2/16/2023

Mathematics Association of Nairobi University
isaak@students.uonbi.ac.ke

Control Flow

`Control flow` describes the order in which your lines of code are run. This order is usually different than the sequence in which the lines of code appear! Execution can flow from one place in the code to another

we'll learn about several tools in Python we can use to affect our code's control flow:

- Conditional Statements
- Boolean Expressions
- For and While Loops
- Break and Continue

Indentation

Some other languages use braces to show where blocks of code begin and end. In Python we use indentation to enclose blocks of code. For example, `if` statements use indentation to tell Python what code is inside and outside of different clauses.

In Python, indents conventionally come in multiples of four spaces. Be strict about following this convention, because changing the indentation can completely change the meaning of the code. If you are working on a team of Python programmers, it's important that everyone follows the same indentation convention!

Spaces or Tabs?

The Python Style Guide (https://peps.python.org/pep-0008/#tabs-or-spaces) recommends using 4 spaces to indent, rather than using a tab. Whichever you use, be aware that "Python 3 disallows mixing the use of tabs and spaces for indentation."

If Statement

An if statement is a conditional statement that runs or skips code based on whether a condition is true or false. Here's a simple example.

In [1]:

```python
phone_balance = 20
bank_balance = 1000

if phone_balance < 20:
    phone_balance += 50
    bank_balance -= 50

print("Your phone_balance is: ", phone_balance)
print("Your bank_balance is: ", bank_balance)
```

```
Your phone_balance is:  20
Your bank_balance is:  1000
```

In [2]:

```python
number = 145

if number % 2 == 0: # == is called comparison operators
    print("Number " + str(number) + " is even.")
else:
    print("Number " + str(number) + " is odd.")
```

```
Number 145 is odd.
```

To get inputs from the user we use `input()` function

In [3]:

```python
#Example of how to get input from users
name = input('Please Enter your Name: ')
```

```
print("Hello " + name + " and Welcome to Data Analysis Class")
```

```
Please Enter your Name: Jane Doe
Hello Jane Doe and Welcome to Data Analysis Class
```

In [6]:

```
# A simple program for checking whether a number is odd or even
name = input('Please Enter your Name: ')
number = input("Please Enter Your Number: ")
number = int(number)

if number % 2 == 0:
    print("Hello " + name + "Your Number: " + str(number) + " is even.")
else:
    print("Hello " + name + " Your Number: " + str(number) + " is odd.")
```

```
Please Enter your Name: John Doe
Please Enter Your Number: 123
Hello John Doe Your Number: 123 is odd.
```

If, Elif, Else

In addition to the `if` clause, there are two other optional clauses often used with an `if` statement.

1. `if` : An if statement must always start with an if clause, which contains the first condition that is checked. If this evaluates to True, Python runs the code indented in this if block and then skips to the rest of the code after the if statement.
2. `elif` : elif is short for "else if." An elif clause is used to check for an additional condition if the conditions in the previous clauses in the if statement evaluate to False. As you can see in the example, you can have multiple elif blocks to handle different situations.
3. `else` : Last is the else clause, which must come at the end of an if statement if used. This clause doesn't require a condition. The code in an else block is run if all conditions above that in the if statement evaluate to False.

For example:

In [8]:

```
season = input('What season are we in? ')

if season == 'spring':
    print('plant the garden!')
elif season == 'summer':
    print('water the garden!')
elif season == 'fall':
    print('harvest the garden!')
elif season == 'winter':
    print('stay indoors!')
else:
    print('unrecognized season')
```

```
What season are we in? spring
plant the garden!
```

In [20]:

```
#Building a simple Calculator for +,-, *, /,

num1 = int(input("num1: "))
op = input("Operator: ")
num2 = int(input("num2: "))

if op == "+":
     print(num1 + num2)
elif op == "-":
     print(num1 - num2)
elif op == "/":
     print(num1 / num2)
elif op == "*":
     print(num1 * num2)
else:
     print("Invalid Operator")

# Assignment: Add more operations in the calculator
```

```
num1: 600
Operator: /
num2: 2
300.0
```

Assignment:

Quiz: Guess My Number

You decide you want to play a game where you are hiding a number from someone. Store this number in a variable called 'answer'. Another user provides a number called 'guess'. By comparing 'guess' to 'answer', you inform the user if their guess is too high or too low.

Fill in the conditionals below to inform the user about how their guess compares to the answer.

In [12]:
```
# '''
# You decide you want to play a game where you are hiding
# a number from someone.  Store this number in a variable
# called 'answer'.  Another user provides a number called
# 'guess'.  By comparing guess to answer, you inform the user
# if their guess is too high or too low.

# Fill in the conditionals below to inform the user about how
# their guess compares to the answer.
# '''
answer = 19 #provide answer
guess =  input('Please Enter Your Guess')#provide guess
guess = int(guess)

if                              #provide conditional
    result = "Oops!  Your guess was too low."
elif                            #provide conditional
    result = "Oops!  Your guess was too high."
elif                            #provide conditional
    result = "Nice!  Your guess matched the answer!"

print(result)
```

```
Please Enter Your Guess19
Nice!  Your guess matched the answer!
```

For Loops

Python has two kinds of loops - `for loops` and `while loops`. A `for` loop is used to "iterate", or do something repeatedly, over an iterable.

An iterable is an object that can return one of its elements at a time. This can include sequence types, such as strings, lists, and tuples, as well as non-sequence types, such as dictionaries and files.

Example Let's break down the components of a `for loop`, using this example with the list `cities`:

In [13]:
```
cities = ['new york city', 'mountain view', 'chicago', 'los angeles']
for city in cities:
    print(city)
print("Done!")
```

```
new york city
mountain view
chicago
los angeles
Done!
```

Components of a for Loop

1. The first line of the loop starts with the `for` keyword, which signals that this is a `for` loop
2. Following that is `city in cities`, indicating city is the `iteration variable`, and `cities` is the `iterable` being looped over. In the first iteration of the loop, `city` gets the value of the first element in cities, which is “`new york city`”.
3. The `for` loop heading line always ends with a colon `:`
4. Following the `for` loop heading is an indented block of code, the body of the loop, to be executed in each iteration of this loop. There is only one line in the body of this loop - `print(city)`.
5. After the body of the loop has executed, we don't move on to the next line yet; we go back to the for heading line, where the iteration variable takes the value of the next element of the iterable. In the second iteration of the loop above, `city` takes the value of the next element in `cities`, which is "mountain view".
6. This process repeats until the loop has iterated through all the elements of the `iterable`. Then, we move on to the line that follows the body of the loop - in this case, `print("Done!")`. We can tell what the next line after the body of the loop is because it is unindented. Here is another reason why paying attention to your indentation is very important in Python! Executing the code in the example above produces this output:

Using the `range()` Function with for Loops

`range()` is a built-in function used to create an iterable sequence of numbers. You will frequently use `range()` with a `for` loop to repeat an action a certain number of times. Any variable can be used to iterate through the numbers, but Python programmers conventionally use `i`, as in this example:

In [14]:
```
for i in range(3):
    print("Hello!")
```

```
Hello!
Hello!
Hello!
```

range(start=0, stop, step=1) The range() function takes three integer arguments, the first and third of which are optional:

- The 'start' argument is the first number of the sequence. If unspecified, 'start' defaults to 0.
- The 'stop' argument is 1 more than the last number of the sequence. This argument must be specified.
- The 'step' argument is the difference between each number in the sequence. If unspecified, 'step' defaults to 1.

Notes on using `range():`

- If you specify one integer inside the parentheses withrange(), it's used as the value for 'stop,' and the defaults are used for the other two. e.g. - `range(4)` returns `0, 1, 2, 3`
- If you specify two integers inside the parentheses withrange(), they're used for 'start' and 'stop,' and the default is used for 'step.' e.g. - `range(2, 6)` returns `2, 3, 4, 5`
- Or you can specify all three integers for 'start', 'stop', and 'step.' e.g. - `range(1, 10, 2)` returns `1, 3, 5, 7, 9`

Creating and Modifying Lists

In addition to extracting information from lists, as we did in the first example above, you can also create and modify lists with `for` loops. You can `create` a list by appending to a new list at each iteration of the `for` loop like this:

In [15]:
```python
# Creating a new list
cities = ['new york city', 'mountain view', 'chicago', 'los angeles']
capitalized_cities = []

for city in cities:
    capitalized_cities.append(city.title())
```

In [16]:
```python
print(capitalized_cities)
```

```
['New York City', 'Mountain View', 'Chicago', 'Los Angeles']
```

While Loops

`For` loops are an example of "`definite iteration`" meaning that the loop's body is run a predefined number of times. This differs from "indefinite iteration" which is when a loop repeats an unknown number of times and ends when some condition is met, which is what happens in a while loop. Here's an example of a while loop.

In [23]:
```python
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

Print a message once the condition is false:

In [24]:
```python
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

```
1
2
3
4
5
i is no longer less than 6
```

Question: What type of loop should we use?

You need to write a loop that takes the numbers in a given list named num_list: `num_list = [422, 136, 524, 85, 96, 719, 85, 92, 10, 17, 312, 542, 87, 23, 86, 191, 116, 35, 173, 45, 149, 59, 84, 69, 113, 166]`

Your code should add up the odd numbers in the list, but only up to the first 5 odd numbers together. If there are more than 5 odd numbers, you should stop at the fifth. If there are fewer than 5 odd numbers, add all of the odd numbers.

use a `while` or a `for` loop to write this code?

We would write a while loop to write this code for the following reasons:

1. We don't need a break statement that a for loop will require. Without a break statement, a for loop will iterate through the whole list, which is not efficient.
2. We don't want to iterate over the entire list, but only over the required number of elements in the list that meets our condition.
3. It is easier to understand because you explicitly control the exit conditions for the loop.

In [25]:
```
num_list = [422, 136, 524, 85, 96, 719, 85, 92, 10, 17, 312, 542, 87, 23, 86, 191, 116, 35, 173, 45, 149, 59, 8

count_odd = 0
list_sum = 0
i = 0
len_num_list = len(num_list)

while (count_odd < 5) and (i < len_num_list):
    if num_list[i] % 2 != 0:
        list_sum += num_list[i]
        count_odd += 1
    i += 1

print ("The numbers of odd numbers added are: {}".format(count_odd))
print ("The sum of the odd numbers added is: {}".format(list_sum))
```

```
The numbers of odd numbers added are: 5
The sum of the odd numbers added is: 993
```

Projects

RE-DO THE FOLLOWING PROJECTS

DON'T COPY PASTE THE CODE!

In [ ]:
```
#Password Generator Application
import random

numbers = list()

for i in range(40): numbers.append(str(i))

while True:
    ans = input("How many password do you want?:")

    if ans not in numbers: print("Your integer must be between 8 and 40. Please enter a valid number.")
    else: break

while True:
    leng = input("How long do you want your password to be?:")

    if leng not in numbers: print("Your integer must be between 8 and 40. Please enter a valid number.")
    else: break

for i in range(int(ans)):
    pas = list()

    C = random.sample("ABCDEFGHIJKLMNOPQRSTUVWXYZ", int(leng)//4)
    CH = random.sample("!'#+%&/*?-_@,.:;", int(leng) // 4)
    NUM = random.sample("0123456789", int(leng) // 4)
    SML = random.sample("abcdefghijklmnopqrstuvwxyz", int(leng)-int(leng)//4*3)

    for i in C: pas+= i
    for i in CH: pas+= i
    for i in NUM: pas+= i
    for i in SML: pas+= i

    random.shuffle(pas)
    password = "".join(pas)
    print (password)

input("Press enter to exit")
```

In [ ]:
```
# Guess game application!
import random
interval=["1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22",
          "25","26","27","28","29","30","31","32","33","34","35","36","37","38","39","40","41","42","43","44","
          "47","48","49","50","51","52","53","54","55","56","57","58","60","61","62","63","64","65","66","67","
          "70","71","72","73","74","75","76","77","78","79","80","81","82","83","84","85","86","87","88","89","
          "92","93","94","95","96","97","98","99"]
print ("!!!Welcome to the most exquisite game of all time :)!!!")
name = input("Please enter your name:")
while True:
    if name=="":
        print ("Name cant be blank. Please type your name.")
        name = input("Please enter your name:")
    else:
        break
print ("Welcome" + " " + name + ". " + "You're in a virtual reality guess game. Here how this game works. ")
print ("I kept a number in my mind which in between 0 and 100. You'll try to figure out the number by guessing.
print ("When you make your guess i'll tell you if you are far away or close. ")
print ("Lets begin. Good Luck :)")
def again():
    while True:
        choice = input("You wanna play again?:")
        if choice == "yes":
            Guess()
        elif choice == "no":
            print ("Ok. It was nice seing you. I hope we'll meet again. Stay safe by then")
            break
        else:
            print ("I dont know what that is. Please type yes or no")
def Guess():
    number = random.randint(1,100)
    answer = input("Your guess is:")
    guess=7
    while guess<8 and guess>1:
        while True:
            if answer not in interval:
                print ("Your entry is  not valid. Please type a number between 0 and 100")
                answer = input("Your guess is:")
            else:
                break
        if answer==str(number):

            print("woooow are you a kahin :)")
            again()
        elif answer>str(number):

            print("You went too far try a smaller number")
            guess -= 1

            print("Now you're left with" + " " + str(guess) + " " + "number of guesses. Choose wisely.")
            answer = input("Your guess is:")
        else:

            print("Is this your best guess? Try bigger numbers")
            guess -= 1
            print("Now you're left with" + " " + str(guess) + " " + "number of guesses. Choose wisely.")
            answer = input("Your guess is:")
    if guess==1:
        print
        print("HAHAHAHAHA. It was" + " " + str(number) + ".")
        again()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js