to_morse' to loop over the dictionary keys.
- Inside your loop, make a variable called 'value' which should be set to 'eng_to_morse[key]'.
- Use 'print(key, value)' to print the items one by one. Check that this prints the letters and their Morse equivalents.

Solution


set eng_to_morse to create with keys and values
A value = " .- "
B value = " -... "
C value = " -.-. "
D value = " -.. "
E value = " . "
print get value of key: " A " in dictionary: eng_to_morse
for each item key in eng_to_morse
do set value to in eng_to_morse get # key
print all of key value

```
eng_to_morse = {"A": ".-", "B": "-...", "C": "-.-.", "D": "-..", "E": "."}
print(eng_to_morse["A"])
for key in eng_to_morse:
  value = eng_to_morse[key]
  print(key, value)
```

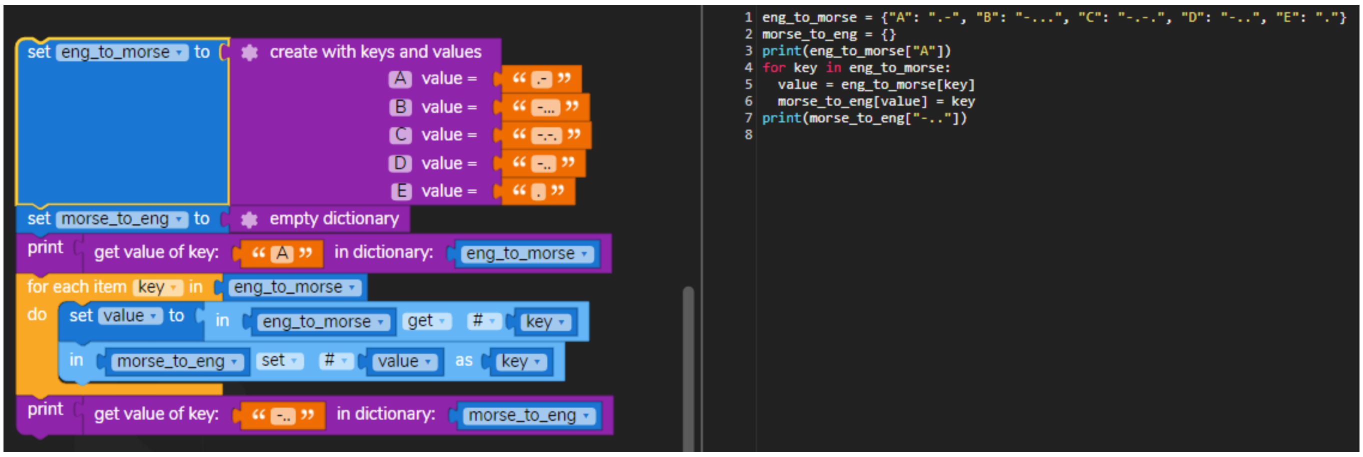
Stage 6- Reversing a dictionary

What if I give you ' - . . ' and I want to look up the corresponding English letter?

Tasks

- At the top of your program make an empty dictionary called 'morse_to_eng'. You can write 'morse_to_eng = { } ' to do this.
- Inside your for loop, set 'morse_to_eng[value] = key'. Notice how key and value are reversed here.
- At the end of your program, print 'morse_to_eng["-.."] and check it gives you "D"

Solution


set eng_to_morse to create with keys and values
A value = " .- "
B value = " -... "
C value = " -.-. "
D value = " -.. "
E value = " . "
set morse_to_eng to empty dictionary
print get value of key: " A " in dictionary: eng_to_morse
for each item key in eng_to_morse
do set value to in eng_to_morse get # key
in morse_to_eng set # value as key
print get value of key: " -.. " in dictionary: morse_to_eng

```
eng_to_morse = {"A": ".-", "B": "-...", "C": "-.-.", "D": "-..", "E": "."}
morse_to_eng = {}
print(eng_to_morse["A"])
for key in eng_to_morse:
  value = eng_to_morse[key]
  morse_to_eng[value] = key
print(morse_to_eng["-.."])
```

Stage 7- Encoding a message

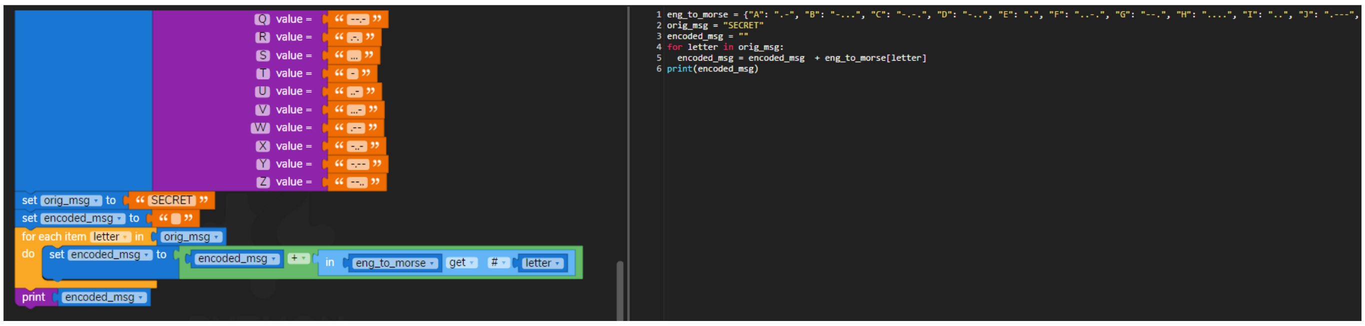
We're finally ready to encode a message. Our algorithm will need to:

- Make an empty string for the encoded message.
- Loop through each letter of the message and find its encoded version.
- Append them to the string

Tasks

- Make an empty string (“”) called ‘encoded_msg’.
- Loop through ‘orig_msg’ using a ‘for letter in orig_msg’ loop.
- For each letter, find the encoded version from the dictionary and append it to the end of ‘encoded_msg’ using ‘ + ‘. You can write ‘encoded_msg = encoded_msg + ….’.
- Run your code and look in the debug panel (bottom-right). You should see that ‘encoded_msg’ contains the encoded combinations of dots and dashes without any spaces between. We'll fix that next.

Solution



```
eng_to_morse = {"A": ".-", "B": "-...", "C": "-.-.", "D": "-..", "E": ".", "F": "..-.", "G": "--.", "H": "....", "I": "..", "J": ".---",
orig_msg = "SECRET"
encoded_msg = ""
for letter in orig_msg:
  encoded_msg = encoded_msg  + eng_to_morse[letter]
print(encoded_msg)
```

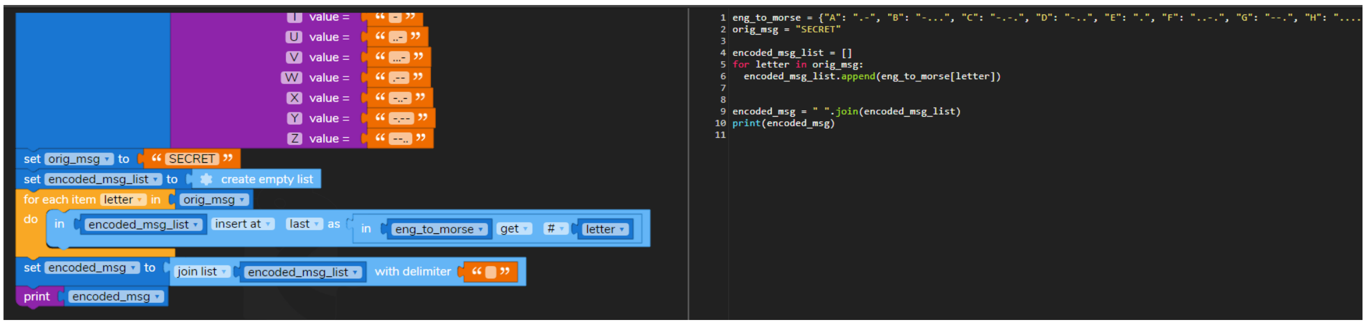
Stage 8- Encoding with spaces

Our encoded message isn't correct. Messages in Morse code need a space between each letter - in reality, this would correspond to a short pause in the transmission. There are many different ways of doing this, the neatest way is to put the items in a list and then ‘join’ the elements of the list together using a space.

Tasks

- Delete the code that creates the empty string ‘encoded_msg’. Replace it with an empty list, ‘encoded_msg_list = [ ]’
- Instead of using + to append to the string, use the list ‘.append’ method to append each Morse symbol to the list.
- At the end of your program, make a new variable called ‘encoded_msg’ which is equal to ‘encoded_msg_list’, joined together by spaces. Use the ‘join’ method.
- Check that the debug panel conains the correct value for ‘encoded_msg’.

Solution



```
eng_to_morse = {"A": ".-", "B": "-...", "C": "-.-.", "D": "-..", "E": ".", "F": "..-.", "G": "--.", "H": "....
orig_msg = "SECRET"

encoded_msg_list = []
for letter in orig_msg:
  encoded_msg_list.append(eng_to_morse[letter])


encoded_msg = " ".join(encoded_msg_list)
print(encoded_msg)

```

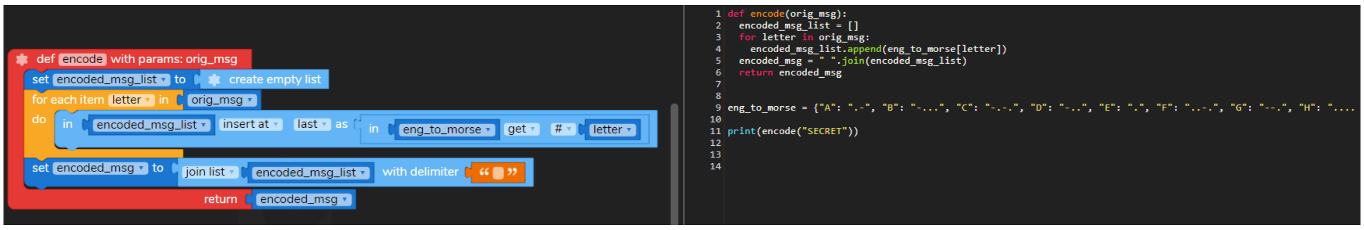
Stage 9- Using a function

Before we work on decoding it will be nice to place our 'encode' algorithm inside a function. Then we can make another function to decode.

Tasks

- Define a function called 'encode'. It should take one parameter called 'orig_msg'
- Delete the line of code ' orig_msg = "SECRET" '. You will be able to pass any string into your function.
- Test your code by adding the line 'print(encode("SECRET"))'. It should print the correct encoded string, "... . -.-. .-. . -".
- Move all your code into the function (apart from the definition of 'eng_to_morse'). At the end of your function, return the variable 'encoded_msg'.

Solution



```
def encode with params: orig_msg
  set encoded_msg_list to create empty list
  for each item letter in orig_msg
  do  in encoded_msg_list insert at last as in eng_to_morse get # letter
  set encoded_msg to join list encoded_msg_list with delimiter " "
  return encoded_msg
```

```
def encode(orig_msg):
  encoded_msg_list = []
  for letter in orig_msg:
    encoded_msg_list.append(eng_to_morse[letter])
  encoded_msg = " ".join(encoded_msg_list)
  return encoded_msg


eng_to_morse = {"A": ".-", "B": "-...", "C": "-.-.", "D": "-..", "E": ".", "F": "..-.", "G": "--.", "H": "....

print(encode("SECRET"))
```

Stage 10- Decoding

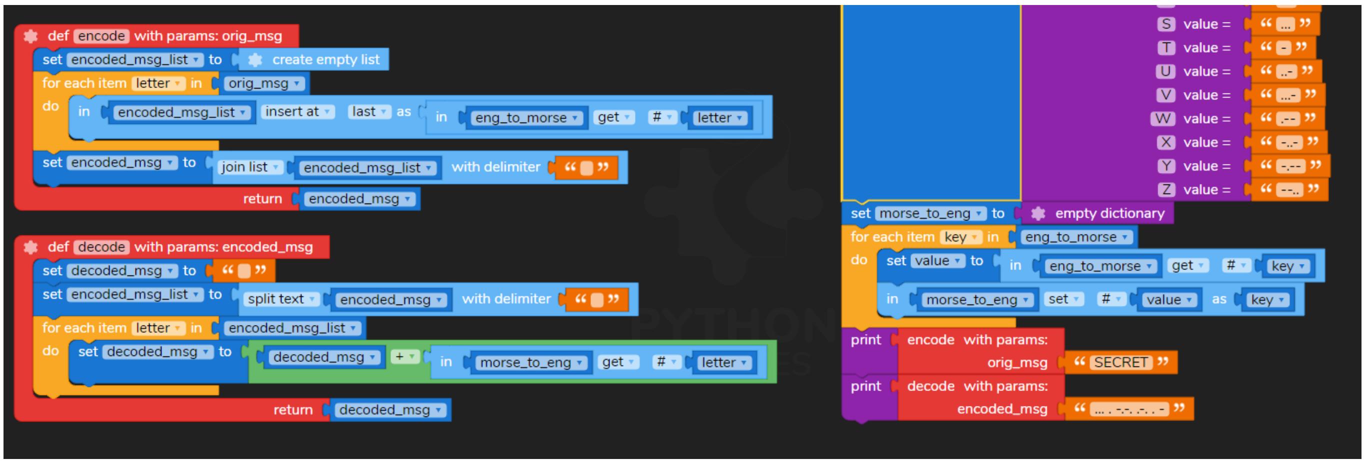
Decoding is harder. We need to split the Morse string using a space in order to get the individual symbols, and we need to be able to look up the English letter given a Morse symbol.
One thing is easier - at the end we don't need to join up the letters using a space because English doesn't need spaces between letters.

Tasks

- Make the reversed dictionary. Define an empty dictionary called 'morse_to_eng'. Loop through 'eng_to_morse', set 'value = eng_to_morse[key]' and set 'morse_to_eng[value] = key'.
- Define a new function called 'decode'. It should take one parameter called 'encoded_msg'. Inside, create an empty string called 'decoded_msg'
- Use the 'split' function to split 'encoded_msg' using space as the delimiter. Store the result as 'encoded_msg_list'.
- Use a 'for symbol in encoded_msg_list' loop to go through the symbols one by one.
- Inside the loop, get the correct English letter in the reversed dictionary and append it to 'decoded_msg'. Finally, return 'decoded_msg'
- Test your code by adding the line 'print(decode("... . -.-. .-. . -")).' It should print the word 'SECRET'

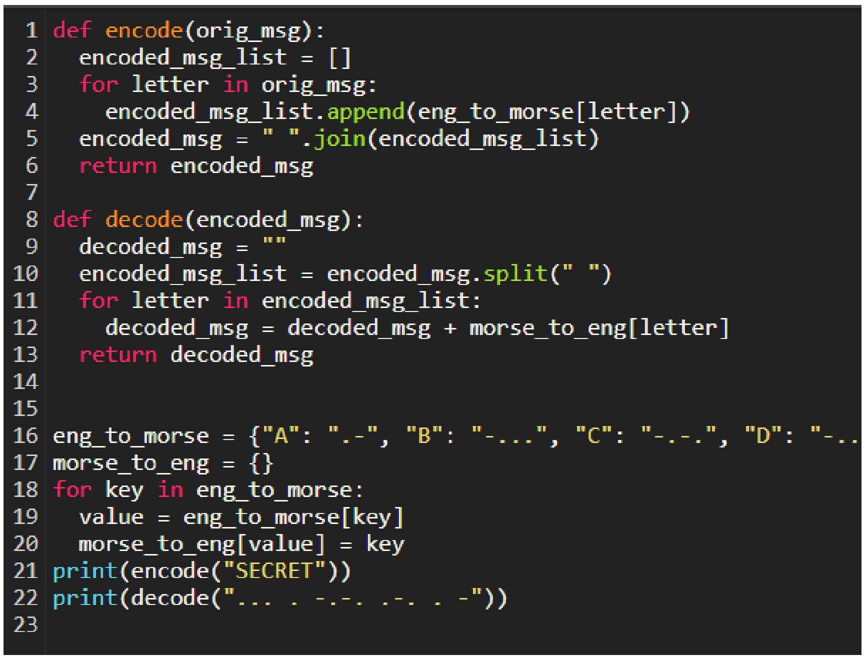
Solutions on following page...

Solution



```
def encode with params: orig_msg
  set encoded_msg_list to create empty list
  for each item letter in orig_msg
  do
    in encoded_msg_list insert at last as in eng_to_morse get # letter
  set encoded_msg to join list encoded_msg_list with delimiter " "
  return encoded_msg

def decode with params: encoded_msg
  set decoded_msg to ""
  set encoded_msg_list to split text encoded_msg with delimiter " "
  for each item letter in encoded_msg_list
  do
    set decoded_msg to decoded_msg + in morse_to_eng get # letter
  return decoded_msg
```

```
S value = "..."
T value = "-"
U value = "..-"
V value = "...-"
W value = ".--"
X value = "-..-"
Y value = "-.--"
Z value = "--.."
set morse_to_eng to empty dictionary
for each item key in eng_to_morse
do
  set value to in eng_to_morse get # key
  in morse_to_eng set # value as key
print encode with params:
            orig_msg "SECRET"
print decode with params:
            encoded_msg "... . -.-. .-. . -"
```



```python
def encode(orig_msg):
  encoded_msg_list = []
  for letter in orig_msg:
    encoded_msg_list.append(eng_to_morse[letter])
  encoded_msg = " ".join(encoded_msg_list)
  return encoded_msg

def decode(encoded_msg):
  decoded_msg = ""
  encoded_msg_list = encoded_msg.split(" ")
  for letter in encoded_msg_list:
    decoded_msg = decoded_msg + morse_to_eng[letter]
  return decoded_msg


eng_to_morse = {"A": ".-", "B": "-...", "C": "-.-.", "D": "-..
morse_to_eng = {}
for key in eng_to_morse:
  value = eng_to_morse[key]
  morse_to_eng[value] = key
print(encode("SECRET"))
print(decode("... . -.-. .-. . -"))

```

Challenges

Try out the following challenges.

Tasks

- Make a new function called 'playMorse'. It should take one parameter called 'morse_ string' which will be a string.
- Inside the function, loop through each character of the string. If it is a dot, play the sound 'sfx/dot.mp3' and sleep for a quarter of a second
- If it is a dash play the sound 'sfx/dash.mp3' and sleep for half second
- If it is a space sleep for a second. Test your code by encoding another word and playing the sounds.
- Go into design mode and find the lantern. For a dot character, show the lantern for a quarter of a second
- For a dash, show the lantern for a second.
- So far we have only dealt with single words. To deal with a sentence you need to encode spaces as slash (/). You will need to edit the dictionary as well as change some of the logic in the encode and decode functions.

Lesson 6 – Linear Search

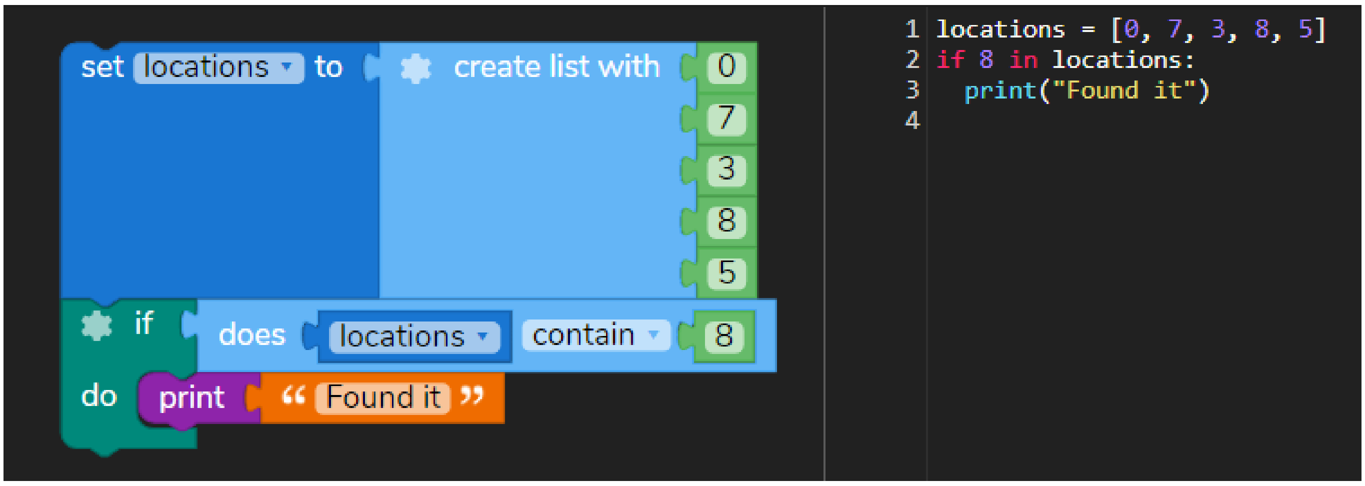
Stage 1- Using ‘in’

Python has a simple and powerful way to search lists. The ‘in’ operator will tell us if a list contains a given item.
It returns True or False.

Tasks

- If the number 8 is in the list print “Found it”. Use the ‘in’ keyword and an ‘if’ statement.

Solution


set locations to create list with 0 7 3 8 5
if does locations contain 8
do print “ Found it ”

```python
locations = [0, 7, 3, 8, 5]
if 8 in locations:
  print("Found it")
```

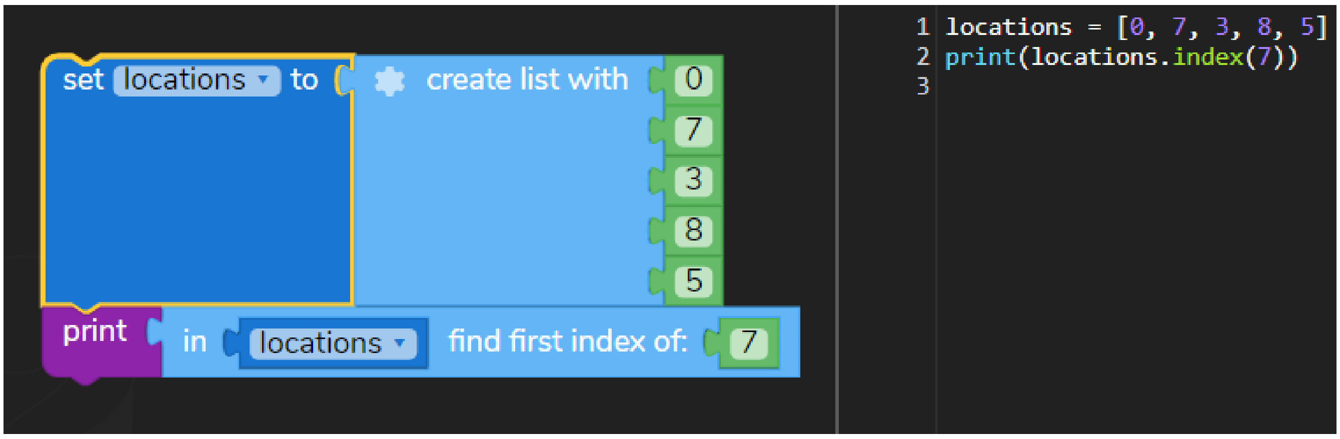
Stage 2- The ‘index’ method

But what if we want to know the position of an item as well? In this case, the ‘in’ keyword doesn’t really help us.
Python lists have a method called ‘index’ that will tell us the index of the first matching item.
Remember that in a list, the index starts from zero.

Tasks

- Print the index of the number 7 in the list. Use the ‘index’ keyword.

Solution


set locations to create list with 0 7 3 8 5
print in locations find first index of: 7

```python
locations = [0, 7, 3, 8, 5]
print(locations.index(7))
```

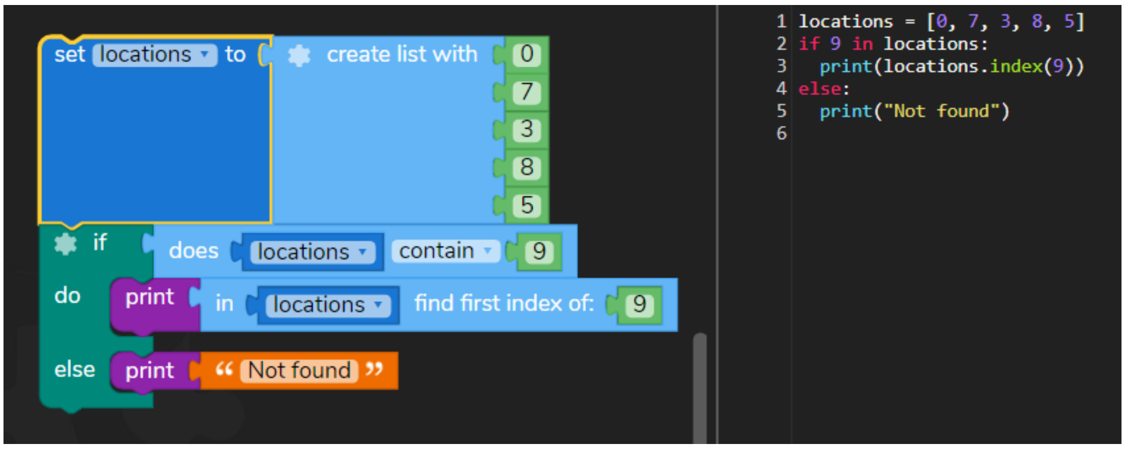
Stage 3- Missing items

If a list doesn't contain a specific item then the 'index' method will give an error.
It is best to first check if the item is there and only then find its index.

Tasks

- Run the code and check that it gives an error. This is because 9 is not in the list. Add an 'if' statement to fix the code. Use 'in' to check if the 9 is in the list and only then get the index.
- Add a corresponding 'else' statement which prints "Not found". Run your code and check that it prints 'Not found' for the item 9.

Solution



```python
locations = [0, 7, 3, 8, 5]
if 9 in locations:
  print(locations.index(9))
else:
  print("Not found")
```

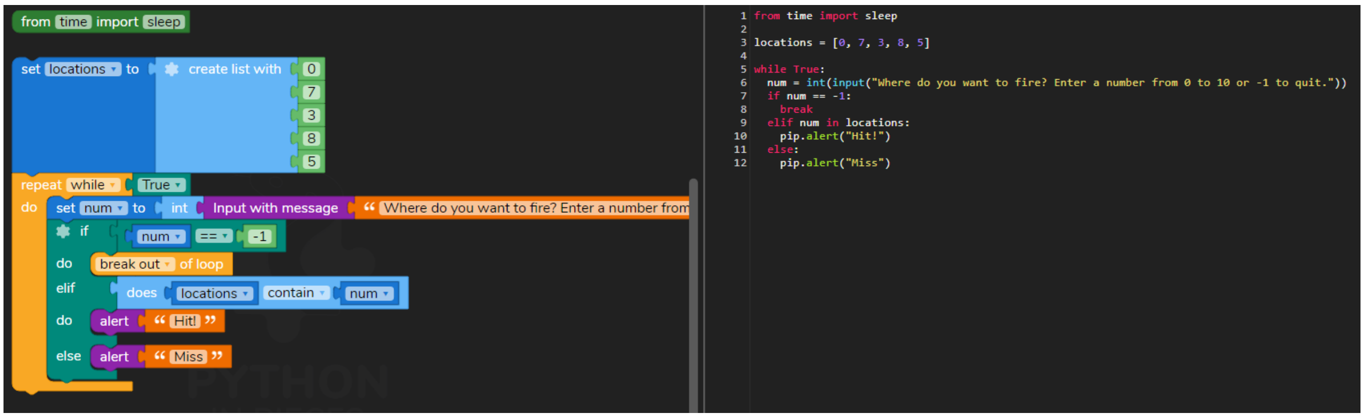
Stage 4- Alien attack

In the next few stages we will make a game using a list. Aliens are attacking and the locations of their spaceships are stored in a list called 'locations'.

Tasks

- Add a 'while True' loop. Inside, add an 'input' statement to ask the user "Where do you want to fire? Enter a number from 0 to 10, or -1 to quit". Save their response in a variable called 'num'. Convert it to a number using the 'int' function.
- Add an 'if' statement and check if 'num' is -1. If so break out of the loop using a 'break' statement.
- Else, check if 'num' is in the list using the 'in' keyword.
- If it is, alert "Hit". Else alert "Miss". Use pip.alert.
- Test that it works by choosing number '5'. Enter -1 to quit when you are ready.

Solution



```python
from time import sleep

locations = [0, 7, 3, 8, 5]

while True:
  num = int(input("Where do you want to fire? Enter a number from 0 to 10 or -1 to quit."))
  if num == -1:
    break
  elif num in locations:
    pip.alert("Hit!")
  else:
    pip.alert("Miss")
```

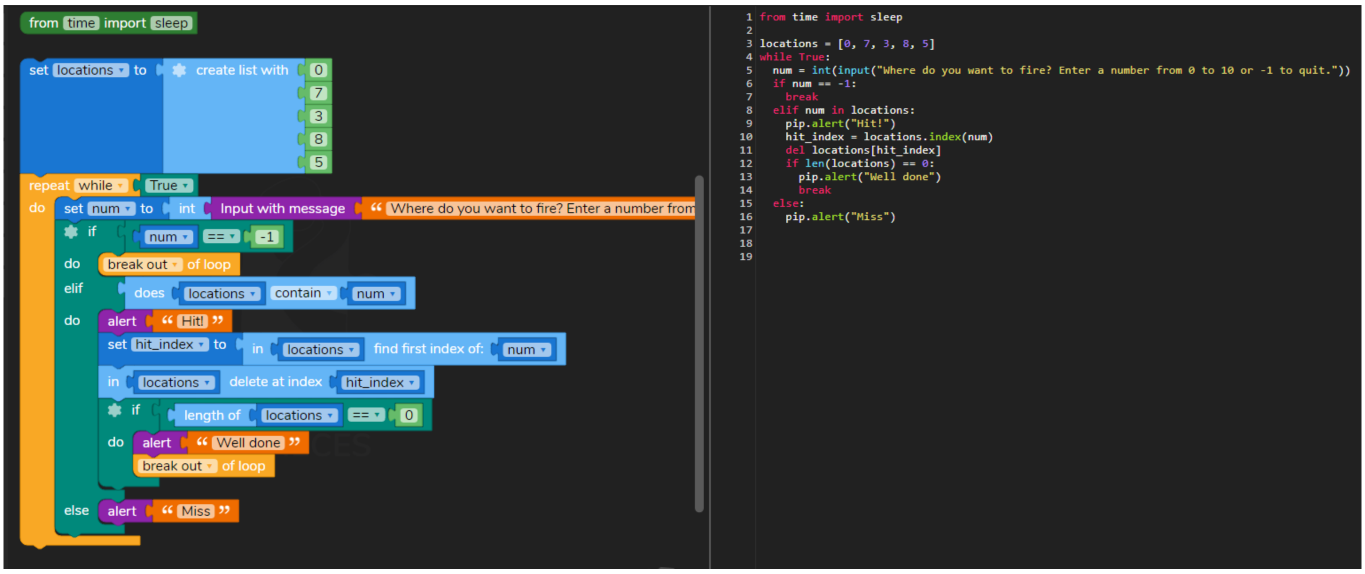
Stage 5- Alien attack 2

When you hit a ship we need to find the index and remove the ship at that index so you can't shoot it again.

Tasks

- After alerting "Hit", find the index of 'num' in the list and store it in a variable called 'hit_index'.
- Use the 'del' keyword to delete the item at index 'hit_index' in the list.
- Check if the length of the list is zero using the 'len' function and if so print "Well done" and break out of the loop.
- Test your game and check that it works by destroying the ships. They are at locations 0, 7, 3, 8 and 5.

Solution



```python
from time import sleep

locations = [0, 7, 3, 8, 5]
while True:
  num = int(input("Where do you want to fire? Enter a number from 0 to 10 or -1 to quit."))
  if num == -1:
    break
  elif num in locations:
    pip.alert("Hit!")
    hit_index = locations.index(num)
    del locations[hit_index]
    if len(locations) == 0:
      pip.alert("Well done")
      break
  else:
    pip.alert("Miss")
```

Stage 6- Linear search

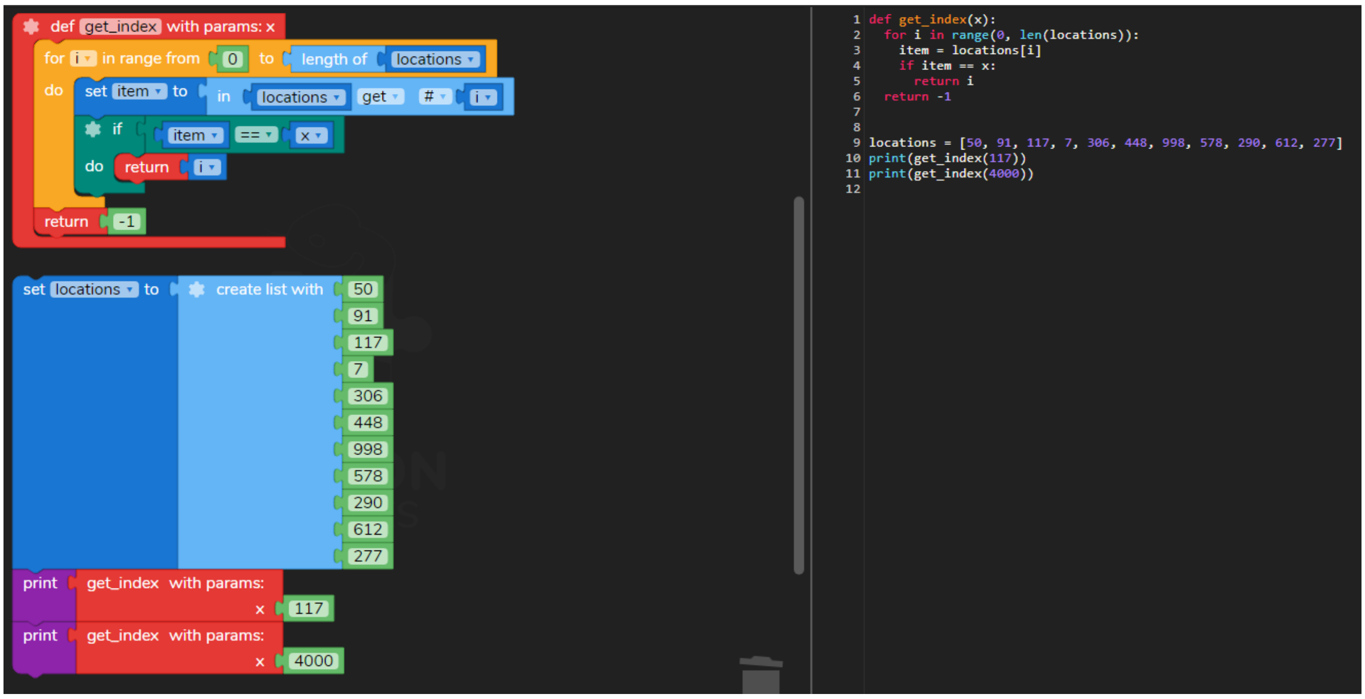
Another way to find the position of an item in a list is to use a 'linear search'. We'll make a function that checks the item at index 0, then index 1, then index 2, and so on. A good way to do this in Python is to use a 'range'.
If we find a match we'll return the index. We'll return -1 from our function to mean that the item wasn't found, this is a standard used in many programming languages.

Tasks

- Create a function called 'get_index' which takes one parameter called 'x'.
- Inside, write a 'for i in range' loop. Make sure that 'i' goes through each index of the list. You should specify the end of the range using the 'len' function.
- Inside the for loop, create a variable, 'item = locations[i]'. Use an 'if' statement to check if 'item' is equal to 'x'. If so, return the value of 'i'.
- At the end of your function return -1. This will happen if the item is never found.
- Test your function by printing 'get_index(117)' and 'get_index(4000)'. Check that it prints 2 and -1.

Solutions on following page...

Solution



```
def get_index(x):
  for i in range(0, len(locations)):
    item = locations[i]
    if item == x:
      return i
  return -1


locations = [50, 91, 117, 7, 306, 448, 998, 578, 290, 612, 277]
print(get_index(117))
print(get_index(4000))
```

Stage 7- Applying a test

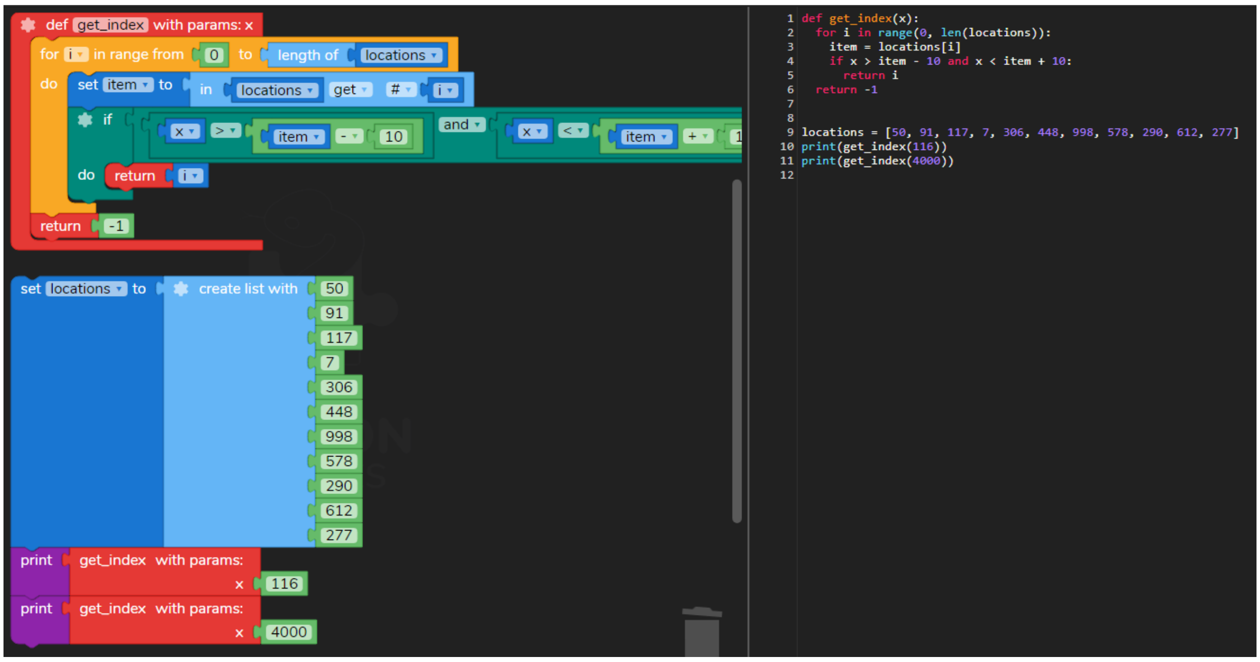
Sometimes we don’t want to check if an item matches something, but instead we want to check if an item passes some kind of test. For example, if we have two sprites we could check if they are within 10 pixels of each other. This is something you can’t do with ‘index’, but you can with a linear search.
First, notice that ‘x’ is within 10 pixels of ‘y’ if x is greater than (y - 10) and less than (y + 10).

Tasks

- Inside your function, change the line which checks if ‘item’ is equal to ‘x’. Instead, check if ‘x > item - 10 and x < item + 10’
- Is there a number close to 116 in the list? Which one? What about close to 4000? Test your function by printing get_index(116) and get_index(4000). Check that it prints 2 and -1.

Solution on following page...

Solution



```
def get_index(x):
  for i in range(0, len(locations)):
    item = locations[i]
    if x > item - 10 and x < item + 10:
      return i
  return -1


locations = [50, 91, 117, 7, 306, 448, 998, 578, 290, 612, 277]
print(get_index(116))
print(get_index(4000))
```

Stage 8- Return of the aliens

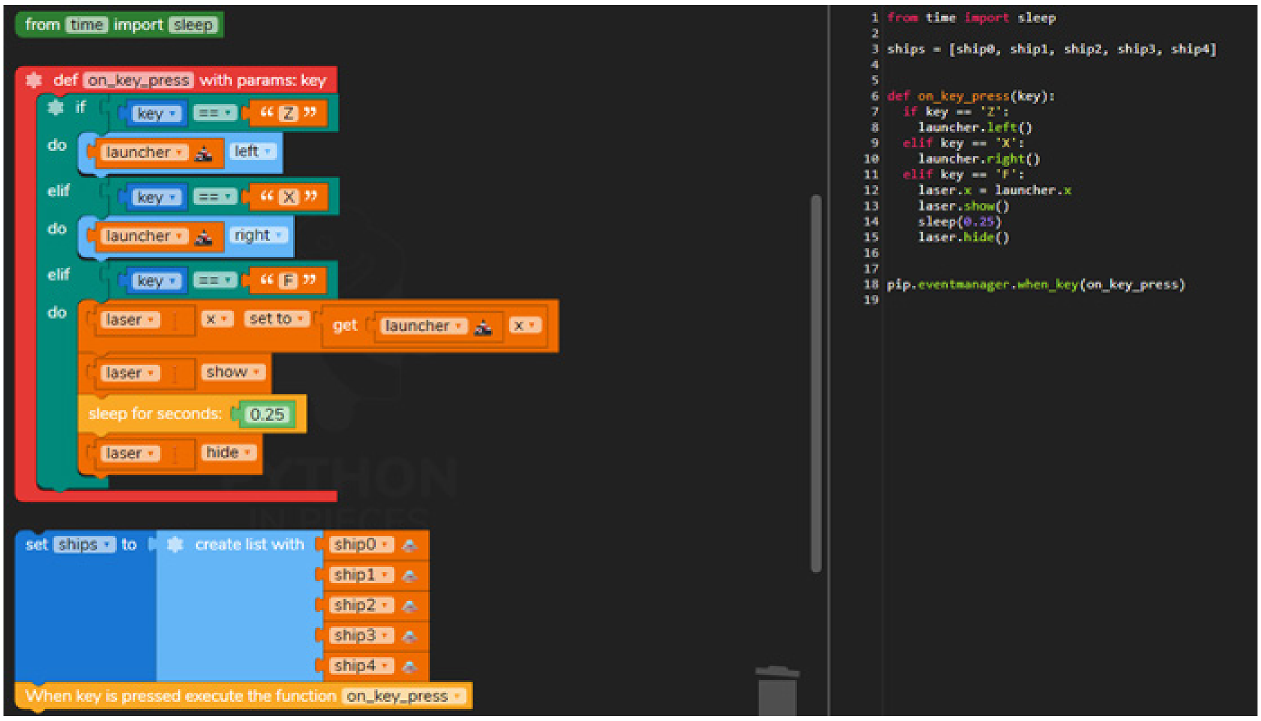
The aliens are attacking again. This time we will add a visual aspect to the game.
There are five ships, called ‘ship0’, ‘ship1’, ‘ship2’, ‘ship3’ and ‘ship4’. They are stored in a list. We will fire lasers at them. We’ll never get the position spot-on but we can check if we are close using a linear search.
First let’s make the player move the launcher and fire.

Tasks

- Create a function called ‘on_key_press’ which takes one parameter called ‘key’.
- Inside, check if ‘key’ is equal to ‘Z’. If so, make the launcher move left. Else if it equals ‘X’, make it move right.
- Else if it equals ‘F’, set ‘laser.x’ equal to ‘launcher.x’, show the laser, and after a quarter of a second, hide it.
- Finally, make your program execute the function ‘on_key_press’ when any key is pressed. You can use the ‘when key is pressed’ block or the code ‘pip.eventmanager.when_key(on_key_press)’.
- Test that your program works by moving left and right a few times.

Solutions on following page...

Solution



```
from time import sleep

def on_key_press with params: key
    if key == "Z"
    do launcher left
    elif key == "X"
    do launcher right
    elif key == "F"
    do laser x set to get launcher x
       laser show
       sleep for seconds: 0.25
       laser hide

set ships to create list with ship0 ship1 ship2 ship3 ship4
When key is pressed execute the function on_key_press
```

```
from time import sleep

ships = [ship0, ship1, ship2, ship3, ship4]


def on_key_press(key):
  if key == 'Z':
    launcher.left()
  elif key == 'X':
    launcher.right()
  elif key == 'F':
    laser.x = launcher.x
    laser.show()
    sleep(0.25)
    laser.hide()


pip.eventmanager.when_key(on_key_press)
```

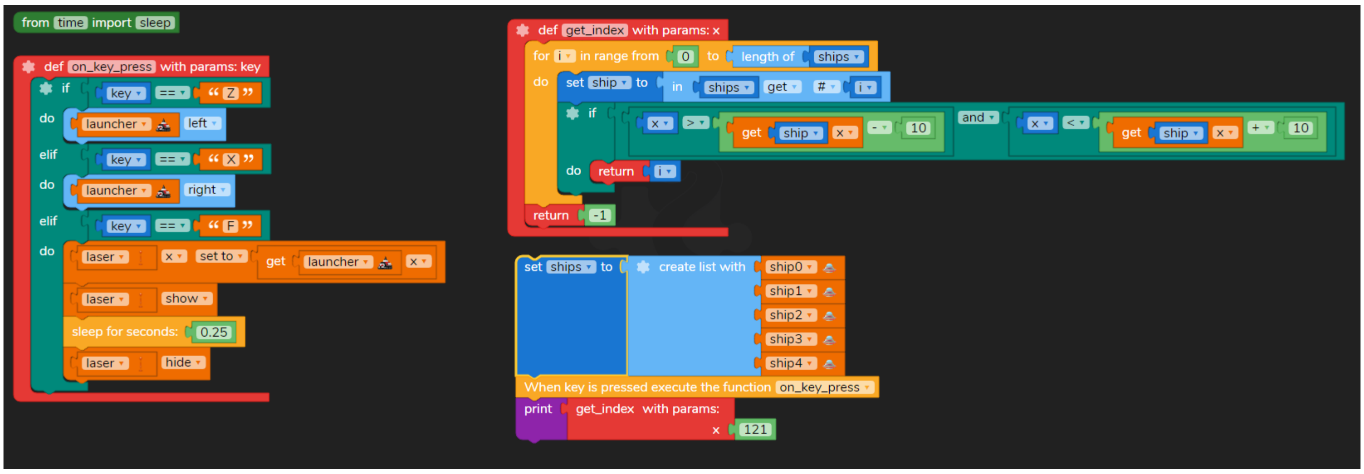
Stage 9- Return of the aliens 2

We need to check we are close to one of the ships. We'll use a function called 'get_index' which will tell us which index (if any) we are close to. Remember -1 means that we are not close to anything.

Tasks

- Create a new function called 'get_index' which takes one parameter called 'x'.
- Inside, add a 'for i in range' loop. Use the 'len' function to make sure that 'i' goes through each index of the list 'ships'.
- Set 'ship = ships[i]'. This will be the ship at index 'i'.
- Use an 'if' statement to check if 'ship.x' is close to 'x'. To do this you can write 'if x > ship.x - 10 and x < ship.x + 10'. If it is, return 'i'.
- At the end of your function return -1."
- Test your function by printing the value of get_index(121). It should print 0, since the x-coordinate of the first ship is 120.

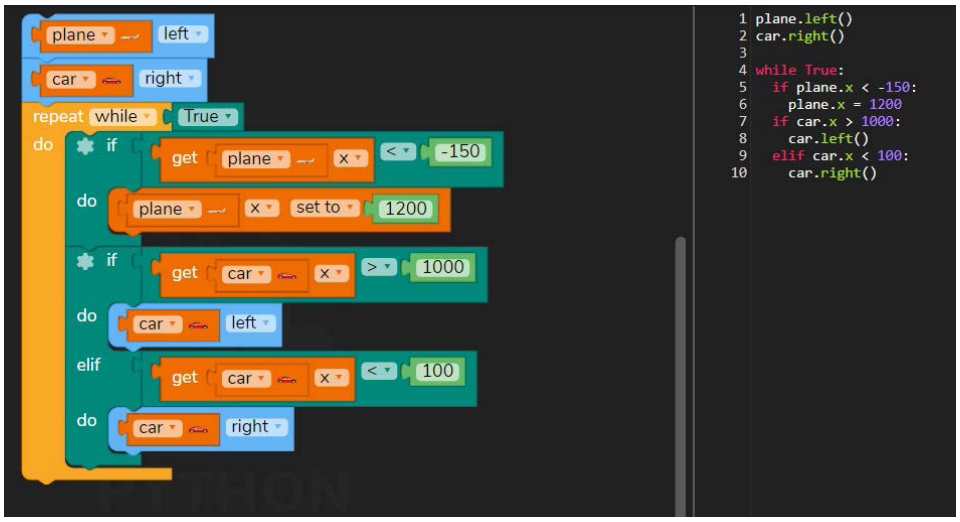
Solution



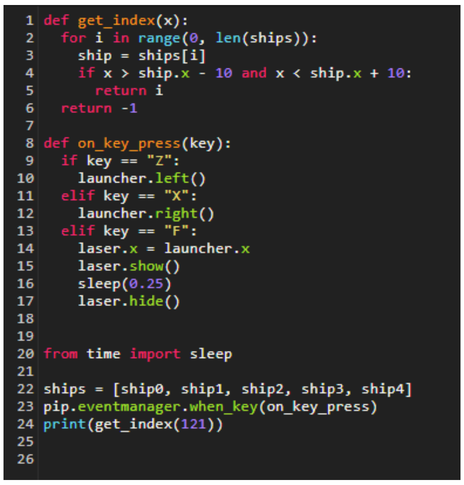
```
from time import sleep

def on_key_press with params: key
    if key == "Z"
    do launcher left
    elif key == "X"
    do launcher right
    elif key == "F"
    do laser x set to get launcher x
       laser show
       sleep for seconds: 0.25
       laser hide

def get_index with params: x
    for i in range from 0 to length of ships
    do set ship to in ships get # i
       if x > get ship x - 10 and x < get ship x + 10
       do return i
    return -1

set ships to create list with ship0 ship1 ship2 ship3 ship4
When key is pressed execute the function on_key_press
print get_index with params: x 121
```

Solution continued on following page...

Solution



```
plane.left()
car.right()

while True:
  if plane.x < -150:
    plane.x = 1200
  if car.x > 1000:
    car.left()
  elif car.x < 100:
    car.right()
```



```
def get_index(x):
  for i in range(0, len(ships)):
    ship = ships[i]
    if x > ship.x - 10 and x < ship.x + 10:
      return i
  return -1

def on_key_press(key):
  if key == "Z":
    launcher.left()
  elif key == "X":
    launcher.right()
  elif key == "F":
    laser.x = launcher.x
    laser.show()
    sleep(0.25)
    laser.hide()


from time import sleep

ships = [ship0, ship1, ship2, ship3, ship4]
pip.eventmanager.when_key(on_key_press)
print(get_index(121))


```

Stage 10- Return of the aliens 3

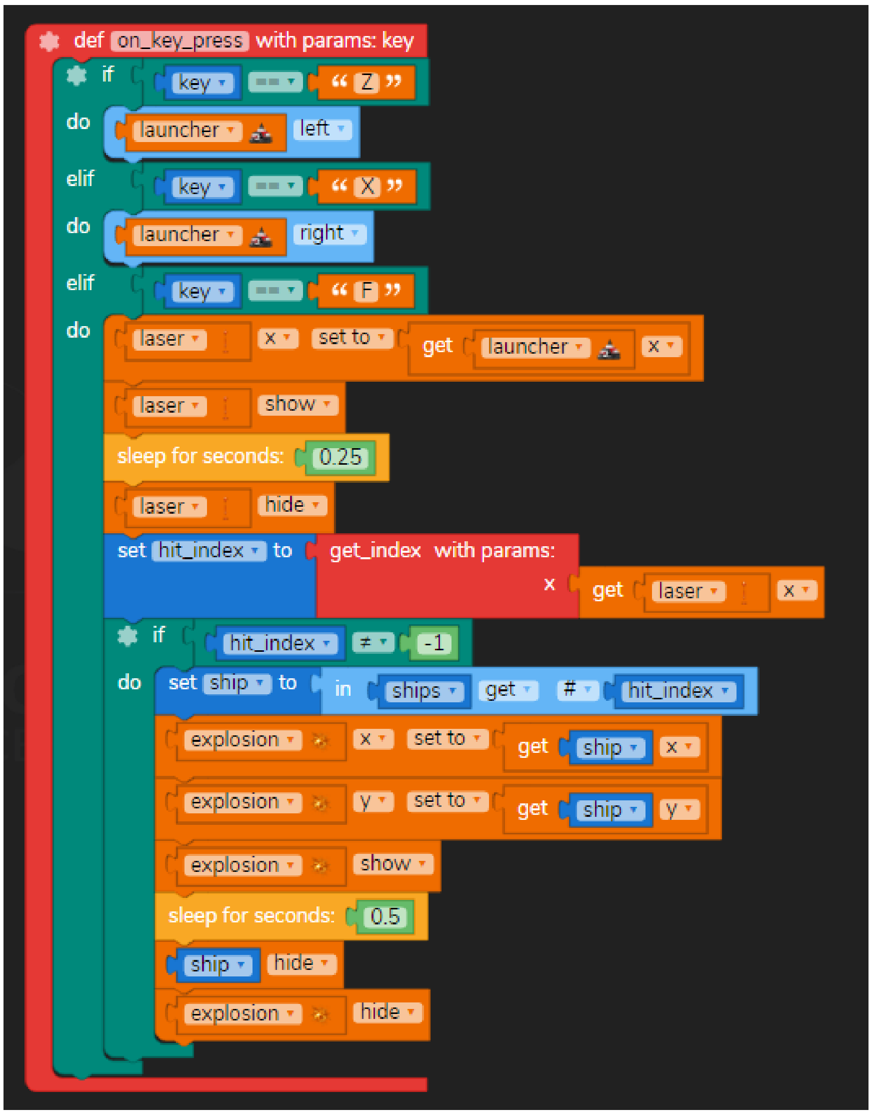
Let’s make the spaceship look like it has been hit. We need to position an explosion and then hide the ship.

Tasks

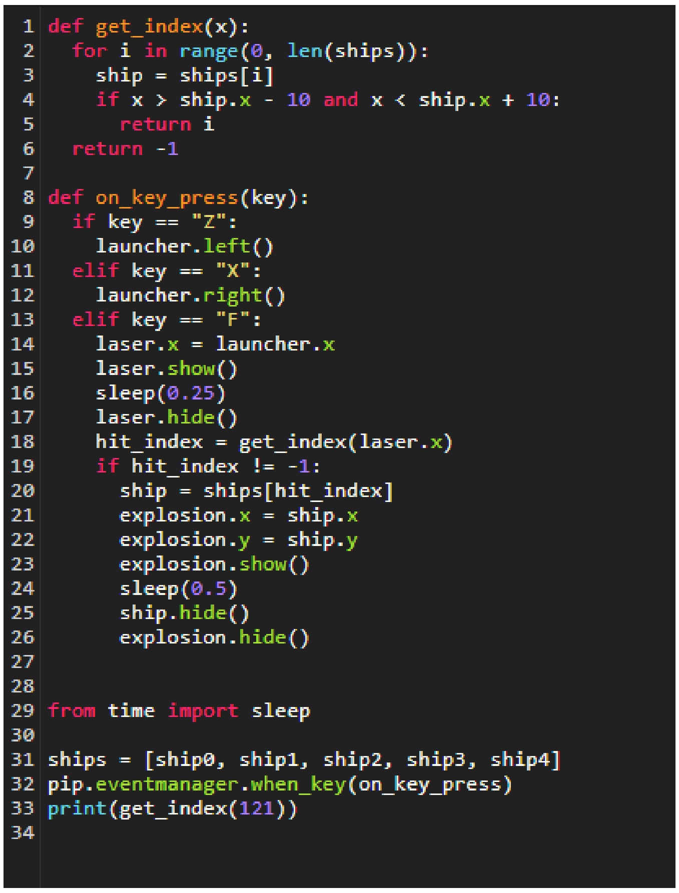
- At the end of the ‘on_key_press’ function, after hiding the laser, set ‘hit_index’ equal to ‘get_index(laser.x)’. This will tell us the index of the ship we got close to with our laser.
- Use an ‘if’ statement to check if ‘hit_index’ is not -1. In that case we hit a ship. Make a variable called ‘ship = ships[hit_index]’. This is the ship that we hit.
- Set ‘explosion.x = ship.x’ and ‘explosion.y = ship.y’, show the explosion, and after half a second hide it and also hide the ship.”
- Test your game and check that it works.

Solution on following page...

Solution



```
def on_key_press with params: key
  if key == "Z"
  do launcher left
  elif key == "X"
  do launcher right
  elif key == "F"
  do laser x set to get launcher x
     laser show
     sleep for seconds: 0.25
     laser hide
     set hit_index to get_index with params:
                              x get laser x
     if hit_index ≠ -1
     do set ship to in ships get # hit_index
        explosion x set to get ship x
        explosion y set to get ship y
        explosion show
        sleep for seconds: 0.5
        ship hide
        explosion hide
```



```python
def get_index(x):
  for i in range(0, len(ships)):
    ship = ships[i]
    if x > ship.x - 10 and x < ship.x + 10:
      return i
  return -1

def on_key_press(key):
  if key == "Z":
    launcher.left()
  elif key == "X":
    launcher.right()
  elif key == "F":
    laser.x = launcher.x
    laser.show()
    sleep(0.25)
    laser.hide()
    hit_index = get_index(laser.x)
    if hit_index != -1:
      ship = ships[hit_index]
      explosion.x = ship.x
      explosion.y = ship.y
      explosion.show()
      sleep(0.5)
      ship.hide()
      explosion.hide()


from time import sleep

ships = [ship0, ship1, ship2, ship3, ship4]
pip.eventmanager.when_key(on_key_press)
print(get_index(121))

```

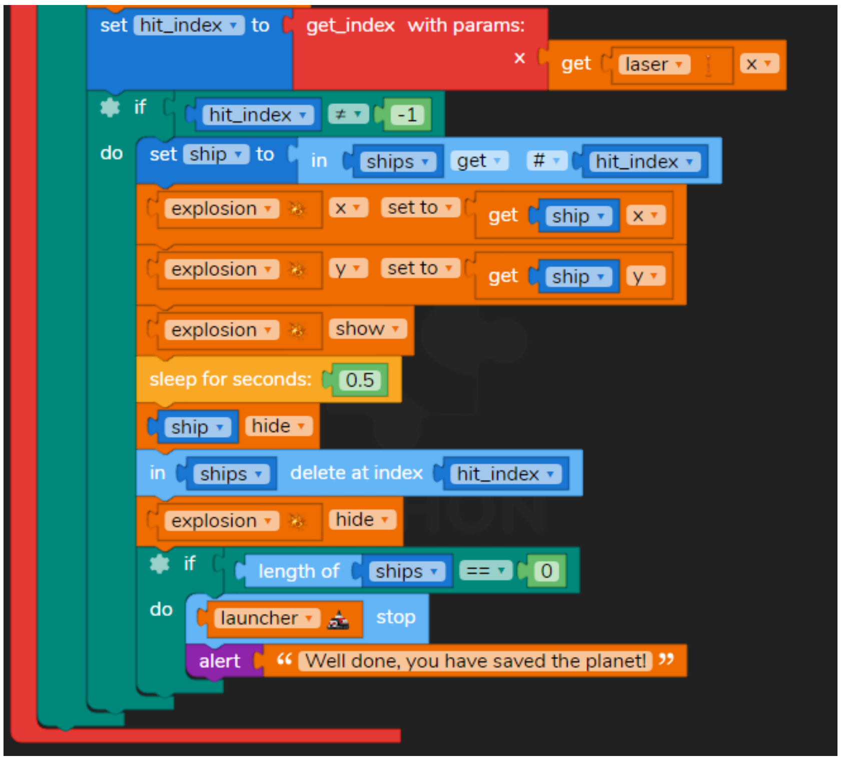
Stage 11- Return of the aliens 4

Now let's delete the ship from the list. We need to do this so you can't hit it again next time.

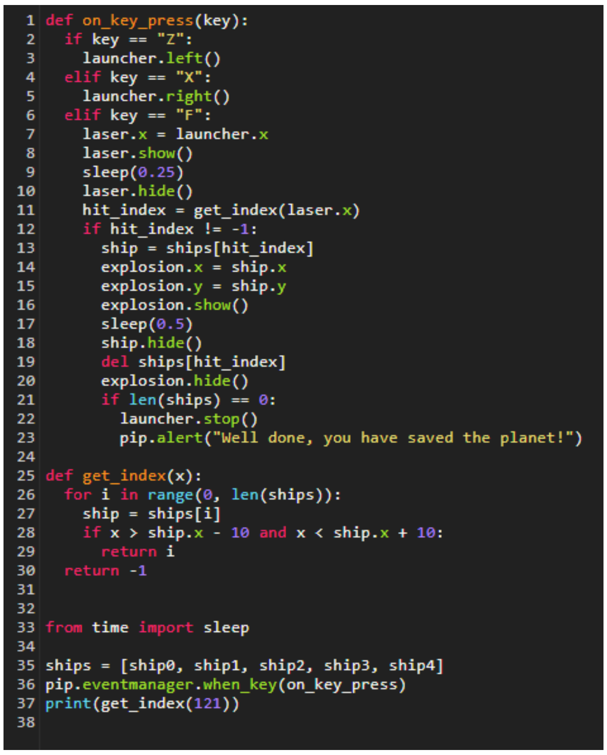
Tasks

- After hiding the ship, use the 'del' keyword to delete 'ships[hit_index]'.
- Check if the length of the list is zero using the 'len' function and if so, use pip.alert to say ("Well done") to the user, and then stop the launcher moving.
- Test your game and check that it works.

Solution



```
set hit_index to get_index with params: x get laser x
if hit_index ≠ -1
do  set ship to in ships get # hit_index
    explosion x set to get ship x
    explosion y set to get ship y
    explosion show
    sleep for seconds: 0.5
    ship hide
    in ships delete at index hit_index
    explosion hide
    if length of ships == 0
    do  launcher stop
        alert "Well done, you have saved the planet!"
```



```
def on_key_press(key):
  if key == "Z":
    launcher.left()
  elif key == "X":
    launcher.right()
  elif key == "F":
    laser.x = launcher.x
    laser.show()
    sleep(0.25)
    laser.hide()
    hit_index = get_index(laser.x)
    if hit_index != -1:
      ship = ships[hit_index]
      explosion.x = ship.x
      explosion.y = ship.y
      explosion.show()
      sleep(0.5)
      ship.hide()
      del ships[hit_index]
      explosion.hide()
      if len(ships) == 0:
        launcher.stop()
        pip.alert("Well done, you have saved the planet!")

def get_index(x):
  for i in range(0, len(ships)):
    ship = ships[i]
    if x > ship.x - 10 and x < ship.x + 10:
      return i
  return -1


from time import sleep

ships = [ship0, ship1, ship2, ship3, ship4]
pip.eventmanager.when_key(on_key_press)
print(get_index(121))

```

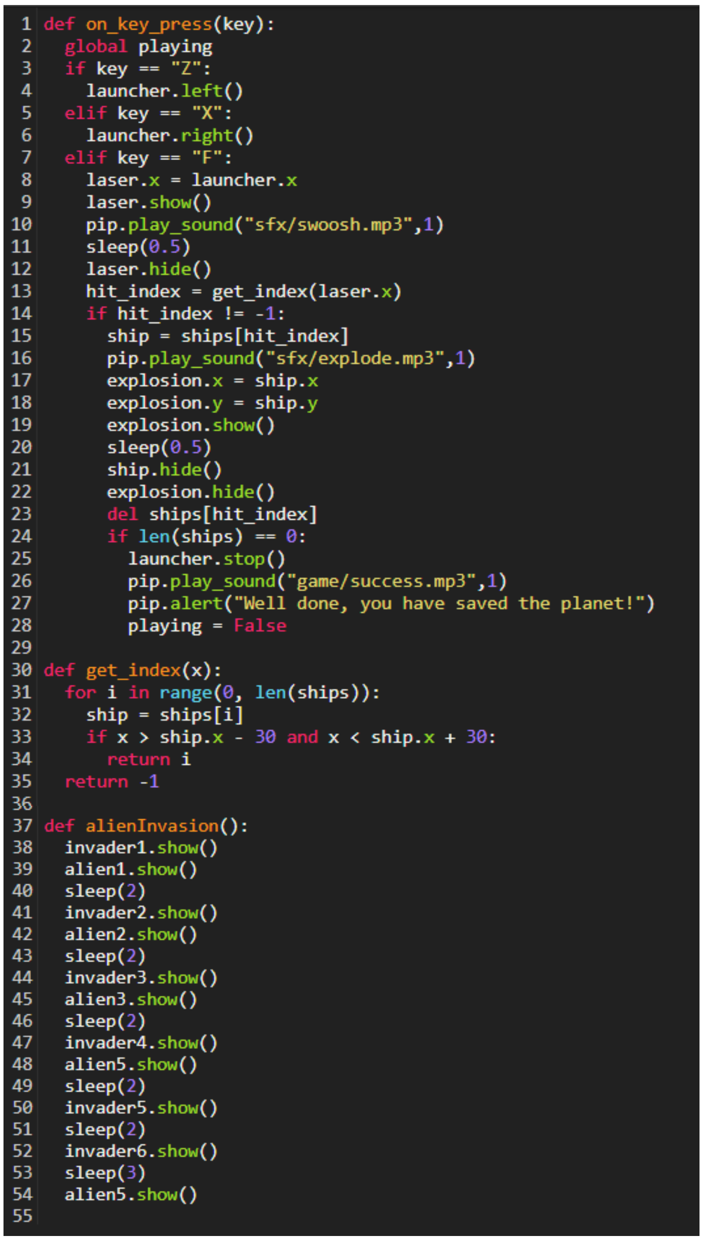
Challenge

Try out the following challenges.

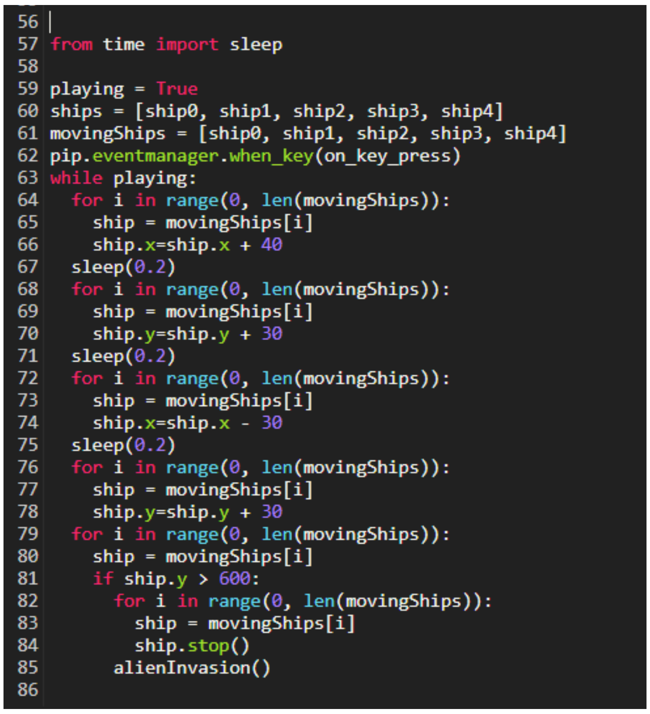
Suggestions

- Add some sound effects. Play a sound when the laser fires, when a ship is hit and at the end of the game
- Make the enemy ships fall out of the sky when they are hit
- Add a textfield and show the player's score in it. Update the score each time they hit a ship.

Possible solution



```python
def on_key_press(key):
  global playing
  if key == "Z":
    launcher.left()
  elif key == "X":
    launcher.right()
  elif key == "F":
    laser.x = launcher.x
    laser.show()
    pip.play_sound("sfx/swoosh.mp3",1)
    sleep(0.5)
    laser.hide()
    hit_index = get_index(laser.x)
    if hit_index != -1:
      ship = ships[hit_index]
      pip.play_sound("sfx/explode.mp3",1)
      explosion.x = ship.x
      explosion.y = ship.y
      explosion.show()
      sleep(0.5)
      ship.hide()
      explosion.hide()
      del ships[hit_index]
      if len(ships) == 0:
        launcher.stop()
        pip.play_sound("game/success.mp3",1)
        pip.alert("Well done, you have saved the planet!")
        playing = False

def get_index(x):
  for i in range(0, len(ships)):
    ship = ships[i]
    if x > ship.x - 30 and x < ship.x + 30:
      return i
  return -1

def alienInvasion():
  invader1.show()
  alien1.show()
  sleep(2)
  invader2.show()
  alien2.show()
  sleep(2)
  invader3.show()
  alien3.show()
  sleep(2)
  invader4.show()
  alien5.show()
  sleep(2)
  invader5.show()
  sleep(2)
  invader6.show()
  sleep(3)
  alien5.show()

```



```python

from time import sleep

playing = True
ships = [ship0, ship1, ship2, ship3, ship4]
movingShips = [ship0, ship1, ship2, ship3, ship4]
pip.eventmanager.when_key(on_key_press)
while playing:
  for i in range(0, len(movingShips)):
    ship = movingShips[i]
    ship.x=ship.x + 40
  sleep(0.2)
  for i in range(0, len(movingShips)):
    ship = movingShips[i]
    ship.y=ship.y + 30
  sleep(0.2)
  for i in range(0, len(movingShips)):
    ship = movingShips[i]
    ship.x=ship.x - 30
  sleep(0.2)
  for i in range(0, len(movingShips)):
    ship = movingShips[i]
    ship.y=ship.y + 30
  for i in range(0, len(movingShips)):
    ship = movingShips[i]
    if ship.y > 600:
      for i in range(0, len(movingShips)):
        ship = movingShips[i]
        ship.stop()
      alienInvasion()

```