Creative Computing for Engineers

Lecture 9:
Computer Programming using Python (7)

HANGMAN GAME 1

Invent Your Own Computer Games with Python

Orientation

- “Hangman”
  - Sample Run
- ASCII Art
- Designing a Program with a Flowchart

“Hangman”

- Sample Run

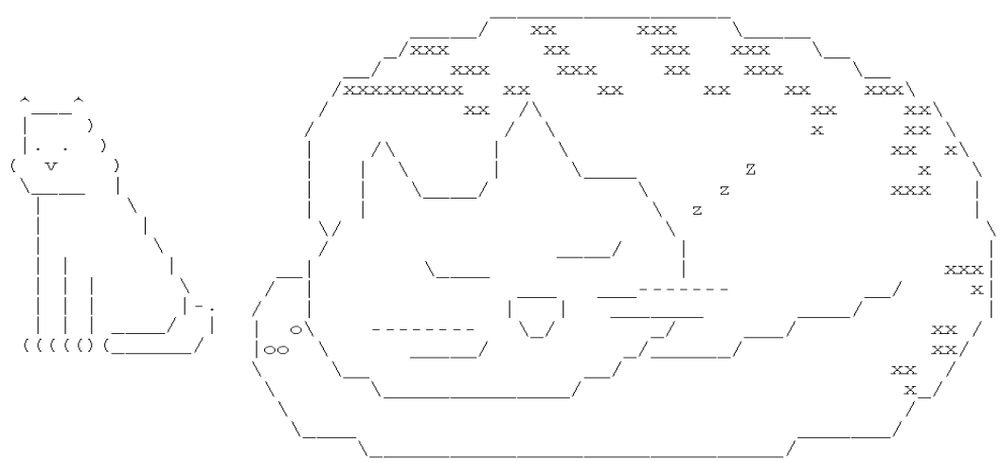
```
H A N G M A N

  +---+                 +---+
  |   |                 |   |
      |                     |
      |                     |
      |                     |
      |                     |
=========             =========
Missed letters:       Missed letters:
_ _ _                 _ a _
Guess a letter.       Guess a letter.
a                     o

  +---+                 +---+                 +---+
  |   |                 |   |                 |   |
  O   |                 O   |                 O   |
      |                 |   |                 |   |
      |                     |                     |
      |                     |                     |
=========             =========             =========
Missed letters: o     Missed letters: or    Missed letters: or
_ a _                 _ a _                 _ a t
Guess a letter.       Guess a letter.       Guess a letter.
r                     t                     a
                                            You have already guessed that letter. Choose again.
                                            Guess a letter.
                                            c
                                            Yes! The secret word is "cat"! You have won!
                                            Do you want to play again? (yes or no)
                                            no
```

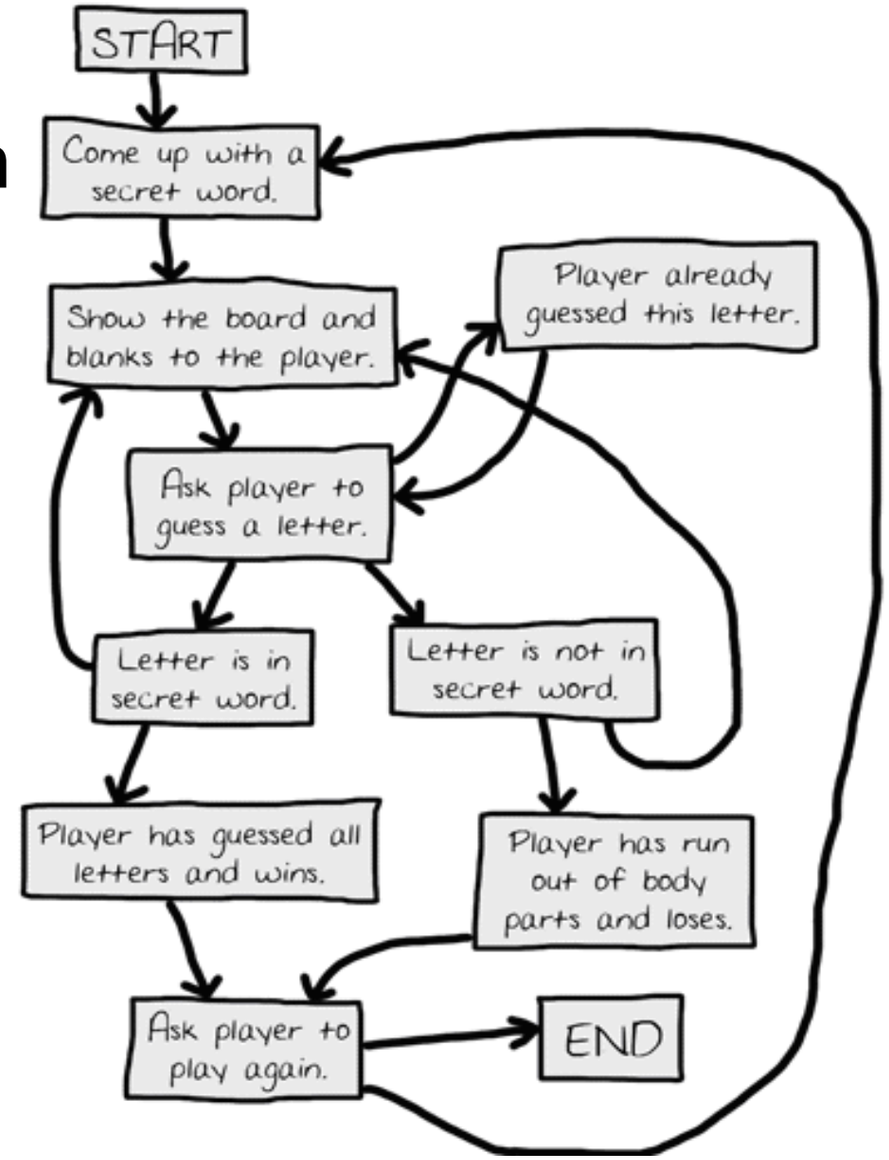
ASCII Art

- ASCII Art
  - Half of the lines of code in the Hangman **aren't really code at all.**
  - Multiline Strings that use **keyboard characters** to draw pictures.
    - **ASCII** stands for **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange



Code Explanation

- Review of the Functions We Defined
  - The complete flow chart of Hangman

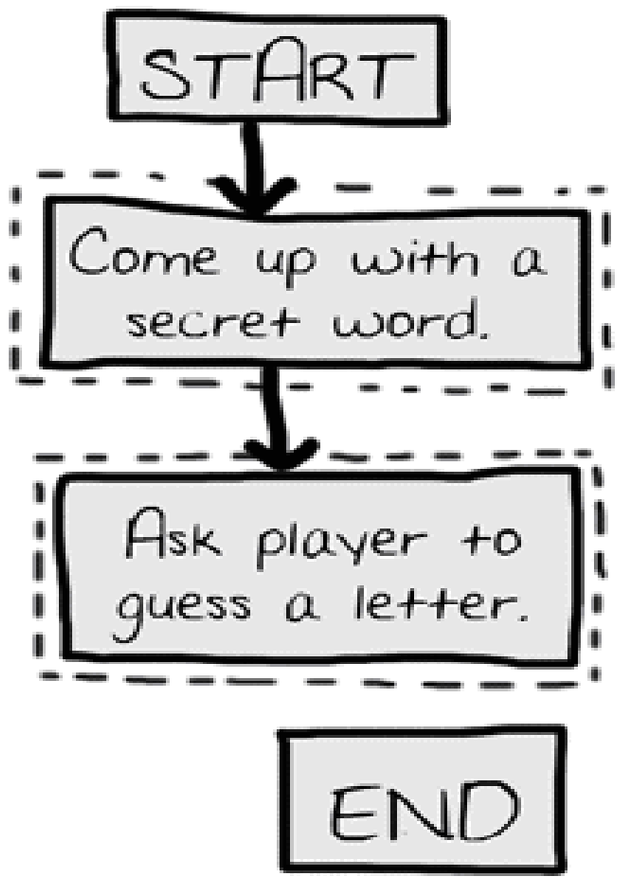


Designing the Program with a Flowchart

- Creating the Flow Chart
  - Create a **flow chart** to help us visualize what this program will do.
  - A flow chart is a diagram that shows a series of steps as a number of boxes connected with arrows.
  - **Begin your flow chart with a Start and End box.**

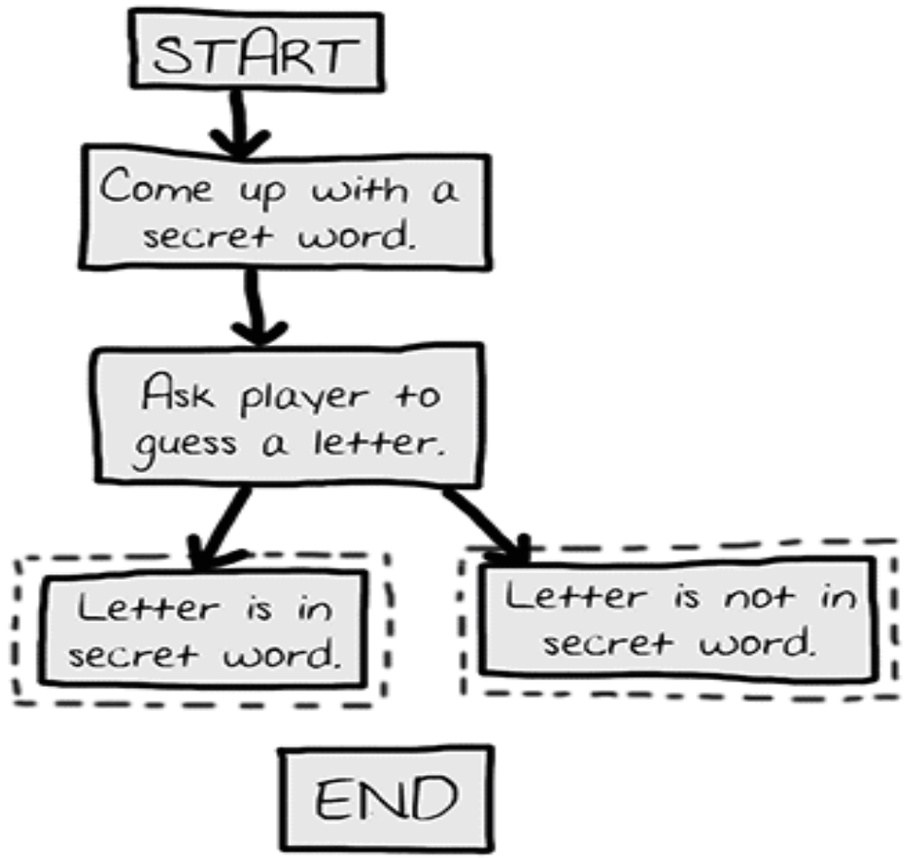
Designing the Program with a Flowchart

- Creating the Flow Chart
  - Draw out the first **two steps** of Hangman as boxes with descriptions.



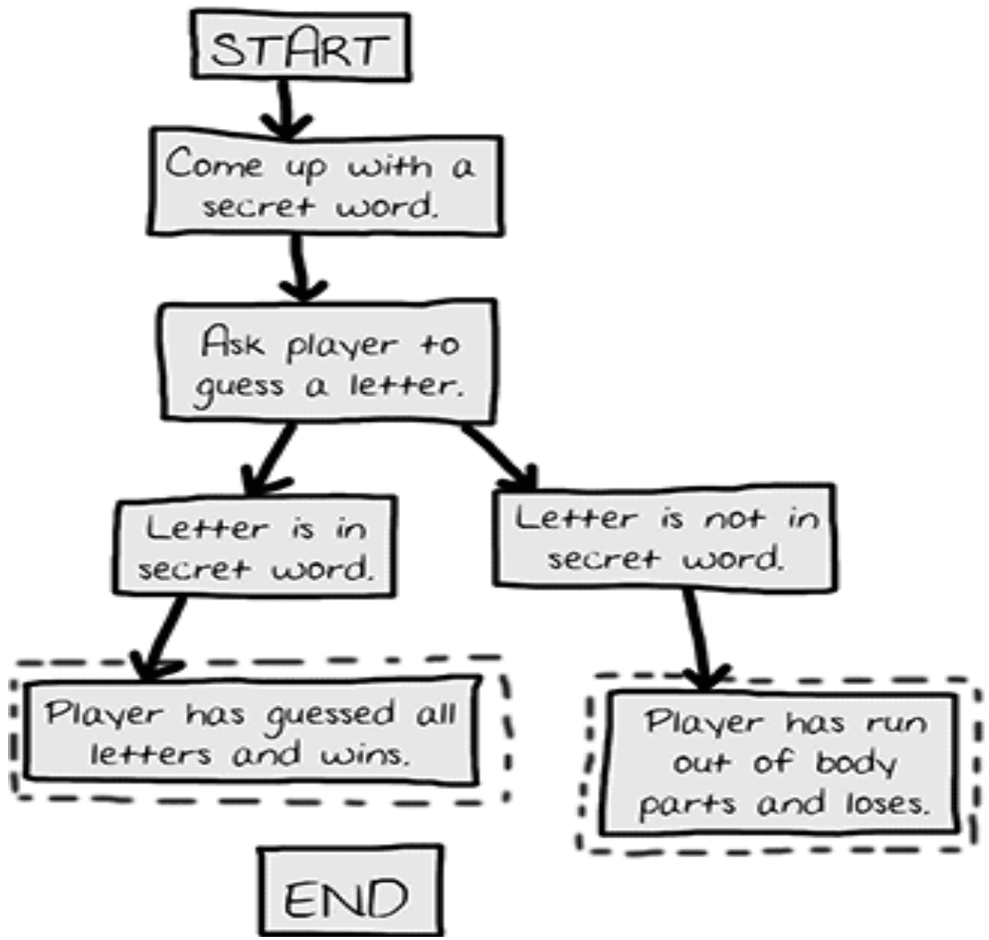
Designing the Program with a Flowchart

- Branching from a Flowchart Box
  - There are **two different things** that could happen after the player guesses, so have two arrows going to separate boxes.



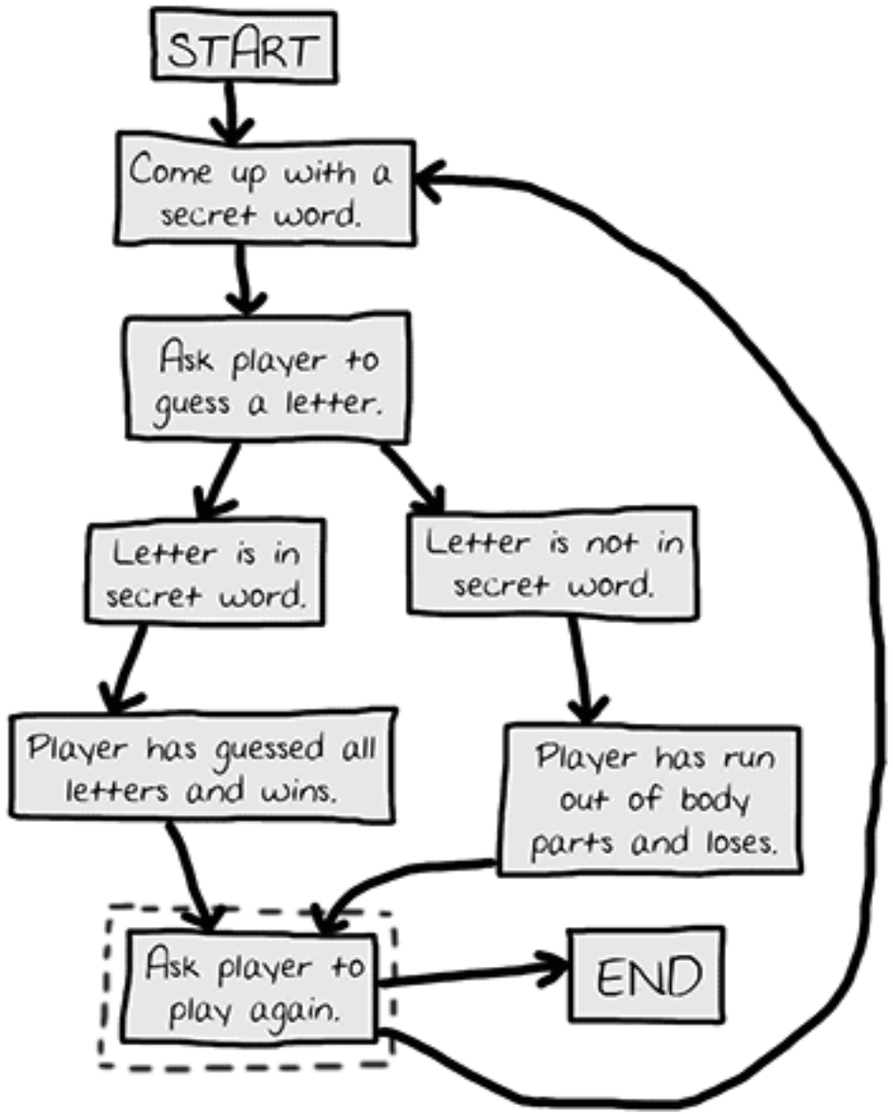
Designing the Program with a Flowchart

- Branching from a Flowchart Box
  - After the branch, the steps continue on their separate paths.



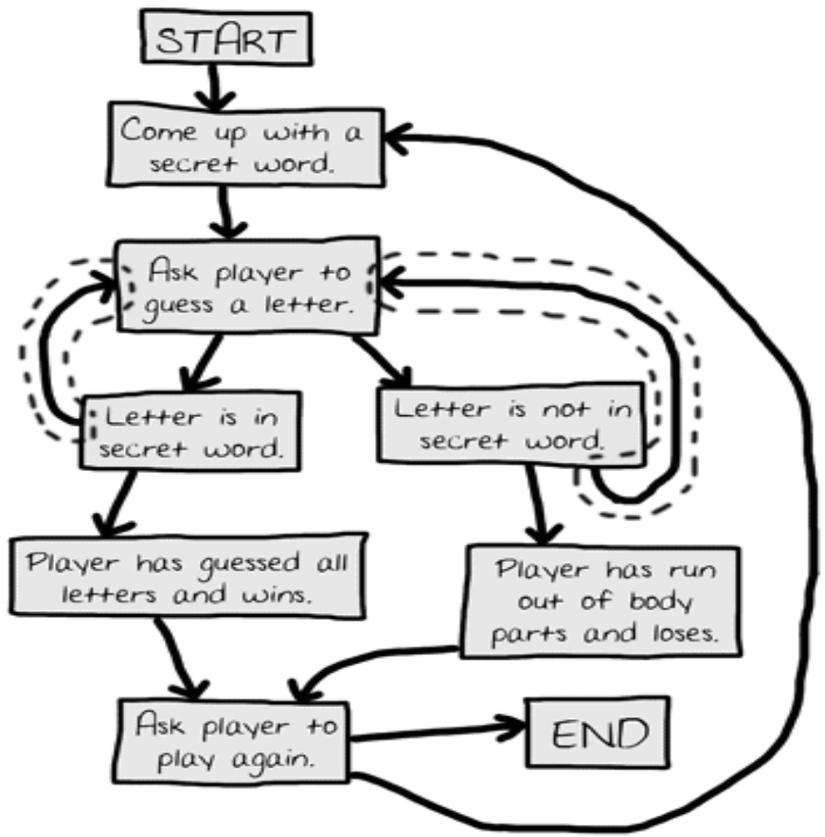
Designing the Program with a Flowchart

- Ending or Restarting the Game
  - The game ends if the player doesn't want to play again, or the game goes back to the beginnir



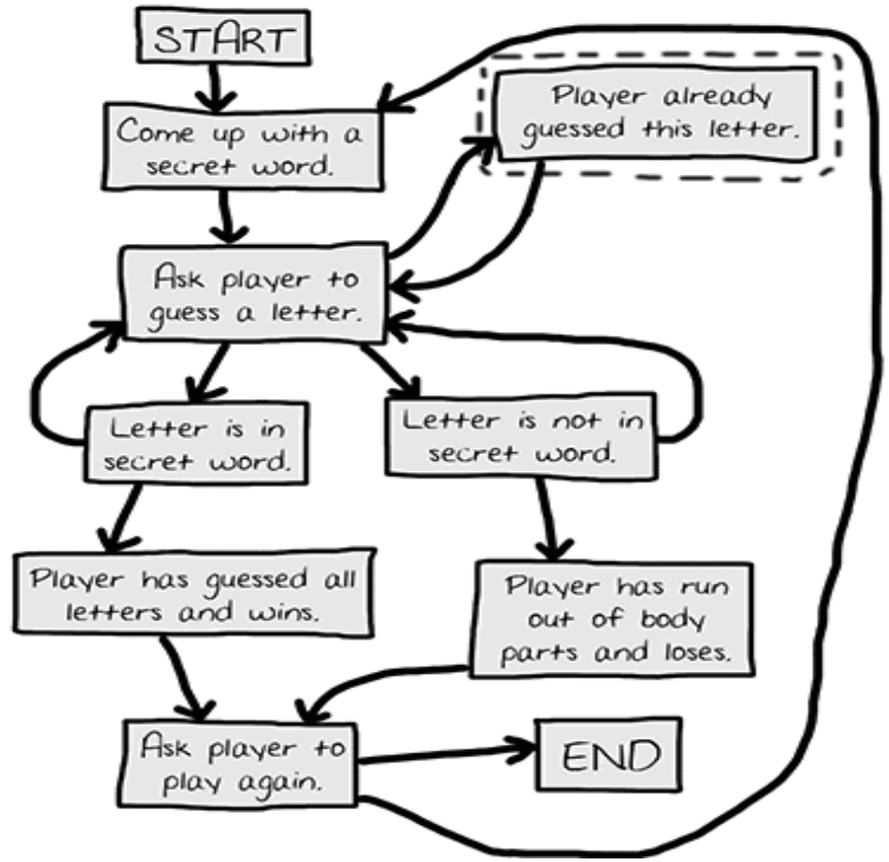
Designing the Program with a Flowchart

- Guessing Again
  - The game does not always end after a guess. The new arrows show that the player can guess again.



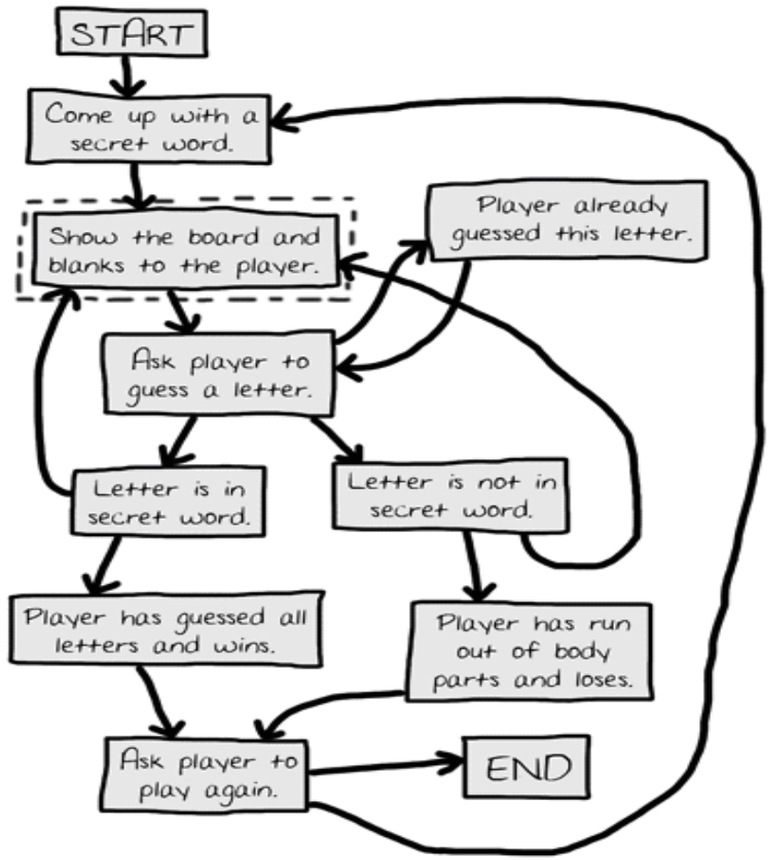
Designing the Program with a Flowchart

- Guessing Again
  - Adding a step in case the player guesses a letter they already guessed.



Designing the Program with a Flowchart

- Offering Feedback to the Player
  - Adding "Show the board and blanks to the player." to give the player feedback.



Things Covered In This Chapter

- **ASCII Art**
- **flow chart**

HANGMAN GAME 2

Invent Your Own Computer Games with Python

Orientation

- “Hangman”
  - Source Code
- Code Explanation
  - Multi-line Strings
  - Constant Variables
  - Lists
  - Lists of Lists

“Hangman”: Source Code(1/4)

```
import random
HANGMANPICS = ['''

   +---+
   |   |
       |
       |
       |
       |
 =========''', '''

   +---+
   |   |
   o   |
       |
       |
       |
 =========''', '''

   +---+
   |   |
   o   |
   |   |
       |
       |
 =========''', '''
```

“Hangman”: Source Code(2/4)

```
  +---+
  |   |
  o   |
 /|   |
      |
      |
=========''', '''

  +---+
  |   |
  o   |
 /|\  |
      |
      |
=========''', '''

  +---+
  |   |
  o   |
 /|\  |
 /    |
      |
=========''', '''

  +---+
  |   |
  o   |
 /|\  |
 / \  |
      |
=========''']
```

“Hangman”: Source Code(3/4)

```
words = 'ant baboon badger bat bear beaver camel cat clam cobra cougar coyote crow deer dog donkey duck eagle ferret fox frog goat goose
hawk lion lizard llama mole monkey moose mouse mule newt otter owl panda parrot pigeon python rabbit ram rat raven rhino salmon seal sha
rk sheep skunk sloth snake spider stork swan tiger toad trout turkey turtle weasel whale wolf wombat zebra'.split()

def getRandomWord(wordList):
    # This function returns a random string from the passed list of strings.
    wordIndex = random.randint(0, len(wordList) - 1)
    return wordList[wordIndex]

def displayBoard(HANGMANPICS, missedLetters, correctLetters, secretWord):
    print(HANGMANPICS[len(missedLetters)])
    print()
    print('Missed letters:', end=' ') # The end=' ' is just to say that you want a space after the end of the statement instead of a new line character.
    for letter in missedLetters:
        print(letter, end=' ')
    print()

    blanks = '_' * len(secretWord)

    for i in range(len(secretWord)): # replace blanks with correctly guessed letters
        if secretWord[i] in correctLetters:
            blanks = blanks[:i] + secretWord[i] + blanks[i+1:]

    for letter in blanks: # show the secret word with spaces in between each letter
        print(letter, end=' ')
    print()

def getGuess(alreadyGuessed):
    # Returns the letter the player entered. This function makes sure the player entered a single letter, and not something else.
    while True:
        print('Guess a letter.')
        guess = input()
        guess = guess.lower()
        if len(guess) != 1:
            print('Please enter a single letter.')
        elif guess in alreadyGuessed: # the argument variable, alreadyGuessed, is used to check if the new guess was entered before.
            print('You have already guessed that letter. Choose again.')
        elif guess not in 'abcdefghijklmnopqrstuvwxyz':
            print('Please enter a LETTER.')
        else:
            return guess # A return statement ends the execution of the function call and "returns" the result
```

“Hangman”: Source Code(4/4)

```
def playAgain():
    # This function returns True if the player wants to play again, otherwise it returns False.
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')
print('H A N G M A N')
missedLetters = ''
correctLetters = ''
secretWord = getRandomWord(words)
gameIsDone = False

while True:
    displayBoard(HANGMANPICS, missedLetters, correctLetters, secretWord)

    # Let the player type in a letter.
    guess = getGuess(missedLetters + correctLetters)

    if guess in secretWord:
        correctLetters = correctLetters + guess

    # Check if the player has won
    foundAllLetters = True
    for i in range(len(secretWord)):
        if secretWord[i] not in correctLetters:
            foundAllLetters = False
            break # The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop. If break
    if foundAllLetters:
        print('Yes! The secret word is "' + secretWord + '"! You have won!') If break statement is in a loop inside another loop), break will terminate the innermost loop.
        gameIsDone = True
else:
    missedLetters = missedLetters + guess

    # Check if player has guessed too many times and lost
    if len(missedLetters) == len(HANGMANPICS) - 1:
        displayBoard(HANGMANPICS, missedLetters, correctLetters, secretWord)
        print('You have run out of guesses!\nAfter ' + str(len(missedLetters)) + ' missed guesses and ' + str(len(correctLetters)) + ' correct guesses, the word was "' + s
        gameIsDone = True                                                                                                                    + secretWord + '"')

# Ask the player if they want to play again (but only if the game is done).
if gameIsDone:
    if playAgain():
        missedLetters = ''
        correctLetters = ''
        gameIsDone = False
        secretWord = getRandomWord(words)
    else:
        break
```

Code Explanation

- Multi-line Strings
  - if you use **three single-quotes**, the string can be on several lines.

```
>>> fizz = '''Dear Alice,
I will return home at the end of the month. I will see you then.
Your friend,
Bob'''
>>> print fizz
Dear Alice,
I will return home at the end of the month. I will see you then.
Your friend,
Bob
```

Code Explanation

- Constant Variables
  - **HANGMANPICS**'s name is in all capitals. This is the programming convention for constant variables.
    - Constants are variables whose values do not change throughout the program. Although you can change the value in HANGMANPICS just like any other variable, the all-caps name reminds you to not do so.

```
>>> DOZEN = 12
>>> eggs = DOZEN * 6
>>> eggs
72
```

Code Explanation

- Lists
  - A list value can contain several other values in it.

```
>>> spam = ['apples', 'oranges', 'HELLO WORLD']
>>> spam
['apples', 'oranges', 'HELLO WORLD']
```

Code Explanation

- Lists
  - The individual values inside of a list are also called **items.**
    - **The square brackets** and **an index** are used to get an item from a list.

```
>>> animals = ['aardvark', 'anteater', 'antelope', 'albert']
>>> animals[0]
'aardvark'
>>> animals[1]
'anteater'
>>> animals[2]
'antelope'
>>> animals[3]
'albert'
```

Code Explanation

- Lists
  - Lists are very good when we have to **store lots of values.**
    - Without Lists, we would have something like this:

```
>>> animals1 = 'aardvark'
>>> animals2 = 'anteater'
>>> animals3 = 'antelope'
>>> animals4 = 'albert'
```

Code Explanation

- Quiz
  - What happens if we enter an index that is larger than the list's largest index?

```
>>> animals = ['aardvark', 'anteater', 'antelope', 'albert']
>>> animals[4]
```

```
>>> animals = ['aardvark', 'anteater', 'antelope', 'albert']
>>> animals[99]
```

Code Explanation

- Changing the Values of List Items with Index Assignment
  - Use the square brackets to **change the value** of an item in a list.
    - overwritten with a new string.

```
>>> animals = ['aardvark', 'anteater', 'antelope', 'albert']
>>> animals[1] = 'ANTEATER'
>>> animals
['aardvark', 'ANTEATER', 'antelope', 'albert']
```

Code Explanation

- Lists
  - **Using the square brackets**
    - the expression animals[0] + animals[2] is the same as 'aardvark' + 'antelope'.

```
>>> animals[0] + animals[2]
'aardvarkantelope'
```

Code Explanation

- List Concatenation

```
>>> [1, 2, 3, 4] + ['apples', 'oranges'] + ['Alice', 'Bob']
[1, 2, 3, 4, 'apples', 'oranges', 'Alice', 'Bob']
```

Code Explanation

- The `in` Operator
  - To see if a value is inside a list or not.
    - **True** if the value is in the list
    - **False** if the value is **not** in the list.

```
>>> animals = ['aardvark', 'anteater', 'antelope', 'albert']
>>> 'antelope' in animals
True
```

Code Explanation

- Quiz

```
>>> animals = ['aardvark', 'anteater', 'antelope', 'albert']
>>> 'antelope' in animals

>>> 'ant' in animals

>>> 'ant' in ['beetle', 'wasp', 'ant']
```

```
>>> 'hello' in 'Alice said hello to Bob.'
```

Code Explanation

- Removing Items from Lists with **`del`** Statements

```
>>> spam = [2, 4, 6, 8, 10]
>>> del spam[1]
>>> spam
[2, 6, 8, 10]
>>> del spam[1]
>>> spam
[2, 8, 10]
>>> del spam[1]
>>> spam
[2, 10]
```

Code Explanation

- Lists of Lists

```
>>> groceries = ['eggs', 'milk', 'soup', 'apples', 'bread']
>>> chores = ['clean', 'mow the lawn', 'go grocery shopping']
>>> favoritePies = ['apple', 'frumbleberry']
>>> listOfLists = [groceries, chores, favoritePies]
>>> listOfLists
[['eggs', 'milk', 'soup', 'apples', 'bread'], ['clean', 'mow
the lawn', 'go grocery shopping'], ['apple', 'frumbleberry']]
```

Code Explanation

- Lists of Lists

```
>>> groceries = ['eggs', 'milk', 'soup', 'apples', 'bread']
>>> chores = ['clean', 'mow the lawn', 'go grocery shopping']
>>> favoritePies = ['apple', 'frumbleberry']
>>> listOfLists = [groceries, chores, favoritePies]
>>> listOfLists
[['eggs', 'milk', 'soup', 'apples', 'bread'], ['clean', 'mow
the lawn', 'go grocery shopping'], ['apple', 'frumbleberry']]
```

- Quiz How to print ‘go grocery shopping’?

  How to change ‘soup’ to ‘chicken soup’?

  How to delete [‘clean’, ‘mow the lawn’, ‘go grocery shopping’]?

“Hangman”

Source Code

```
HANGMANPICS = ['''

  +---+
  |   |
      |
      |
      |
      |
=========''', '''

  +---+
  |   |
  O   |
      |
      |
      |
=========''', '''

  +---+
  |   |
  O   |
  |   |
      |
      |
=========''', '''

  +---+
  |   |
  O   |
 /|   |
      |
      |
=========''', '''

  +---+
  |   |
  O   |
 /|\  |
      |
      |
=========''', '''

  +---+
  |   |
  O   |
 /|\  |
 /    |
      |
=========''', '''

  +---+
  |   |
  O   |
 /|\  |
 / \  |
      |
=========''']
```

Quiz

How to print a particular figure?

Things Covered In This Chapter

- Multi-line Strings
- Lists
- List indexes
- List concatenation
- The in operator
- The del operator
- Lists of Lists

HANGMAN GAME 3

Invent Your Own Computer Games with Python

Orientation

- Code Explanation
  - Methods
  - For Loops
  - Displaying the Secret Word with Blanks

“Hangman”: Source Code(1/4)

```
import random
HANGMANPICS = ['''

   +---+
   |   |
       |
       |
       |
       |
 =========''', '''

   +---+
   |   |
   o   |
       |
       |
       |
 =========''', '''

   +---+
   |   |
   o   |
   |   |
       |
       |
 =========''', '''
```

“Hangman”: Source Code(2/4)

```
  +---+
  |   |
  o   |
 /|   |
      |
      |
=========''', '''

  +---+
  |   |
  o   |
 /|\  |
      |
      |
=========''', '''

  +---+
  |   |
  o   |
 /|\  |
 /    |
      |
=========''', '''

  +---+
  |   |
  o   |
 /|\  |
 / \  |
      |
=========''']
```

“Hangman”: Source Code(3/4)

```
words = 'ant baboon badger bat bear beaver camel cat clam cobra cougar coyote crow deer dog donkey duck eagle ferret fox frog goat goose
hawk lion lizard llama mole monkey moose mouse mule newt otter owl panda parrot pigeon python rabbit ram rat raven rhino salmon seal sha
rk sheep skunk sloth snake spider stork swan tiger toad trout turkey turtle weasel whale wolf wombat zebra'.split()

def getRandomWord(wordList):
    # This function returns a random string from the passed list of strings.
    wordIndex = random.randint(0, len(wordList) - 1)
    return wordList[wordIndex]

def displayBoard(HANGMANPICS, missedLetters, correctLetters, secretWord):
    print(HANGMANPICS[len(missedLetters)])
    print()
    print('Missed letters:', end=' ') # The end=' ' is just to say that you want a space after the end of the statement instead of a new line character.
    for letter in missedLetters:
        print(letter, end=' ')
    print()

    blanks = '_' * len(secretWord)

    for i in range(len(secretWord)): # replace blanks with correctly guessed letters
        if secretWord[i] in correctLetters:
            blanks = blanks[:i] + secretWord[i] + blanks[i+1:]

    for letter in blanks: # show the secret word with spaces in between each letter
        print(letter, end=' ')
    print()

def getGuess(alreadyGuessed):
    # Returns the letter the player entered. This function makes sure the player entered a single letter, and not something else.
    while True:
        print('Guess a letter.')
        guess = input()
        guess = guess.lower()
        if len(guess) != 1:
            print('Please enter a single letter.')
        elif guess in alreadyGuessed: # the argument variable, alreadyGuessed, is used to check if the new guess was entered before.
            print('You have already guessed that letter. Choose again.')
        elif guess not in 'abcdefghijklmnopqrstuvwxyz':
            print('Please enter a LETTER.')
        else:
            return guess # A return statement ends the execution of the function call and "returns" the result
```

“Hangman”: Source Code(4/4)

```
def playAgain():
    # This function returns True if the player wants to play again, otherwise it returns False.
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')
print('H A N G M A N')
missedLetters = ''
correctLetters = ''
secretWord = getRandomWord(words)
gameIsDone = False

while True:
    displayBoard(HANGMANPICS, missedLetters, correctLetters, secretWord)

    # Let the player type in a letter.
    guess = getGuess(missedLetters + correctLetters)

    if guess in secretWord:
        correctLetters = correctLetters + guess

    # Check if the player has won
    foundAllLetters = True
    for i in range(len(secretWord)):
        if secretWord[i] not in correctLetters:
            foundAllLetters = False
            break # The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop. If break
    if foundAllLetters:
        print('Yes! The secret word is "' + secretWord + '"! You have won!') If break statement is in a loop inside another loop), break will terminate the innermost loop.
        gameIsDone = True
else:
    missedLetters = missedLetters + guess

    # Check if player has guessed too many times and lost
    if len(missedLetters) == len(HANGMANPICS) - 1:
        displayBoard(HANGMANPICS, missedLetters, correctLetters, secretWord)
        print('You have run out of guesses!\nAfter ' + str(len(missedLetters)) + ' missed guesses and ' + str(len(correctLetters)) + ' correct guesses, the word was "' + s
        gameIsDone = True                                                                                                                    + secretWord + '"')

# Ask the player if they want to play again (but only if the game is done).
if gameIsDone:
    if playAgain():
        missedLetters = ''
        correctLetters = ''
        gameIsDone = False
        secretWord = getRandomWord(words)
    else:
        break
```

Code Explanation

- Methods
  - Methods are like functions, but they are attached to a value.
  - **The `lower()` and `upper()` String Methods**

```
>>> 'Hello world'.lower()
'hello world'
>>> 'Hello world'.upper()
'HELLO WORLD'
```

  - Can call a string method on that variable.

```
>>> fizz = 'Hello world'
>>> fizz.upper()
'HELLO WORLD'
```

Code Explanation

- Quiz

```
>>> 'Hello world'.upper().lower()
```

```
>>> 'Hello world'.lower().upper()
```

Code Explanation

- Methods
  - **The `reverse()` List Method**
    - reverse the order of the items in the list.

```
>>> spam = [1, 2, 3, 4, 5, 6, 'meow', 'woof']
>>> spam.reverse()
>>> spam
['woof', 'meow', 6, 5, 4, 3, 2, 1]
```

Code Explanation

- Methods
  - **The `append()` List Method**
    - add the value you pass as an argument to the end of the list.

```
>>> eggs = []
>>> eggs.append('hovercraft')
>>> eggs
['hovercraft']
>>> eggs.append('eels')
>>> eggs
['hovercraft', 'eels']
>>> eggs.append(42)
>>> eggs
['hovercraft', 'eels', 42]
```

Code Explanation

- Methods
  - **The `split()` List Method**
    - The `split()` method changes this long string into a list, with each word making up **a single list item.**

```
words = 'ant baboon badger bat bear beaver camel cat
 clam cobra cougar coyote crow deer dog donkey duck
eagle ferret fox frog goat goose hawk lion lizard ll
ama mole monkey moose mouse mule newt otter owl pand
a parrot pigeon python rabbit ram rat raven rhino sa
lmon seal shark sheep skunk sloth snake spider stork
 swan tiger toad trout turkey turtle weasel whale wo
lf wombat zebra'.split()
```

Code Explanation

- Methods
  - **The `split()` List Method**
    - For an example of how the `split()` string method works.

```
>>> 'My very energetic mother just served us nine pies'.split()
['My', 'very', 'energetic', 'mother', 'just', 'served', 'us', '
nine', 'pies']
```

“Hangman”: Source Code(3/4)

```
words = 'ant baboon badger bat bear beaver camel cat clam cobra cougar coyote crow deer dog donkey duck eagle ferret fox frog goat goose
hawk lion lizard llama mole monkey moose mouse mule newt otter owl panda parrot pigeon python rabbit ram rat raven rhino salmon seal sha
rk sheep skunk sloth snake spider stork swan tiger toad trout turkey turtle weasel whale wolf wombat zebra'.split()

def getRandomWord(wordList):
    # This function returns a random string from the passed list of strings.
    wordIndex = random.randint(0, len(wordList) - 1)
    return wordList[wordIndex]

def displayBoard(HANGMANPICS, missedLetters, correctLetters, secretWord):
    print(HANGMANPICS[len(missedLetters)])
    print()
    print('Missed letters:', end=' ') # The end=' ' is just to say that you want a space after the end of the statement instead of a new line character.
    for letter in missedLetters:
        print(letter, end=' ')
    print()

    blanks = '_' * len(secretWord)

    for i in range(len(secretWord)): # replace blanks with correctly guessed letters
        if secretWord[i] in correctLetters:
            blanks = blanks[:i] + secretWord[i] + blanks[i+1:]

    for letter in blanks: # show the secret word with spaces in between each letter
        print(letter, end=' ')
    print()

def getGuess(alreadyGuessed):
    # Returns the letter the player entered. This function makes sure the player entered a single letter, and not something else.
    while True:
        print('Guess a letter.')
        guess = input()
        guess = guess.lower()
        if len(guess) != 1:
            print('Please enter a single letter.')
        elif guess in alreadyGuessed: # the argument variable, alreadyGuessed, is used to check if the new guess was entered before.
            print('You have already guessed that letter. Choose again.')
        elif guess not in 'abcdefghijklmnopqrstuvwxyz':
            print('Please enter a LETTER.')
        else:
            return guess # A return statement ends the execution of the function call and "returns" the result
```

Code Explanation

- The `len()` Function
  - Returns the integer of how many items are in a list.

```
>>> animals = ['aardvark', 'anteater', 'antelope', 'albert']
>>> len(animals)
4
>>> people = ['Alice', 'Bob']
>>> len(people)
2
>>> len(animals) + len(people)
6
```

Code Explanation

- The `len()` Function
  - The square brackets by themselves are also a list value known as the **empty list.**

```
>>> len([])
0
>>> spam = []
>>> len(spam)
0
```

Code Explanation

- The **`getRandomWord()`** Function
  - store a random index for this list in the `wordIndex` variable.
  - do this by calling `randint()` with two arguments.
    - The reason we need the **- 1** is because the **indexes for lists start at 0.**

```
def getRandomWord(wordList):
    # This function returns a random string from the
passed list of strings.
    wordIndex = random.randint(0, len(wordList) - 1)
    return wordList[wordIndex]
```

“Hangman”

- Sample Run

```
H A N G M A N

  +---+
  |   |
      |
      |
      |
      |
=========
Missed letters:
_ _ _
Guess a letter.
a

  +---+
  |   |
  O   |
      |
      |
      |
=========
Missed letters: o
_ a _
Guess a letter.
r

  +---+
  |   |
  O   |
  |   |
      |
      |
=========
Missed letters: or
_ a _
Guess a letter.
t

  +---+
  |   |
  O   |
 /|   |
      |
      |
=========
Missed letters: or
_ a t
Guess a letter.
a
You have already guessed that letter. Choose again.
Guess a letter.
c
Yes! The secret word is "cat"! You have won!
Do you want to play again? (yes or no)
no
```

Code Explanation

- The **`displayBoard()`** Function
  - This function has four parameters.

```
def displayBoard(HANGMANPICS, missedLetters, correctLetters, secretWord):
    print(HANGMANPICS[len(missedLetters)])
    print()
```

`HANGMANPICS`	a list of multi-line strings that will display the board as ASCII art
`missedLetters`	a string made up of the letters the player has guessed that are not in the secret word.
`correctLetters`	a string made up of the letters the player has guessed that are in the secret word.
`secretWord`	the secret word that the player is trying to guess.

Code Explanation

- The **`range()`** Function
  - When called with one argument,
    - `range()` will return a range object of integers from 0 up to the argument.

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(10000))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,...
      ...The text here has been skipped for brevity...
...9989, 9990, 9991, 9992, 9993, 9994, 9995, 9996, 9997,
9998, 9999]
```

Code Explanation

- The `range()` Function
  - The list is so huge, that it won't even all fit onto the screen.
    - But we can save the list into the variable just like any other list by entering this.

```
>>> spam = list(range(10000))
```

- If you pass **two arguments** to range(),
  - the list of integers it returns is from the first argument up to the second argument.

```
>>> list(range(10, 20))
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

Code Explanation

- Strings Act Like Lists
  - Just think of strings as "list" of one-letter strings.

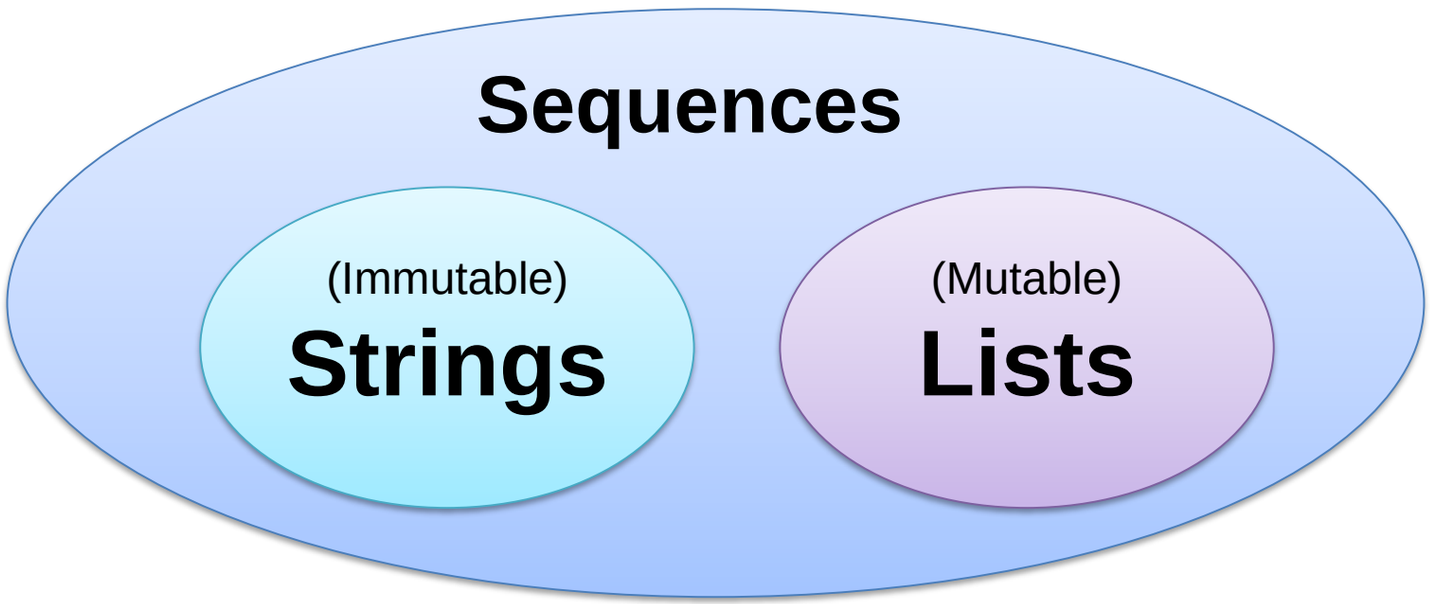
```
>>> fizz = 'Hello world!'
>>> fizz[0]
'H'
```

  - You can also find out how many characters are in a string with the `len()` function.

```
>>> fizz = 'Hello world!'
>>> fizz[0]
'H'
>>> len(fizz)
12
```

Code Explanation

- Strings Act Like Lists
  - You cannot change a character in a string or remove a character with `del` statement.
    - **List: mutable sequence** (changeable)
    - **String: immutable sequence** (cannot be changed)



Code Explanation

- **`for` Loops**
  - The `for` loop is very good at looping over a list of values.
  - begins with the **`for`** keyword, followed by a variable name, the **`in`** keyword, a sequence or a range object, and then a colon.
  - Each time the program execution goes through the loop (on each **iteration** through the loop)

```
>>> for i in range(10):
        print(i)
```

Code Explanation

- `for` Loops
  - For example

```
>>> for i in range(10):
        print(i)

0
1
2
3
4
5
6
7
8
9
```

Code Explanation

- **`for` Loops**
  - we used the for statement with the **list** instead of `range()`.

```
>>> for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:
        print(i)

0
1
2
3
4
5
6
7
8
9
```

Code Explanation

- Quiz

```
for thing in ['cats', 'pasta', 'programming', 'spam']:
    print('I really like ' + thing)
```

Code Explanation

- `for` Loops
  - uses a single character from the string on each iteration.

```
>>> for i in 'Hello world!':
        print(i)

H
e
l
l
o

w
o
r
l
d
!
```

“Hangman”

- Sample Run

```
H A N G M A N

  +---+
  |   |
      |
      |
      |
      |
=========
Missed letters:
_ _ _
Guess a letter.
a

  +---+
  |   |
  O   |
      |
      |
      |
=========
Missed letters: o
_ a _
Guess a letter.
r

  +---+
  |   |
  O   |
  |   |
      |
      |
=========
Missed letters: or
_ a _
Guess a letter.
t

  +---+
  |   |
  O   |
 /|   |
      |
      |
=========
Missed letters: or
_ a t
Guess a letter.
a
You have already guessed that letter. Choose again.
Guess a letter.
c
Yes! The secret word is "cat"! You have won!
Do you want to play again? (yes or no)
no
```

Code Explanation

- **`for` Loop**
  - This `for` loop will display all the missed guesses that the player has made.
  - If `missedLetters` was `'ajtw`, then this for loop would display `a j t w`.

```
print('Missed letters:', end=' ')   # The end=' ' is just to say that
for letter in missedLetters:          you want a space after the end
    print(letter, end=' ')          of the statement instead of
print()                             a new line character.
```

Code Explanation

- A **`while`** Loop Equivalent of a **`for`** Loop
  - You can make a `while` loop that acts the same way as a `for` loop by adding extra code.

```
>>> sequence = ['cats', 'pasta', 'programming', 'spam']
>>> index = 0
>>> while (index < len(sequence)):
        thing = sequence[index]
        print('I really like ' + thing)
        index = index + 1

I really like cats
I really like pasta
I really like programming
I really like spam
```

Code Explanation

- List Slicing and Substrings
  - **Slicing**
    - Indexing with multiple indexes instead of just one.
    - Put two indexes separated by a colon.
    - Can use slicing to get a part of a string(called a **substring** from a string.)

```
>>> animals = ['aardvark', 'anteater', 'antelope', 'albert']
>>> animals[0:3]
['aardvark', 'anteater', 'antelope']
>>> animals[2:4]
['antelope', 'albert']
```

Code Explanation

- Quiz

```
>>> animals = ['aardvark', 'anteater', 'antelope', 'albert']
>>> animals[0:0]
```

```
>>> 'Hello world!'[3:8]
```

“Hangman”

- Sample Run

```
H A N G M A N

  +---+
  |   |
      |
      |
      |
      |
=========
Missed letters:
_ _ _
Guess a letter.
a

  +---+
  |   |
  O   |
      |
      |
      |
=========
Missed letters: o
_ a _
Guess a letter.
r

  +---+
  |   |
  O   |
  |   |
      |
      |
=========
Missed letters: or
_ a _
Guess a letter.
t

  +---+
  |   |
  O   |
 /|   |
      |
      |
=========
Missed letters: or
_ a t
Guess a letter.
a
You have already guessed that letter. Choose again.
Guess a letter.
c
Yes! The secret word is "cat"! You have won!
Do you want to play again? (yes or no)
no
```

Code Explanation

- Displaying the Secret Word with Blanks
  - We can use the _ **character** (called the underscore character) for this.

secret word	blanked string
otter	_____ (five _ characters)

correctLetters	blanked string
rt	_tt_r

“Hangman”: Source Code(3/4)

```
words = 'ant baboon badger bat bear beaver camel cat clam cobra cougar coyote crow deer dog donkey duck eagle ferret fox frog goat goose
hawk lion lizard llama mole monkey moose mouse mule newt otter owl panda parrot pigeon python rabbit ram rat raven rhino salmon seal sha
rk sheep skunk sloth snake spider stork swan tiger toad trout turkey turtle weasel whale wolf wombat zebra'.split()

def getRandomWord(wordList):
    # This function returns a random string from the passed list of strings.
    wordIndex = random.randint(0, len(wordList) - 1)
    return wordList[wordIndex]

def displayBoard(HANGMANPICS, missedLetters, correctLetters, secretWord):
    print(HANGMANPICS[len(missedLetters)])
    print()
    print('Missed letters:', end=' ') # The end=' ' is just to say that you want a space after the end of the statement instead of a new line character.
    for letter in missedLetters:
        print(letter, end=' ')
    print()

    blanks = '_' * len(secretWord)

    for i in range(len(secretWord)): # replace blanks with correctly guessed letters
        if secretWord[i] in correctLetters:
            blanks = blanks[:i] + secretWord[i] + blanks[i+1:]

    for letter in blanks: # show the secret word with spaces in between each letter
        print(letter, end=' ')
    print()

def getGuess(alreadyGuessed):
    # Returns the letter the player entered. This function makes sure the player entered a single letter, and not something else.
    while True:
        print('Guess a letter.')
        guess = input()
        guess = guess.lower()
        if len(guess) != 1:
            print('Please enter a single letter.')
        elif guess in alreadyGuessed: # the argument variable, alreadyGuessed, is used to check if the new guess was entered before.
            print('You have already guessed that letter. Choose again.')
        elif guess not in 'abcdefghijklmnopqrstuvwxyz':
            print('Please enter a LETTER.')
        else:
            return guess # A return statement ends the execution of the function call and "returns" the result
```

Code Explanation

- Displaying the Secret Word with Blanks
  - * operator can also be used on a string and an integer.
    - so the expression `'hello' * 3` evaluates to `'hellohellohello'`
  - This will make sure that `blanks` has the same number of underscores as `secretWord` has letters.

```
blanks = '_' * len(secretWord)

for i in range(len(secretWord)): # replace blanks with correctly guessed letters
    if secretWord[i] in correctLetters:
        blanks = blanks[:i] + secretWord[i] + blanks[i+1:]

for letter in blanks: # show the secret word with spaces in between each letter
    print(letter, end=' ')
print()
```

Code Explanation

- Replacing the Underscores with Correctly Guessed Letters
  - Let's pretend
    - the value of `secretWord` is **`'otter'`**
    - the value in `correctLetters` is **`'tr'`**
  - Then `len(secretWord)` will return `5`.
  - Then `range(len(secretWord))` becomes `range(5)`, which in turn returns the list `[0, 1, 2, 3, 4]`.

```
for i in range(len(secretWord)): # replace blanks with correctly guessed letters
    if secretWord[i] in correctLetters:
        blanks = blanks[:i] + secretWord[i] + blanks[i+1:]
```

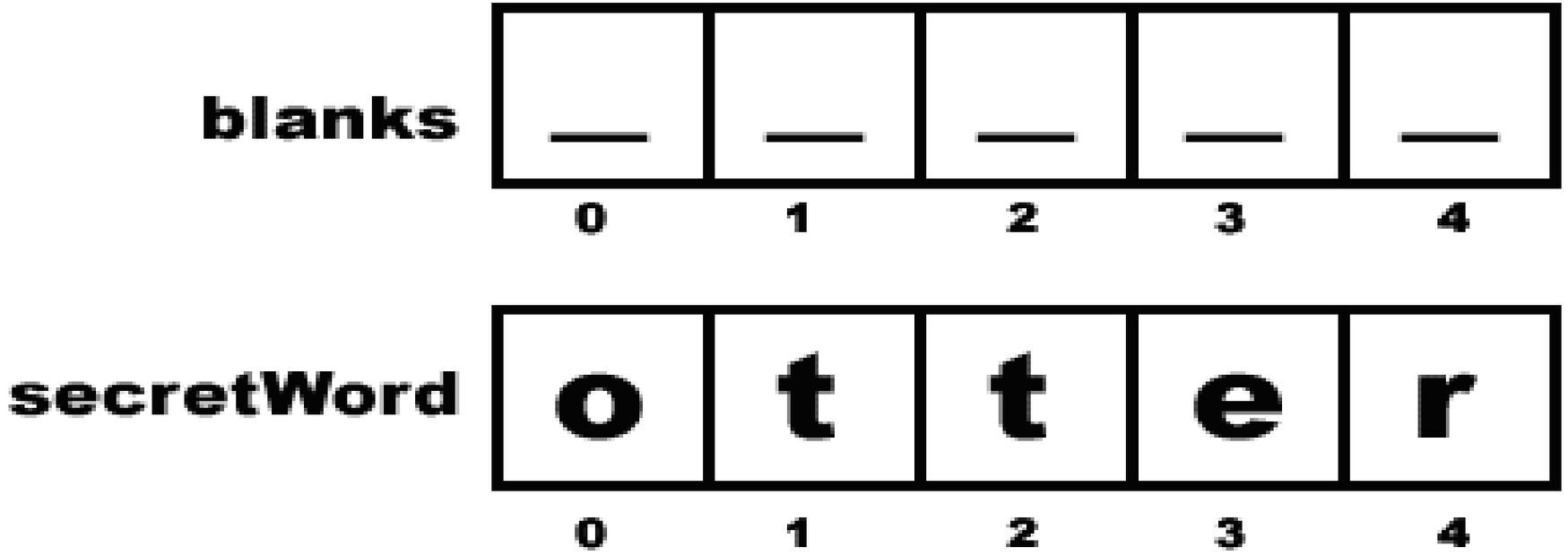
Code Explanation

- Replacing the Underscores with Correctly Guessed Letters
  - The value of `i` will take on each value in `[0, 1, 2, 3, 4]`
    - then the **`for` loop** code is equivalent to this (called **loop unrolling**).

```
if secretWord[0] in correctLetters:
blanks = blanks[:0] + secretWord[0] + blanks[1:]
if secretWord[1] in correctLetters:
blanks = blanks[:1] + secretWord[1] + blanks[2:]
if secretWord[2] in correctLetters:
blanks = blanks[:2] + secretWord[2] + blanks[3:]
if secretWord[3] in correctLetters:
blanks = blanks[:3] + secretWord[3] + blanks[4:]
if secretWord[4] in correctLetters:
blanks = blanks[:4] + secretWord[4] + blanks[5:]
```

Code Explanation

- Replacing the Underscores with Correctly Guessed Letters
  - It shows the value of the `secretWord` and `blanks` variables.
    - the index for each letter in the string.



Code Explanation

- Replacing the Underscores with Correctly Guessed Letters
  - The **unrolled loop** code would be the same as this.

```
if 'o' in 'tr': # False, blanks == '_____'
    blanks = '' + 'o' + '____' # This line is skipped.
if 't' in 'tr': # True, blanks == '_____'
    blanks = '_' + 't' + '___' # This line is executed.
if 't' in 'tr': # True, blanks == '_t___'
    blanks = '_t' + 't' + '__' # This line is executed.
if 'e' in 'tr': # False, blanks == '_tt__'
    blanks = '_tt' + 'e' + '_' # This line is skipped.
if 'r' in 'tr': # True, blanks == '_tt__'
    blanks = '_tt_' + 'r' + '' # This line is executed.
# blanks now has the value '_tt_r'
```

Code Explanation

- Replacing the Underscores with Correctly Guessed Letters
  - This `for` loop will print out each character in the string `blanks`.
  - Show the secret word with spaces in between each letter

```
    for letter in blanks: # show the secret word with spaces in between each letter
        print(letter, end=' ')
    print()
```

Things Covered In This Chapter

- **Methods**
- **The** lower() **and** upper() **String Method**
- **The** reverse() **and** append() **List Methods**
- **The** split() **List Method**
- **The** range() **and** list() **Functions**
- For **Loops**
- **A** while **Loop Equivalent of a** for **Loop**
- **Slicing**
- **Displaying the Secret Word with Blanks**

Acknowledgement

- This course material was prepared for “Creative Computing for Engineers” in the College of Engineering by Professor Heejin Park and was slightly modified for Python 3.