Contents

Chapter 1

Python Introduction

1.1 Python

- Most popular programming language in the world.
- Can write programs in just a few lines.
- Easy to learn.
- Lightweight (small install size).
- It is completely **free**.
- Available on most platforms (e.g. Windows, Apple, Linux, Raspberry Pi).
- Created by Guido van Rossum in 1989.

1.2 Where can I get it from?

Download Python from the Microsoft Store

Open "Microsoft Store", search for "python" and click on "Python 3.10" (Python Software Foundation) and then click on "Get". If Python 3.10 does not work on your computer, then try an older version like Python 3.9.

Note. If you are using your own personal computer then you can download Python from the official Python website:

`https://www.python.org/`

Microsoft Windows and Apple versions are available for free.

1.3 Keyboard Settings for Microsoft Windows

Make sure that your keyboard is set to US (not England) so that you can type single quotes ' and double quotes ".

- On your keyboard, the single quotes key ['] should be on the same key as double quotes ["]

  Use the [SHIFT] key to toggle between the two options.

- If you get @ when you press the double quotes key, then your keyboard is set for England(UK). To fix this, open "Settings" and search for "keyboard". Then install "English(Australia)" or "English(United States)".

- You can quickly change your keyboard version by clicking on, e.g. [ENG US], in the bottom right corner of your screen.

1.4 Your First Program

1.4.1 Console "Hello World"

- Load IDLE (Python)
- File → New File
- In the new window: File → Save as
- Choose an easy to find folder (desktop is fine) and type **first** as your filename
- Type **print("hello")** in the first.py window
- Click on Run → Run Module (or press F5)
- Notice that you now have two windows:
  input (**first.py**) and output (**Shell**)

Note

We can use the **Shell** as a calculator:

```
1+1 <Enter>
```

will ouput the answer 2

and

```
x = 4 <Enter>
y = 8 <Enter>
x/y <Enter>
```

will output 0.5

1.5 Calculator Program

Go back to **first.py** and type in the following:

Example

```
x = 2   # set x equal to 2
y = 7
print("x = ",x)
print("y = ",y)
print("x+y = ",x+y)
print("x-y = ",x-y)
print("x*y = ",x*y)
print("x/y = ",x/y)
print("x^2 = ",x**2)
print("x^3 = ",x**3)
```

Click on "Run → Run Module" to evaluate $x + y$, $x - y$, $x \times y$, $\frac{x}{y}$, x^2 and x^3

Note

The hash symbol `#` means 'comment' - do not execute the rest of the input line.

1.6 Long Input Lines

Python is designed so that expressions appear on one line only, which can lead to very long lines! If you have a really long line that needs to be broken into two lines, just write `\` at the end of the first line and press `<Enter>` to start the second (continuing) line.

Example

```
x = 1+2+3+4\
 +5+6
print(x)
```

Chapter 2

Loops

2.1 For Loop

- Type the following two lines into **first.py**:

```
for i in range(1,11):
   print(i)
```

print(i) is nested inside the 'for loop' and so must have at least one space before **p**

- Click on Run → Run Module
- Notice that the numbers 1 to 10 are printed in the output window.
- Notice also that the number 11 is NOT printed. Try this:

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

The 'range(10)' command outputs 10 numbers *starting* at 0 (the last number 10 is not included). So range(n) does not include n.

Warning!

You must type in the commands exactly as they are written. Do **NOT** write

```
For i in Range(10):
   Print(i)
```

This will **NOT** run !!

- Now run the following code:

```
for i in range(10):
   print(i)
   print("hi")
```

  Notice that **hi** is printed 10 times

- If we change the previous code to

```
for i in range(10):
   print(i)
print("hi")
```

  then **hi** is only printed once (since it is outside the 'for loop')

2.2 While Loop

We can also create a "while" loop with variables and conditions.

Example

```
n = 1
while n < 100:
  print(n)
  n = n + 1
```

Note:

The command

```
n = n + 1
```

adds 1 to n and then places that sum into n, that is, $n \leftarrow n + 1$. This simply means that the value of n **increases** by 1 each time the command is executed.

Note that

```
n = n + 1
```

is **not** a maths equation, and so we do not solve for n.

2.2.1 Semi-infinite Loop

A semi-infinite loop is a loop that will eventually stop, but we don't know when it will stop.

Example

```
n = 1
while n < 10000:
  print(n)
  n = n + 1
```

This loop will take a long time to complete. To exit the program:

- Close the **Shell** window (click on the cross); or
- In the **Shell** window click on Shell → Interrupt Execution

2.2.2 Infinite Loop

We can create loops that will run forever.

Example

```
while 1 == 1:
  print("Infinite")
```

Infinite loops are harder to exit. To exit the above program:

- Close the **Shell** window (click on the cross); or
- In the **Shell** window click on Shell → Restart Shell

2.2.3 Order of Execution

Compare

```
n = 1
while n < 4:
  print("n= ",n)
  n = n + 1
```

with

```
n = 1
while n < 4:
  n = n + 1
  print("n= ",n)
```

Note that changing the order of the commands **changes** the output of the program.

Chapter 3

Variables and their Type

Variable names are **case sensitive** and so 'A' is **different** from 'a'. The main 'types' of variables are Integer, Real (floating point), Boolean (True/False), and String.

3.1 Integer, Real, Boolean, String

3.1.1 Integer

Integer numbers are written as a number with no decimal point.

Example

```
i = 10
```

The above code makes the variable `i` have the integer 'type' and sets its value to 10.

3.1.2 Real (float)

Real numbers are written as a number with a decimal point, and are often called "floating point" numbers.

Example

```
x = 7.234
```

3.1.3 Boolean

Boolean variables can have the value `True` or `False`

Example

```
B = True
```

The above code makes the variable `B` have the Boolean 'type' and sets its value to `True`.

3.1.4 String

A string is any sequence of characters (including letters and numbers). We always start the string with " and end the string with ". Python also allows the use of single quotes ' when entering strings.

Example

```
s = "this is a string"
t = 'this is a string written with single quotes'
```

3.2 Automatic typing of a variable

You do not need to declare the type (int, real, string) of a variable (unlike nearly all other languages). Python automatically chooses the appropriate type. We can check the type of a variable by using the **type()** function.

Example

```
i = 10
print("i has type", type(i))
x = 7.234
print("x has type", type(x))
isprime = True
print("isprime has type", type(isprime))
s = "this is a string"
print("s has type", type(s))
```

Division (/) always returns a variable of type float.

Example

```
y = 1/2
print("y has type", type(y))
```

The output of mixed types (int and float) is float

Example

```
t = 0
print("t=0 has type", type(t))
t = t + 1/2
print("t=t+1/2 now has type", type(t))
t = t + 1/2
print("t = ",t)
print("type(t) = ", type(t)) # t= 1.0 is still a float
```

3.3 Large Integers

Unlike other languages, integers can have any size.

Example

```
x = 123456789
y = 123456789
z = x*y
print("z = ",z)
print("z**2 = ",z**2)
print("z**3 = ",z**3)
print("z**10 = ",z**10)
```

3.4 Real Numbers

Real numbers (float) are stored with a limited number of decimal places. We can normally get 16 significant figures which means at most 16 decimal places.

Example

```
x = 1.23456789
print("x = ",x)
print("x**2 = ",x**2)
print("x**3 = ",x**3)
```

Note

Binary numbers cannot store decimal numbers exactly, and so sometimes we will get slightly inaccurate results when we do math with real numbers.

Example

```
x = 10.1
print("x = ",x)
print("x**2 = ",x**2)
print("x**3 = ",x**3)
print("x**100 = ",x**100)
```

3.4.1 Rounding to n decimal places

The last few digits might be incorrect in a float number. Do not Panic. This is normal. We normally deal with this by rounding to 7 decimal places.

Example

```
x = 10.1
print("x = ",x)
print("x**2 = ",round(x**2,7))
```

Also note that `2.7048138294215165e+100` means

$$2.7048138294215165 \times 10^{100}$$

Note

The function `round(x,7)` rounds the number `x` to 7 decimal places.
For example, `round(0.12345678,7) = 0.1234568`

Also, after rounding, trailing zeros are removed.
For example, `round(0.12000000000,7) = 0.12`

3.5 Converting the Type of a Variable

Sometimes we need to convert one type of variable to another type. We can do this with the `int()`, `float()` and `str()` functions.

What is wrong with the following example?

Example

```
num1 = input("Enter first number: ")
num2 = input("Enter second number: ")
print(num1,"+",num2," = ",num1+num2)
```

The input function always returns a string, and so `num1+num2` is the two strings joined together!

We can fix this by immediately converting the output of `input` to a float.

Example

```
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
print(num1,"+",num2," = ",num1+num2)
```

3.5.1 char() and ord()

- `char()` converts from ASCII (number) to a character
- `ord()` converts from character to an ASCII (number) value

Example

```
print(ord('a'))
print(chr(65))
```

Chapter 4

If ... Else

We can control program flow with `if` and `else`

Example

```
y = 1
if y < 0:
    print("y is negative")
else:
    print("y >= 0")
```

Note that an `if` statement can be written in different ways.

Example

```
B = False

if B is True:
 print("B is true")
else:
 print("B is false")

if B == True:
 print("B is true")
else:
 print("B is false")

if B:
 print("B is true")
else:
 print("B is false")
```

4.1 elif

The command `elif` means 'else if'

Example

```
y = float(input("Input a number: "))
if y < 0:
    print(y," is negative")
elif y == 0:   # we use   ==   to test if two numbers are equal
    print(y," is zero")
else:
    print(y," is positive")
```

Note:

- Use == to test if two expressions are equal
- Use != to test if two expressions are NOT equal

4.2 Example: Average

In this section we are going to write a program to find the average of a set of numbers, but before we do that, let's look at a simpler example.

Example (find $1+2+...+n$)

```
# this program will find 1+2+...+n
Sum = 0  # Sum will be the sum of the numbers added so far
         # we need Sum to start at 0 so that
         # Sum = Sum + i starts with Sum = 1 when i = 1
n = 4   # Find the sum  1+2+...+4
i  = 1   # i is the number we will be adding to Sum
while i <=  n:
        Sum = Sum + i
        #print("** Debug: i = ",i,"  Sum = ",Sum," **")
        i = i + 1
print("Sum = 1+2+...+",n," = ", Sum)
print("i = ",i) # final value of i is n+1
```

Example (average of a set of numbers)

```
# this program will find the average of a set of numbers
# entered one by one
xin = "" # xin is the number we are entering (as a string)
         # "" is the empty string
         #           (anything not equal to "e" is okay)
Sum = 0  # Sum will be the sum of the numbers entered so far
         # we need Sum to start at 0 so that
         # Sum = Sum + x starts with Sum = x
         # (after reading the first number x)
n  = 0   # n is the count of the numbers we have entered so far
while xin != "e":
    xin = input("Enter Number (type e <Enter> to exit): ")
    if xin != "e":
        x = float(xin) # convert xin from a string to a real number
        Sum = Sum + x
        n = n + 1
average = Sum/n
print("Sum = ", Sum)
print("Average = ",average)
print("n = ",n)
```

Variable names and reserved words

In the above program, we used the variable names

`xin, Sum, n, x`

There is nothing special about these names. We could also choose

`xinput, Sumofnumbers, count, xreal`

but we must make sure that we do not use the 35 Python reserved words (which appear in red in IDLE) like

`False, True, and, or, if, else, elif, except, for, global, in, not, return, try, while, with`

For a full list of reserved words, type the following in "Shell"

```
import keyword
keyword.kwlist
```

4.3 Exercise

Exercise

Write a program to find the sum the squares of the natural numbers from 1 to 10, that is, find

$$1^2 + 2^2 + 3^2 + \ldots + 10^2$$

Hint: rearrange the following commands in the correct order

```
print("Sum = ", Sum)
Sum = Sum + i**2
i = i + 1
Sum = 0
n = 10
while i <=  n:
i  = 1
```

Answer: Your program should output

```
Sum = 385
```

Chapter 5

Random Library

The `random.randint(1,n)` function randomly chooses a number between 1 and n (including 1 and n). For example,

Example

```
import random

MyNumber = random.randint(1,3)
print(MyNumber)
```

Now let's choose 10 random numbers between 1 and 3

Example

```
import random

for i in range(10):
    MyNumber = random.randint(1,3)
    print(MyNumber)
```

Chapter 6

Turtle Graphics

To draw pictures with Turtle, we need to import the **turtle module**.

To do this, we just write

```
import turtle
```

at the top of our Python script.

Example

```
import turtle

turtle.forward(100)
```

Note

To avoid writing `turtle.` all the time, we can write

```
from turtle import *

forward(100)
```

Then all the turtle functions can be written without the `turtle.` prefix.

Warning

Because we are importing `turtle`, we must **not** use the name 'turtle.py' for our python input file!!!! Use something else like 'test.py'

6.1 turtle.forward(n)

Draws a line of length n pixels from the centre of the window $(x, y) = (0, 0)$ in the current direction the turtle is facing.

Example

```
import turtle

turtle.forward(100)
```

6.2 turtle.left(θ)

Turns turtle left θ degrees.

Example

```
import turtle

turtle.forward(100)
turtle.left(90)
turtle.forward(100)
```

6.3 turtle.right(θ)

Turns turtle right θ degrees.

Example

```
import turtle

turtle.forward(100)
turtle.right(30)
turtle.forward(100)
```

6.4 turtle.goto(x, y)

Moves turtle to the coordinates (x, y), and draws a line from its previous position to (x, y).

Example

```
import turtle

turtle.goto(20,200)
```

6.5 turtle.penup() and turtle.pendown()

When the pen is down, turtle draws a line.

Example

```
import turtle

turtle.forward(100)

turtle.penup()  # move without drawing
turtle.forward(100)

turtle.pendown()
turtle.forward(100)
```

6.6 turtle.speed(n)

We can slow turtle by using `turtle.speed(n)` where

- `n=1` is slowest
- `n=10` is fastest (actually `n=0` is 'draw as fast as possible')

Example

```
import turtle

turtle.speed(1)
turtle.forward(100)
turtle.left(30)
turtle.forward(100)
```

We can now use a `for loop` to draw a square

Example

```
import turtle

turtle.speed(1)

for i in range(10):
  turtle.forward(100)
  turtle.left(90)
```

and we can adjust the forward distance in the `for loop` to draw a spiral

Example

```
import turtle

turtle.speed(1)

for i in range(400):
 turtle.right(44)
 turtle.forward(20+i)
 turtle.dot()
```

6.7 Colour and Pen Size

We can change the colour and pen size.

Example

```
import turtle

turtle.color("blue", "blue")  # (pen colour, fill colour)
turtle.width(20)
turtle.forward(140)
```

Notice that turtle is too small in the above example.

We can use `shapesize(x stretch, y stretch, pen size)` to make turtle bigger.

Example

```
import turtle

turtle.color("blue", "yellow")
turtle.shapesize(4,4, 8)
turtle.width(10)
turtle.forward(140)
```

6.8 Turtle Shape

Example

```
import turtle

turtle.shape("turtle")
 # can use arrow, turtle, circle, square, triangle, classic
turtle.color("blue", "yellow")
turtle.shapesize(4,4, 8)
turtle.width(10)
turtle.forward(140)
```

6.9 Writing Text

We can write text at the current position of turtle.

Example

```
import turtle

turtle.write("Hello World")
```

and we can make the text larger by choosing the font

Example

```
import turtle

turtle.write("Hello World", font=("arial",20))
```

and we can also write other objects to the screen

Example

```
import turtle
x = 10
turtle.write(x)
```

Example

```
import turtle
x = 10
y = 20
turtle.write((x,y))
```

We can align the text with `"left"`, `"center"` or `"right"`

Example

```
import turtle
turtle.write((0,0), align="center")

turtle.forward(100)

turtle.write((0,100), align="center")
```

6.10 For Loop or While Loop?

We use a `for` loop to iterate a fixed (predetermined) number of times.

We use a `while` loop to iterate undetermined (even infinite) number of times.

Example

```
from turtle import *
import random

speed(3)
write("Loop", align="center",font=("Arial",60,"bold"))

color("blue","yellow")
penup()
goto(-75,-125)
pendown()
width(20)
shapesize(4, 4, 8)

while True:
 for i in range(8):
   forward(140)
   left(45)
 red = random.random()
 green = random.random()
 blue = random.random()
 pencolor(red,green,blue)
```

Chapter 7

Strings

7.1 Definition

A string is a sequence of characters. We enter a string with single or double quotes:

Example

```
mystring = 'hello'
mystring2 = " there"
print(mystring + mystring2)
```

7.2 Index

We can read individual characters in a string `s` by using an index: `s[i]` is the ith character (note that `s[0]` is the first character)

Example

```
mystring = 'hello'
print(mystring[1])
```

7.3 Slice

We can read a substring in a string `s` by using a slice: `s[i:j]` is the substring starting at i and ending at j-1

Example

```
mystring = 'hello'
print(mystring[1:3])
```

Options

(a) `s[i:]` is the substring of `s` starting at position `i` up to the end of the string `s`

(b) `s[:i]` is the substring of `s` starting at the beginning and ending at position `i-1`

(c) `s[:-1]` is the substring of `s` starting at the beginning and ending at one character before the end of the string `s`

Note: Strings are immutable (individual characters cannot be changed)

```
mystring = 'hello'
mystring[2] = 'x'   # gives an error
```

If we want to change the character at position 2, then we need to create a new string and use split:

```
mystring = 'hello'
mystring2 = mystring[:2]+'x'+mystring[3:]
print(mystring2)  # changes first "l" to an "x"
```

Example (String Subtraction)

We can subtract from a string by creating a new string that contains parts of the old string.

```
#remove Like from TheyLikeComputers

string1 = "TheyLikeComputers"  # string[0] = 'T'
                               # string1[8] = 'C'

string2 = string1[0:4]+string1[8:]
  #string1[8:] starts at position 8 and ends at end of string1

print(string2) # removed Like from string1
```

7.4 String Replace

We can replace a substring of string `s` with a different string by using `s.replace()`. But note that `s` will not be changed, and so you must use a different string to store the result.

Example (String Replace)

```
#replace Like from TheyLikeComputers with Have

string1 = "TheyLikeComputers"
string2 = string1.replace("Like","Have")
print("string1 = ", string1)
print("string2 = ", string2)
```

Note: All matches are replaced

```
string1 = "TheyLLLComputers"
string2 = string1.replace("L","Have")
print("string1 = ", string1)
print("string2 = ", string2)
```

7.5 Uppercase and Lowercase

We can return the uppercase and lowercase versions of a string `s` with `s.upper()` and `s.lower()`. But note that `s` will not be changed, and so you must use a different string to store the result.

Example (String Replace)

```
string1 = "TheyLikeComputers"
string2 = string1.lower()
string3 = string1.upper()
print("string1 = ", string1)
print("string2 = ", string2)
print("string3 = ", string3)
```

7.6 Remove spaces from beginning and end of string

We can remove trailing spaces from a string `s` with `s.strip()`. But note that `s` will not be changed, and so you must use a different string to store the result.

Example (Remove Spaces)

```
string1 = "    Like    "
string2 = string1.strip()
print(":"+string1+":")
print(":"+string2+":")
```

7.7 Length

The length of a string `s` is given by `len(s)`

Example

```
mystring = 'hello'
print(len(mystring))
```

7.8 Position of a substring in a string

We find the position of a substring `sub` in a string `s` by using `s.find(sub)`

Example

```
mystring = 'hello'
print(mystring.find('l'))
```

Note

`s.find(sub)` is zero based, and so if `sub` is located at the start of `s` then `s.find(sub)` will return `0`

If `sub` is not contained in `s` then `s.find(sub)` will return `-1`

7.9 split()

We can chop a string `s` in pieces with `s.split()`
The result is actually a list of strings (we will look at lists later).

The default **delimiter** (place to chop) is a space.

Example

```
mystring = 'hello there you'
print(mystring.split())
```

We can specify the **delimiter** by inserting it inside the brackets of `split()`

Example

```
mystring = 'hello,there,you'
print(mystring.split(','))
```

7.10 isdigit()

We can test if a string `s` represents a natural number with `s.isdigit()`

Example

```
mystring = 'hello'
print(mystring.isdigit())  # False

mystring = '1234'
print(mystring.isdigit())  # True

mystring = '1234.8'
print(mystring.isdigit())  # False

mystring = '-12'
print(mystring.isdigit())  # False
```

Note.

Unfortunately `s.isnumeric()` and `s.isdecimal()` give the same results in the above example. This is because `s.isdecimal()` is used to check if unicode characters are digits, and `s.isdecimal()` is used for other languages.

Example.

We can use `s.isdigit()` to make sure natural numbers are entered:

```
s1 = input("first number = ")
s2 = input("second number = ")
if s1.isdigit() and s2.isdigit():
    print ("Sum = ",int(s1)+int(s2))
else:
    print("You must enter natural numbers!")
```

Chapter 8

Time Library

We can access the clock in our computer by writing `import time`

```
import time

start = time.time()

while True:
  print(time.time()-start)
```

`time.time()` gives the number of seconds elapsed since midnight 1st January, 1970.

Note

This "starting date" can be different on other operating systems. You can check the start date with

```
time.localtime(0)
```

You can access current time and date with

```
time.localtime()
```

Example: Time and Date

```
import time
import datetime
t  = time.localtime()
mysec = str(t.tm_sec)
mymin = str(t.tm_min)
myhour = str(t.tm_hour)
print(myhour+":"+mymin+":"+mysec  )

today = datetime.date.today()
print(today.strftime("%d %B %Y"))
```

8.1 Speed Test Examples

Example: Number of integer multiplications in 1 second

```
import time
print("One Second Speed Test")
count = 0
start = time.time()
while (time.time() - start < 1):
    x2 = 47586+count
    y2 = 86869+count
    z2 = x2*y2
    count = count + 1
print("integer multiplications: ",count)
```

Example: Number of real number divisions in 1 second

```
import time
print("One Second Speed Test")
count = 0
start = time.time()
while (time.time()-start < 1):
   x3 = 687.6979+count
   y3 = 78.96969+count
   z3 = x3 / y3
   # print(z3)
   count = count + 1
print("floating point divisions:",count)
```

It is best to avoid checking the time with `time.time()` in the while loop since that "operating system call" will slow down the program.

Example.

```
import time
print("Speed Test")
count = 0
start = time.time()
while (count < 20000000): # 20 million
    x2 = 47586+count
    y2 = 86869+count
    z2 = x2*y2
    count = count + 1
print("time =  ",time.time()-start, " seconds")
  # time =   2.55 seconds on 2022 Surface Pro (4.1 GHz)
  #                          i5 - 1145G7
```

Note: The C++ version is much faster

The C++ version is much faster since it is compiled to machine code rather than interpreted like Python.

```
#include <iostream>
#include <ctime>
int main()
{
    clock_t starttime;
    int count = 0;
    int x2, y2, z2;
    z2 = 0;
    starttime = clock();
    while (count < 20000000) // 20 million
    {
        x2 = 47586 + count;
        y2 = 86869 + count;
        z2 = x2 * y2;
        count++;
    }
    std::cout << z2 << "\n"; // need this to remove optimizations
    // (the compiler ignores unnecessary code with Compile->Release)
    std::cout << "time = " << (clock() - starttime) / 1000.0 << " seconds \n";
}
```

Chapter 9

Integer Division and Remainder after Division

When we divide an integer a by a positive integer b, we can write the answer as an integer quotient q and an integer remainder r

$$\frac{a}{b} = q + \frac{r}{b} \qquad \text{where } 0 \leq r < b$$

You may have used the process called **long division** in school to calculate q and r .

Python has the operation `//` to calculate the quotient q, and the operation `%` to calculate the remainder r .

These operations are often used in programs that use computer graphics (pictures), and for finding prime numbers.

Example

```
q = 10 // 3      # q is the integer part of 10/3
r = 10 % 3       # r is the remainder when we divide 10 by 3
print("10/3 = ",10/3)
print("10/3 = ",q,"+",r,"/ 3")
```

If a is negative, then q is rounded **down**

```
q = -10 // 3
r = -10 % 3
print("-10/3 = ",-10/3)
print("-10/3 = ",q,"+",r,"/ 3")
```

9.1 Odd and Even Numbers

An integer n is called **even** if it has a remainder of 0 when it is divided by 2, that is `n % 2 = 0`

An integer n is called **odd** if it has a remainder of 1 when it is divided by 2, that is `n % 2 = 1`

Example

```
n = int(input("Enter an integer: "))
if n % 2 == 0:
   print(n," is even")
if n % 2 == 1:       # or we could use   else    here
   print(n," is odd")
```

Note that we use the `int()` function to convert the string from `input()` into an integer.

Now let's improve the above program, so that we can enter as many numbers as we like

Example

```
while True:
   n = int(input("Enter an integer: "))
   if n % 2 == 0:
     print(n," is even")
   if n % 2 == 1:
     print(n," is odd")
```

The above program is in an infinite loop, and so we should let the user enter `x` to exit.

Example

```
exit = False
while exit == False:
   s = input("Enter an integer (x to exit): ")
   if s != "x":
     n = int(s)
     if n % 2 == 0:
       print(n," is even")
     if n % 2 == 1:
       print(n," is odd")
   else:
     exit = True
```

Notice that the indentation (spaces) in the above program are **super-important** since they tell Python when a block of code begins and ends. Python does not use brackets `{}` for beginning and ending blocks of code.

Example

Now let's print all even numbers between 0 and 100

```
for n in range(0,101):
     if n % 2 == 0:
       print(n)
```

Exercise : Print all odd numbers between 0 and 100

Exercise : Print all odd numbers between -100 and 100

Chapter 10

Guess Number Game

The first game that most programmers write (since the 1970s) is the famous "guess the number" game. This game is easy to write in Python.

Example

```
import random

MyNumber = random.randint(1,20)

print("I am thinking of a number between 1 and 20 (including 1 and 20)")
print("Try and guess the number")
print("Enter -1 to quit")

Guess = 0
n = 0 # number of guesses

while Guess != -1:   # we use   !=   to test if two numbers are NOT equal
   n = n + 1
   Guess = int(input("Input your guess: "))
   if Guess < MyNumber:
      print("My number is bigger")
   if Guess > MyNumber:
      print("My number is smaller")
   if Guess == MyNumber:
      print("You guessed my number!!!")
      Guess= -1  # exit while loop
      print("Number of guesses = ",n)
print("Game Over")
```

Notice that there is a minor bug in this program: after we enter -1 to exit, the program still tells us if -1 is bigger or smaller than `MyNumber`

We can fix the bug by adding another `if` statement

Example

```
import random

MyNumber = random.randint(1,20)

print("I am thinking of a number between 1 and 20 (including 1 and 20)")
print("Try and guess the number")
print("Enter -1 to quit")

Guess = 0
n = 0 # number of guesses

while Guess != -1:   # we use   !=   to test if two numbers are NOT equal
   n = n + 1
   Guess = int(input("Input your guess: "))
   if Guess != -1:   # jump over this part if Guess = -1
     if Guess < MyNumber:
      print("My number is bigger")
     if Guess > MyNumber:
      print("My number is smaller")
     if Guess == MyNumber:
      print("You guessed my number!!!")
      Guess= -1  # exit while loop
      print("Number of guesses = ",n)
print("Game Over")
```

Chapter 11

Functions

We can make the function $f(x) = x^3$ as follows

Example

```
def cube(x):
    return x**3  # return is the output of the function

print(cube(3))
print(cube(7))
print(cube(10))
```

We can choose any name for the function, and we can even call the function `f`

Example

```
def f(x):
    return x**3

print(f(3))
```

The parameter `x` can be changed to any other variable

Example

```
def f(a):
    return a**3

print(f(3))
```

All parameters (and variables) inside the function are **local** and cannot be accessed outside the function.

Example

We get an error when we run the following program:

```
def f(a):
    return a**3

print(a)     # error: a is not defined outside the function
print(f(3))
```

11.1 Function Parameters

Functions can have as many parameters as we like

Example

```
def quad(a,b,c,x):
    return a*x**2+b*x+c

a=1
b=3
c=1
x=1.1
print("a =",a)
print("b =",b)
print("c =",c)
print("x=",x," => ", "ax**2+bx+c = ",quad(a,b,c,x))
```

11.2 Function Return Types

Functions can return any type of variable.

Example

```
def oddeven(n):
    if n % 2 == 0:
       return "even"
    else:
       return "odd"

n=7
print(n," is ",oddeven(n))
```

Example

Now let's just print the even numbers between 0 to 100

```
def oddeven(n):
    if n % 2 == 0:
       return "even"
    else:
       return "odd"

for n in range(101):
  if oddeven(n) == "even":
     print(n," is ",oddeven(n))
```

Exercise : Print all odd numbers between 0 and 100

Chapter 12

More on Functions

12.1 ord() and char()

`ord()` converts a **letter** to its corresponding ASCII code **number**

Example

```
print(ord("A"))
print(ord("B"))
```

`chr()` converts a **number** to its corresponding ASCII code **letter**

Example

```
print(chr(65))
print(chr(66))
```

12.2 Function Return Value

If we don't supply a return value then the function returns `None`

Example

```
def f(x):
    x = x +1

print(f(1))
```

We can use `None` to report an error in the function's evaluation

Example

```
def f(x):
    if x != 0:
        return 1/x
    else:
        return None

print(f(0))
```

After the program gets to `return` the function stops executing and returns to the main program.

Notice the difference between

```
def  f(x):
     print("hello")
     return x

print(f(3))
```

and

```
def  f(x):
     return x
     print("hello")

print(f(3))
```

12.3 Function Default Parameters

Sometimes we don't want to enter all the parameters in a function, and we can use **default** values in that case. First consider the following example.

Example

We have seen how to iterate through the numbers 0, . . . , 9 with a for loop

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

Well, we can use a `for loop` to iterate through the letters (characters) in a string

```
for i in "hello":
 print(i)
```

Now let's convert that into a function with two parameters `mystring` and `timegap`

`mystring` is the input string

`timegap` is the delay between printing characters

```
import time

def slowprint(mystring, timegap):
  for c in mystring:
    print(c)
    time.sleep(timegap)

slowprint("Hello There!",0.6)
```

We can even print out the ascii values

Example

```
import time

def slowprint(mystring, timegap):
  for c in mystring:
    print(c," = ",ord(c))
    time.sleep(timegap)

slowprint("Hello There!",0.6)
```

If we don't want to input `timegap` then we can set it as a **default** value

Example

```
import time

def slowprint(mystring, timegap=0.1): # default value of timegap is 0.1
  for c in mystring:
    print(c," = ",ord(c))
    time.sleep(timegap)

slowprint("Default speed") # we are not choosing timegap
                          # so the default value is 0.1 is used.
print("------")
slowprint("Choose speed",0.6)
```

12.4 Function Parameter Names

When calling a function, we can specify the parameters by name

Example

```
slowprint(mystring="hello", timegap=0.2)
```

and even change the order of the parameters (if they are specified by name)

Example

```
slowprint(timegap=0.2,mystring="hello")
```

but

```
slowprint(0.2,"hello")
```

will generate an error.

12.5 Functions and Global Variables

Sometimes we want a function to change external variables defined outside the function. These variables are called **global** variables.

Example

```
x  = 10

def f():
  x = 2
  print("function done")
  return

f()
print("x = ",x)
```

Notice that `x=10` is not changed by the function. If we want to change the value of `x`, then we must declare `x` as a global variable **inside** the function.

```
x  = 10

def f():
  global x
  x = 2
  print("function done")
  return

f()
print("x = ",x)
```

We can always access an external variable inside a function.

Example

```
x  = 10

def f():
  print("x = ",x)
  return

f()
```

but we cannot **change** the external variable inside the function (unless we use `global`).

12.6 Roulette Wheel

A roulette wheel randomly chooses a number from 0 to 36.

Example

```
import random

r = random.randint(0,36)

print(r)
```

Suppose that we bet $1 on **even**: if the number chosen by the wheel is even, then the payout is 1 to 1 (meaning that we win an extra $1); otherwise we lose our $1.

Note

Even though 0 **is** an even number, this game does **not** allow 0 to be even. This makes **odd** and **even** have the same probability, and it also ensures that the Casino makes money :) Some Casinos have 0 **and** 00 to make even more money.

In this section, we are going to make a very simple simulation of a roulette wheel.

```
import random

money = 10  # this is our global variable

def beteven(bet, wheel):  #bet is the amount of money we are betting
    global money
    if (wheel % 2 == 0) and (wheel != 0):
        print("You win!")
        money = money + bet
    else:
        print("You lose")
        money = money - bet

r = random.randint(0,36) # r is the number chosen by the
                         # rolling ball in the wheel

print("wheel number chosen = ",r)

beteven(1,r)  # bet one dollar that r is even
              # and update money depending on whether we win or lose

print("money = ",money)
```

Let's make the program more interactive.

Example

```
import random, time

money = 10

def beteven(bet, wheel):
    global money
    if (wheel % 2 == 0) and (wheel != 0):
        print("You win!")
        money = money + bet
    else:
        print("You lose")
        money = money - bet

def betodd(bet, wheel):
    global money
    if (wheel % 2 == 1):
        print("You win!")
        money = money + bet
    else:
        print("You lose")
        money = money - bet

choice = input("Input bet even(e), bet odd(o), or exit(x): ")
while choice != "x":
  print("rolling ...")
  time.sleep(1)
  r = random.randint(0,36)
  print("wheel number chosen = ",r)
  if choice == "e":
      beteven(1,r)
  if choice == "o":
      betodd(1,r)
  print("money = ",money)
  choice = input("Input bet even(e), bet odd(o), or exit(x): ")
```

Finally, if a user enters a letter different from `e`, `o`, `x` then tell the user that they have not made a bet.

Example

```
import random, time

money = 10

def beteven(bet, wheel):
    global money
    if (wheel % 2 == 0) and (wheel != 0):
        print("You win!")
        money = money + bet
    else:
        print("You lose")
        money = money - bet

def betodd(bet, wheel):
    global money
    if (wheel % 2 == 1):
        print("You win!")
        money = money + bet
    else:
        print("You lose")
        money = money - bet

choice = input("Input bet even(e), bet odd(o), or exit(x): ")
while choice != "x":
  print("rolling ...")
  time.sleep(1)
  r = random.randint(0,36)
  print("wheel number chosen = ",r)
  if choice == "e":
      beteven(1,r)
  elif choice == "o":
      betodd(1,r)
  elif choice != "x":
    print("No bet made!")
    print("Please enter e or o  to make a bet of $1")
  print("money = ",money)
  choice = input("Input bet even(e), bet odd(o), or exit(x): ")
```

12.7 More Examples

Note

`def` is short-hand notation for 'define function'

Example

There is a difference between `f(2)` and `print(f(2))`

Compare

```
def f(x):
  print("function done")
  return x*x

f(2)
```

with

```
def f(x):
  print("function done")
  return x*x

print(f(2))
```

Note that `print(f(2))`

- First executes `f(2)` and so prints 'function done'
- Returns $2 \times 2 = 4$
- Then **prints** the return value of `f(2)`, namely 4.

Note

If `f(x)` does not return a value

```
def f(x):
  print("function done")
  return

print(f(2))
```

then the return value of `f(x)` will be `None` and so `print(f(2))` will print `None`.

If a function does not return a value then we can omit writing `return` and then Python will exit the function when it finds a blank line.

Example

```
def f(x):
  print("function done")

f(2)
```

12.8 Prime Numbers

A **prime** number is a natural number $n > 1$ which is divisible only by 1 and n.

For example, 2 is prime but $6 = 2 \times 3$ is not prime.

Let's write a function to test for prime numbers.

```
def testprime(n):
    for i in range(2,n):
     if n % i == 0:   # divide n by i and check remainder
       return False
    return True
```

The next example shows how to use this function.

Example

```
def testprime(n):
    for i in range(2,n):
     if n % i == 0:   # divide n by i and check remainder
       return False
    return True

n = 10

if testprime(n):
  print(n," is prime")
else:
  print(n," is not prime")
```

Now let's print all prime numbers up to `limit=1000`

Example

```
limit = 1000

def testprime(n):
    for i in range(2,n):
     if n % i == 0:   # divide n by i and check remainder
       return False
    return True

print("The prime numbers up to ",limit," are")

for i in range(2,limit+1):
   if testprime(i) == True:
     print(i)
```

12.9 Multiple Return Values

Functions can return multiple values by returning a **tuple**

Example

```
def f(s):  # input string s, and output (1) capital version and (2) size
  capital = s.upper()
  size = len(s)   # number of characters in string
  return(capital,size)

s = "hello there"
t = f(s)
print(t)
```

In mathematical terminology, a tuple is a vector $\mathbf{v} = (v_1, v_2, \ldots)$, but in Python (and C++), a tuple's index starts at zero. This just means that the first element in the tuple `t` is `t[0]` and the second element is `t[1]`

Example

```
def f(s): # input string s, and output (1) capital version and (2) size
  capital = s.upper() # convert string to CAPITALS
  size = len(s)  # number of characters in string
  return(capital,size)

s = "hello there"
t = f(s)
print("Capital version = ",t[0])
print("length of string = ",t[1])
```

Chapter 13

The Bisection Method for Solving Equations

Let's try to solve $x^2 + 3x + 1 = 0$

Example

```
def f(x):
    return x**2+3*x+1

x=1
print("f(",x,") = ",f(x))
x=0
print("f(",x,") = ",f(x))
x=-1
print("f(",x,") = ",f(x))
```

Notice that there is a solution between $x = 0$ and $x = -1$

Our next guess is the average $x = \frac{0+(-1)}{2} = -0.5$

First let's make it easier to input the numbers

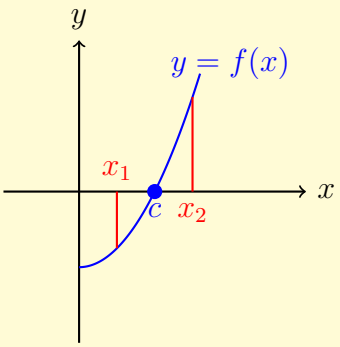
Example

```
def f(x):
    return x**2+3*x+1

while True:
    x = float(input("x = "))
    print("f(",x,") = ",f(x))
```

Now let's write a program to find the solution.

If f is a continuous function with $f(x_1) < 0$ and $f(x_2) > 0$ then there is a number c such that $f(c) = 0$.



We first guess the value of c by taking the average of x_1 and x_2, namely $x_a = \frac{x_1+x_2}{2}$.

- if $f(x_a) > 0$ then we repeat the above process with $f(x_1) < 0$ and $f(x_a) > 0$
- if $f(x_a) < 0$ then we repeat the above process with $f(x_a) < 0$ and $f(x_2) > 0$

Let's try to solve $x^2 + 3x + 1 = 0$ again.

First print some values of $f(x)$ to see where it is positive and where it is negative.

Example

```
def f(x):
    return x**2+3*x+1

for i in range(-10,10):
   print("f(",i,")=",f(i))
```

We see that $f(-1) < 0$ and $f(0) > 0$

and so let $x_1 = -1$ and $x_2 = 0$

Example

```
def f(x):
    return x**2+3*x+1

for i in range(-10,10):
   print("f(",i,")=",f(i))

x1 = -1.0 # make sure these are real numbers (float)
x2 = 0.0
xa = (x1+x2)/2
print("xa=",xa)
print("f(xa)=",f(xa))
```

Now put the above in a loop making sure that

```
f(x1) < 0
f(x2) > 0
```

To get started, let's use an infinite loop

Example

```
def f(x):
    return x**2+3*x+1

x1 = -1.0 # make sure these are real numbers (float)
x2 = 0.0
xa = (x1+x2)/2
while True:  # always have f(x1) < 0 and f(x2) > 0
 print("xa=",xa)
 print("f(xa)=",f(xa))
 if f(xa) > 0:
     x1=x1 # so x1 does not change!
     x2=xa
 if f(xa) < 0:
     x1=xa
     x2=x2 # so x2 does not change!
 xa = (x1+x2)/2
```

Now lets tidy up the program, but before we do that, let's look at the `round` function.

13.1 round(number, decimal places)

The `round` function rounds a `float` to a specified number of decimal places.

Example

```
print(round(1.49,1)) # round up to 1.5
print(round(1.44,1)) # round down to 1.4
print(round(1.45,1)) # sometimes up, sometimes down
                     # decimal numbers are approximate in binary
```

If we don't specify the number of decimals places then `round` rounds to the nearest integer.

Example

```
print(round(1.9)) # round up to 2
```

Now lets stop the program when $f(x) = 0$ to 12 decimal places:

Example

```
def f(x):
    return x**2+3*x+1

x1 = -1.0 # make sure these are real numbers (float)
x2 = 0.0
xa = (x1+x2)/2
while round(f(xa),12) != 0:
 if f(xa) > 0:
     x1=x1 # so x1 does not change!
     x2=xa
 if f(xa) < 0:
     x1=xa
     x2=x2 # so x2 does not change!
 xa = (x1+x2)/2

print("f(x)=0 when x = ",round(xa,8))
```

13.2 Choosing the size of the error

We can choose the maximum size of the error in our solution by checking $|x_1 - x_2|$.

Example

```
def f(x):
    return x**2+3*x+1

x1 = -1.0 # make sure these are real numbers (float)
x2 = 0.0
xa = (x1+x2)/2
while abs(x1-x2) > 0.00000001:
 if f(xa) > 0:
     x1=x1 # so x1 does not change!
     x2=xa
 if f(xa) < 0:
     x1=xa
     x2=x2 # so x2 does not change!
 xa = (x1+x2)/2

print("f(x)=0 when x = ",round(xa,8)," to 8 decimal places")
```

Finally, lets get the program to guess the initial values of x_1 and x_2, trying all integers between -10 and 10. Then we have a good chance of finding all solutions.

Example

```
def f(x):
    return (x+2)*(x-3)*(x-6)

for i in range(-10,10):
    for j in range(-10,10):
        if (f(i) < 0) and (f(j) > 0):
            x1=i
            x2=j
            xa = (x1+x2)/2
            count = 0
            while (abs(x1-x2) > 0.00000001) and (count < 1000):
             count=count+1
             if f(xa) > 0:  # always have f(x1) < 0 and f(x2) > 0
               x1=x1 # so x1 does not change!
               x2=xa
             if f(xa) < 0:
               x1=xa
               x2=x2 # so x2 does not change!
             xa = (x1+x2)/2
            print("f(x)=0 when x = ",round(xa,8), "to 8 decimal places")
```

Chapter 14

More Turtle Graphics

We can use `import turtle` to draw graphics (pictures).

Warning

Because we are importing `turtle`, we must **not** use the name 'turtle.py' for our python input file!!!! Use something else like 'test.py'

14.1 Window Size

We can adjust the window size as follows.

Example

```
import turtle

turtle.setup(width=0.8,height=0.6)  # sets width and height
                                    # of window to 0.8 of screen width
                                    # and 0.6 of screen height
turtle.goto(100,100)
```

Example

```
import turtle

turtle.setup(width=800,height=600)  # sets width = 800 pixels
                                    # and height = 600 pixels
turtle.goto(100,100)
```

We can find the width and height of the window as follows.

Example

```
import turtle

turtle.setup(width=.8,height=.6)

turtle.goto(100,100)
print("Window width = " + str(turtle.window_width()) )
print("Window height = " + str(turtle.window_height()) )
```

14.2 Filled Rectangle

Example

```
import turtle

turtle.fillcolor("yellow")

turtle.begin_fill()

for i in range(2):
    turtle.forward(200)
    turtle.right(90)
    turtle.forward(100)
    turtle.right(90)

turtle.end_fill()
```

14.3 Keyboard Input

Turtle can read keyboard input by usuing the `onkey()` method.

Example

```
import turtle

def pointup():
    turtle.setheading(90)

def pointleft():
    turtle.setheading(180)

def pointright():
    turtle.setheading(0)

def pointdown():
    turtle.setheading(270)

while True:
    turtle.forward(1)
    turtle.onkey(pointup,"Up") # arrow key up
    turtle.onkey(pointleft,"Left")
    turtle.onkey(pointright,"Right")
    turtle.onkey(pointdown,"Down")
    turtle.listen()
```

Notice that we use a `while True:` loop rather than a `for` loop, so that the loop does not end.

14.4 Mouse Click Location

Turtle can read the location of a mouse click.

Example

```
import turtle

def showclicklocation(x,y):
    print("x,y= ",x,",",y)

turtle.onscreenclick(showclicklocation)
```

14.5 Exit Turtle (close program)

To close the program use `turtle.bye()`

Example

```
import turtle

def showclicklocation(x,y):
    print("x,y= ",x,",",y)
    if y < 100:
      turtle.bye()

turtle.onscreenclick(showclicklocation)
```

14.6 Clock

Here is an example for creating a real-time clock in turtle.

Example

```
import turtle

import time

turtle.speed(0) #fastest

def zstr(num):
    if num > 9:
        return str(num)
    else:
        return "0"+str(num)

turtle.hideturtle()
turtle.penup()
turtle.pencolor("blue")

while True:
    timedata = time.localtime()
    hour = str(timedata.tm_hour)
    minute = zstr(timedata.tm_min)
    second = zstr(timedata.tm_sec)
    mytime = hour+":"+minute+":"+second
    turtle.tracer(False) # do not update screen (speed up drawing)
    turtle.clear()
    turtle.goto(-430,-100)
    turtle.write(mytime,font=("Arial", 160, "bold"))
    turtle.tracer(True) # update screen
    time.sleep(1)
```

Note

For a description of all turtle commmands, have a look at

https://docs.python.org/3.3/library/turtle.html

Chapter 15

Lists

Python has 3 high-level data objects called **lists**, **tuples** and **dictionaries**. We will first look at lists.

15.1 Introduction to Lists

Lists are like sequences $a_0, a_1, a_2, \ldots$ where a_i can be any object (even another list).

15.1.1 List creation

The following example shows how to create a list containing 2 strings and one number.

Example

```
L=["hello","there", 1.23]

print(L)
print(L[0])
print(L[1])
print(L[2])
```

Notice that `L[0]` is the first element of the list.

15.1.2 Empty list

An empty list contains no elements. The following example shows how to create an empty list.

Example

```
L=[]
```

15.1.3 Number of items in list

We use the function `len()` to output the number of elements in a list.

Example

```
L=["hello","there", 1.23]

print("L=",L)
print("number of items = ", len(L))
```

15.1.4 Iterate through list

We can 'look' at each element of a list by using a `for` loop.

Example

```
L=["hello","there", 1.23]

for i in L:
  print(i)
```

Note

To count each element we could write

```
L=["hello","there", 1.23]

count = 0

for i in L:
  count = count + 1
  print(str(count)+":",i)
```

15.1.5 Sorting a list

Lists are **objects** and so have **methods** (built-in functions)

One method (function) of a list is to **sort** the list.

Example

```
L=["hello","there", "a"]

print("L = ",L)

L.sort()

print("sorted L = ",L)
```

We can also sort a list in reverse.

Example

```
L=["hello","there", "a"]

print("L = ",L)

L.sort()
L.reverse()

print("reverse sorted L = ",L)
```

15.1.6 Add element to list

We can use the `append` method to add an element to a list.

Example

```
L=["hello","there", "a"]

print("L = ",L)

L.append("BP")

print("Appended L = ",L)
```

15.1.7 Remove last element from list

We can use the `pop` method to remove the last element from a list.

Example

```
L=["hello","there", "a"]

print("L = ",L)

E = L.pop()

print("new L = ",L)
print("popped element = ",E)
```

Note

`pop` is useful when we want to deal a card from a pack of cards. See the following example for details.

15.1.8 Clearing a list

We can remove every element from a list `L` with `L.clear()`

Warning

Don't use `L=[]` since that will create a new empty list and make `L` point to that new list. The old list will remain in memory, but (should) be cleared at garbage collection time.

15.2 Example: Deck of Cards

Let's make a deck of cards.

Example

```
Nums = ["Ace"]

for i in range(2,11):
    Nums.append(str(i))

print(Nums)
```

We still need to add Jack, Queen, and King.

Example

```
Nums = ["Ace"]

for i in range(2,11):
    Nums.append(str(i))

Nums = Nums + ["Jack","Queen","King"]

print(Nums)
```

Notice that we can use `+` to add two lists together (just like strings).

We also need the suits Hearts, Diamonds, Spades and Clubs.

Example

```
Suits = ["Hearts", "Diamonds", "Spades", "Clubs"]

print(Suits)
```

We now can join `Nums` to `Suits` to make our Cards list

Example

```
Nums = ["Ace"]

for i in range(2,11):
    Nums.append(str(i))

Nums = Nums + ["Jack","Queen","King"]
Suits = ["Hearts", "Diamonds", "Spades", "Clubs"]
Cards = []

for s in Suits:
    for n in Nums:
        Cards.append(n+" of " + s)

print(Cards)
print("Number of cards = ",len(Cards))
```

15.2.1 Shuffle the cards

Now let's swap the first card with the last card.

Example

```
def swap(i,j):
    global Cards
    temp = Cards[i]
    Cards[i] = Cards[j]
    Cards[j] = temp

Nums = ["Ace"]

for i in range(2,11):
    Nums.append(str(i))

Nums = Nums + ["Jack","Queen","King"]

Suits = ["Hearts", "Diamonds", "Spades", "Clubs"]

Cards = []

for s in Suits:
    for n in Nums:
        Cards.append(n+" of " + s)

print("Number of cards = ",len(Cards))

print(Cards)

print("Top card = ",Cards[0])
print("Bottom card = ", Cards[51])

swap(0,51)
print(Cards)

print("Top card = ",Cards[0])
print("Bottom card = ", Cards[51])
```

To shuffle the cards, we perform 100 random swaps

Example

```
import random

def swap(i,j):
    global Cards
    temp = Cards[i]
    Cards[i] = Cards[j]
    Cards[j] = temp

Nums = ["Ace"]

for i in range(2,11):
    Nums.append(str(i))

Nums = Nums + ["Jack","Queen","King"]

Suits = ["Hearts", "Diamonds", "Spades", "Clubs"]

Cards = []

for s in Suits:
    for n in Nums:
        Cards.append(n+" of " + s)

print(Cards)

print("Shuffling")

for i in range(100):
    swap(random.randint(0,51),random.randint(0,51))

print(Cards)
```

15.2.2 Dealing cards

Finally, we can deal 5 cards from the deck by using `pop`

Example

```
print(Cards)
print(Cards.pop())
print(Cards.pop())
print(Cards.pop())
print(Cards.pop())
print(Cards.pop())
print(Cards)
```

And we can store the cards in a hand

Example

```
import random

def swap(i,j):
    global Cards
    temp = Cards[i]
    Cards[i] = Cards[j]
    Cards[j] = temp

Nums = ["Ace"]
for i in range(2,11):
    Nums.append(str(i))

Nums = Nums + ["Jack","Queen","King"]
Suits = ["Hearts", "Diamonds", "Spades", "Clubs"]
Cards = []
for s in Suits:
    for n in Nums:
        Cards.append(n+" of " + s)

print(Cards)

print("Shuffling")

for i in range(100):
    swap(random.randint(0,51),random.randint(0,51))

print(Cards)

hand = [Cards.pop(),Cards.pop(),Cards.pop(),Cards.pop(),Cards.pop()]

print("Hand of cards = ",hand)
```

15.3 List Membership

We can use `in` to test if an object is in a list.

Example

```
L = ["a","b","d"]

if "d" in L:
    print("Yes, d is in ", L)
else:
    print("No, d is NOT in ",L)

if "c" in L:
    print("Yes, c is in ", L)
else:
    print("No, c is NOT in ",L)
```

Notice that the test is case-sensitive and does not find 'partial' matches.

Example

```
L = ["a","b","D","do"]

if "d" in L:
    print("Yes, d is in ", L)
else:
    print("No, d is NOT in ",L)
```

We can test if an element is **not** in a list with `not in`

Example

```
L = ["a","b","D","do"]

if "e" not in L:
    print("No, e is NOT in ",L)
```

If we want to find **partial matches** then we can iterate through the entire list with a `for` loop, and check for a partial match with each string in the list.

Note

We can test for a partial match in a string by using `in`

```
if "d" in "do":
    print("d is in do")
```

Example

```
L = ["a","b","D","add", "donut"]

for item in L:
 if "d" in item:
    print("Yes, d is in ", item, " in the list ", L)
```

15.4 More List Methods

15.4.1 count

We use the count method to find the number of occurrences of an item in the list.

Example

```
L = ["a","b","b", "D","add", "donut"]

print(L)
print("a count:",L.count("a"))
print("b count:",L.count("b"))
print("c count:",L.count("c"))
print("d count:",L.count("d"))
```

15.4.2 index

We can find the location of an item in a list by using `index`

Example

```
L = ["a","b","b", "D","add", "donut"]

print(L)
print("a location",L.index("a"))
print("b location",L.index("b"))
```

15.4.3 insert

We can add an item at a specific location by using `insert`

Example

```
L = ["a","b","b", "D","add", "donut"]
print(L)

L.insert(1,"z")
print(L)

L.insert(0,"z") # adds z to the start of the list
print(L)

L.insert(-1,"z") # adds z just before the end of list
print(L)
```

15.4.4 remove

We can remove an item by *name* by using `remove`

Example

```
L = ["a","b","b", "D","add", "donut"]
print(L)

L.remove("D")
print(L)
```

15.4.5 pop by location

We can remove an item by *location* by using `pop`

Example

```
L = ["a","b","b", "D","add", "donut"]
print(L)
L.pop() # remove last item in list
print(L)
L.pop(3) # remove item at position 3
         # first item is at position 0
print(L)
```

15.5 Example: Perfect Numbers

A **perfect number** is equal to the sum of all of its divisors (except itself).

Example

Note that 6 is a perfect number since

$$6 = 1 + 2 + 3$$

and the divisors of 6 are 1, 2, 3, 6 .

We can list the divisors of a number as follows.

Example

```
n = 6

for i in range(1,n):
    if n % i == 0:
        print(n, " has divisor ",i)
```

Now let's put the divisors in a list

Example

```
n = 6

divisors = []

for i in range(1,n):
    if n % i == 0:
        divisors.append(i)

print("The divisors of ",n," are")
print(divisors)
```

and now make this a function

Example

```
n = 6

def finddivisors(n):
    divisors =[]
    for i in range(1,n):
      if n % i == 0:
        divisors.append(i)
    return divisors

mydivisors = finddivisors(n)

print("The divisors of ",n," are")
print(mydivisors)
```

We now write a function to test if a number is perfect

Example

```
n = 6

def finddivisors(n):
    divisors =[]
    for i in range(1,n):
      if n % i == 0:
        divisors.append(i)
    return divisors

def testperfect(n):
    divisors = finddivisors(n)
    sum = 0
    for i in divisors:
        sum = sum + i
    if sum == n:
        return True
    else:
        return False

if testperfect(n):
    print(n ,"is a perfect number.")
    print("Its divisors are ",finddivisors(n))
```

Finally, let's find all perfect numbers up to 10000

Example

```
def finddivisors(n):
    divisors =[]
    for i in range(1,n):
      if n % i == 0:
        divisors.append(i)
    return divisors

def testperfect(n):
    divisors = finddivisors(n)
    sum = 0
    for i in divisors:
        sum = sum + i
    if sum == n:
        return True
    else:
        return False

for n in range(1,10000):
  if testperfect(n):
    print(n ,"is a perfect number.")
    print("Its divisors are ",finddivisors(n))
    print("Looking for more ...")

print("Done.")
```

Chapter 16

Tuples

Tuples are like lists, but their elements cannot be changed.

We use round brackets `()` instead of square brackets

Example

```
T = (1,2,3)

print(T)
print(T[0])
```

Note

The elements of a tuple cannot be changed and so

```
T = (1,2,3)
T[0]=2
```

generates an error.

16.1 Advanced: Sorting a list of objects

We can sort a list of objects by indicating which property of the objects that we want to sort by. For example, a list of tuples

```
L = [("c",20), ("a",1)]
```

can be sorted on the first element (string), or the second element (number). We use a function (key) to indicate (extract) the data that should be used to sort the list.

Example

```
def f(myobject):
   return myobject[0] # extract string  (the first element)

L = [("c",20), ("a",1)]

for i in L:
  print(f(i))  # only prints first element
```

Now we can sort (by the first element) using the function f as the key.

Example

```
def f(myobject):
   return myobject[0] # extract string  (the first element)

L = [("c",3), ("a",4), ("b", 1)]

print(L)
L.sort(key = f)
print(L)
```

We can sort by the second element by changing the function f as follows:

Example

```
def f(myobject):
   return myobject[1] # extract number  (the second element)

L = [("c",3), ("a",4), ("b", 1)]

print(L)
L.sort(key = f)
print(L)
```

Note

Python is famous for writing a lot of code in one line, and we don't want to have to define the function f all the time, and so we can use something called a **lamda function** (embedded in the sort command) to remove the need for creating a separate function f.

```
L = [("c",3), ("a",4), ("b", 1)]

print(L)
L.sort(key = lambda x: x[0])
print(L)
```

In the above code `x` is a dummy variable that represents an element of the list `L` and the lambda function returns `x[0]`.

So `lambda x: x[0]` is a quick way of creating the function `f`. The input is `x` (an element of the list) and the output is `x[0]`.

Chapter 17

Dictionaries

Dictionaries are like lists, but we can access the items with a `key`, and the items are stored in `key:value` pairs.

We use curly braces `{}` to create a dictionary

Example

```
Car = {"colour":"white", "speed":100, "direction":"North"}
print(Car)
print(Car["speed"])
Car["speed"] = 60
print(Car)
```

We can add to a dictionary just by assigning a new key

Example

```
Car = {"colour":"white", "speed":100, "direction":"North"}
print(Car)
Car["Weight"]=1000
print(Car)
```

We can iterate through the dictionary with a for loop

Example

```
Car = {"colour":"white", "speed":100, "direction":"North"}
for key in Car:
   print(key, "=", Car[key])
```

Chapter 18

Math Module

To use math functions in Python, we need to import the **math module**.

To do this, we just write

```
import math
```

at the top of our Python script.

18.1 Factorial

$$n! = n \times (n-1) \times \ldots 2 \times 1$$

In Python, the command for factorial is `math.factorial`

Example

```
import math

print("3! = ", math.factorial(3))
```

Note

To avoid writing `math.` all the time, we can write

```
from math import *

print("3! = ", factorial(3))
```

Then all the math functions can be written without the `math.` prefix.
If we only want to make `factorial` available, then we can write

```
from math import factorial
```

(this is useful when we want to use some variable names that could be in the math library).

18.2 Absolute Value

$$|x| = \begin{cases} -x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

In Python, we use `math.fabs` (always returns a float) or `abs`

Example

```
from math import *

print("|3| = ", fabs(3))

print("|-3| = ", fabs(-3))
```

18.3 Euler's number e

We can evaluate e and e^x as follows.

Example

```
from math import *

print("e = ", exp(1))

print("e*e = ", exp(2))

x = 1.246
print("e^x = ", exp(x))
```

18.4 logs

$$\log_e x \quad \text{and} \quad \log_b x$$

Example

```
from math import *

print("log_e (2) = ", log(2))

print("log_2 (8) = ", log(8,2))
```

18.5 Trigonometry

$$\sin x \quad \text{and} \quad \cos x \quad \text{and} \quad \tan x$$

All angles are in **radians**.

Example

```
from math import *

print("sin (2) = ", sin(2))

print("cos (8) = ", cos(8))

print("tan (1) = ", tan(1))
```

18.6 Inverse Trigonometry

$$\sin^{-1} x \quad \text{and} \quad \cos^{-1} x \quad \text{and} \quad \tan^{-1} x$$

Example

```
from math import *

print("inv sin (.4) = ", asin(.4))

print("inv cos (.4) = ", acos(.4))

print("inv tan (1) = ", atan(1))
```

18.7 Pi

The mathematics constant π is `math.pi` in Python.

Example

```
from math import *

print("pi = ", pi)
print("sin(pi) = ", round(sin(pi),10))
```

18.8 Solving Equations Part 2

We can now improve our program which solves equations.

Example

```
def f(x):
    return x**2+3*x+1

x1 = -1.0
x2 = 0.0
while abs(x1-x2) > 0.00000001:
 xa = (x1+x2)/2
 if f(xa) > 0:
     x1=x1 # so x1 does not change!
     x2=xa
 if f(xa) < 0:
     x1=xa
     x2=x2 # so x2 does not change!

print("f(x)=0 when x = ",round(xa,8)," to 8 decimal places")
```

First let's create a function which solves $f(x) = 0$

Example

```
def f(x):
    return x**2+3*x+1

def solve(x1,x2):
    while abs(x1-x2) > 0.00000001:
      xa = (x1+x2)/2
      if f(xa) > 0:
        x1=x1 # so x1 does not change!
        x2=xa
      if f(xa) < 0:
        x1=xa
        x2=x2 # so x2 does not change!
      if f(xa) == 0:
          return xa
    return xa

x1 = -1.0 # need f(x1) < 0 and f(x2) > 0
x2 = 0.0

xs = solve(x1,x2)
print("f(x)=0 when x = ",round(xs,8)," to 8 decimal places")
```

Notice that we can access the function `f(x)` inside the function `solve(x1,x2)` *without* declaring `f(x)` as 'global'.

Now let's try to find all solutions

Example

```
def f(x):
    return x**2+3*x+1

def solve(x1,x2):
    while abs(x1-x2) > 0.00000001:
      xa = (x1+x2)/2
      if f(xa) > 0:
        x1=x1 # so x1 does not change!
        x2=xa
      if f(xa) < 0:
        x1=xa
        x2=x2 # so x2 does not change!
      if f(xa) == 0:
          return xa
    return xa
for i in range(-10,10):
    for j in range(-10,10):
        if (f(i) < 0) and (f(j) > 0):
            xs = solve(i,j)
            print("f(x)=0 when x = ",round(xs,6)," to 6 decimal places")
```

A `for` loop cannot use smaller steps than 1, and if we need a finer grid (some solutions might be closer together than 1.0), then we should use a while loop.

Example

```
def f(x):
    return x**2+3*x+1

def solve(x1,x2):
    while abs(x1-x2) > 0.00000001:
      xa = (x1+x2)/2
      if f(xa) > 0:
        x1=x1 # so x1 does not change!
        x2=xa
      if f(xa) < 0:
        x1=xa
        x2=x2 # so x2 does not change!
      if f(xa) == 0:
          return xa
    return xa

xi = -10

while xi < 10:
    xi = xi + 0.5
    xj = -10
    while xj < 10:
        xj = xj + 0.5
        if (f(xi) < 0) and (f(xj) > 0):
            xs = solve(xi,xj)
            print("f(x)=0 when x = ",round(xs,6)," to 6 decimal places")
```

Now let's use a list to store the solutions and avoid repeated solutions.

Example

```
def f(x):
    return x**2+3*x+1

def solve(x1,x2):
    while abs(x1-x2) > 0.00000001:
      xa = (x1+x2)/2
      if f(xa) > 0:
        x1=x1 # so x1 does not change!
        x2=xa
      if f(xa) < 0:
        x1=xa
        x2=x2 # so x2 does not change!
      if f(xa) == 0:
          return xa
    return xa

L = []

xi = -10

while xi < 10:
    xi = xi + 0.5
    xj = -10
    while xj < 10:
        xj = xj + 0.5
        if (f(xi) < 0) and (f(xj) > 0):
            xs = round(solve(xi,xj),6)
            if (xs in L) == False:
              L.append(xs)

L.sort()
print("The solutions, to 6 d.p., are")
print(L)
```

Exercise. Use the above program to solve the following Maths 1 equations

1. $x^4 + 10x^3 + 35x^2 + 50x + 24 = 0$
2. $|2x + 1| = |2 + x|$
3. $2\cos\left(2x + \frac{\pi}{2}\right) - 1 = 0 \quad$ where $\quad x \in [0, 2\pi]$

Note. This program will not work if the graph has asymptotes, or just touches the $x-$axis without passing through.

18.9 Drawing the Graph of a Function

Example

```
import turtle
import math

turtle.speed(0)

def f(x):
    return x**4+10*x**3+35*x**2+50*x+24

# x-axis
turtle.penup()
turtle.goto(-200,0)
turtle.pendown()
turtle.goto(200,0)
turtle.write("x")
turtle.stamp()

# y-axis
turtle.penup()
turtle.goto(0,-200)
turtle.pendown()
turtle.goto(0,200)
turtle.write("y")
turtle.setheading(90)
turtle.stamp()
turtle.penup()


x = -6
turtle.goto(x*30,20*f(x))
turtle.pendown()
while x < 2:
 turtle.goto(x*30,20*f(x))
 turtle.dot()
 x = x + 0.1

turtle.hideturtle()
```

Chapter 19

Random Numbers (again)

The random numbers chosen by Python should have a **Uniform** distribution, and so if we store the frequencies in a list, then the frequencies should almost be the same.

First, let's roll a die 20 times

Example

```
from random import *

for i in range(20):
    n = randint(1,6)
    print(n)
```

Now use the list `freq` to store the frequencies

Example

```
from random import *

freq = [0,0,0,0,0,0,0]

for i in range(20):
    n = randint(1,6)
    freq[n] = freq[n] + 1
    print(n,freq)
```

and finally print the percentage frequencies

Example

```
from random import *

freq = [0,0,0,0,0,0,0]

rolls = 10000

for i in range(rolls):
    n = randint(1,6)
    freq[n] = freq[n] + 1

print(freq)

for i in range(1,7):
  print(i," appears in ",round(freq[i]/rolls*100,2), "% of the rolls")
```

If we count **sum** of two dice when they are rolled, then the frequencies don't have a uniform distribution; in fact the distribution is close to the **Normal** distribution.

Example

```
from random import *

freq = []
for i in range(18):
  freq.append(0)


rolls = 100000

for i in range(rolls):
    d1 = randint(1,6)
    d2 = randint(1,6)
    n = d1+d2
    freq[n] = freq[n] + 1

print(freq)

for i in range(1,13):
  print(i," appears in ",round(freq[i]/rolls*100,2), "% of the rolls")
```

Chapter 20

Two Dimensional Lists (Matrices)

We can use lists inside lists to make a two dimensional list (matrix).

Example

```
M = [[1,2,3],[4,5,6],[7,8,9]]  # [row 0,row 1,row 2]
print(M)
```

We can print the whole matrix

Example

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

and we can include the row numbers as follows

Example

```
M = [[1,2,3],[4,5,6],[7,8,9]]
print("The matrix is")
j = 0
for i in M:
 print(i, "   (row ",j,")")
 j = j + 1
```

Note

The element in the ith row and jth column of M is `M[i][j]` where i and j start at zero.

For example,

```
M = [[1,2,3],[4,5,6],[7,8,9]]
print(M[1][2])  # row 1, column 2. Remember these are zero-based!
                # M[1][2] = 6
```

The size of a matrix is $m \times n$ where m is the nunber of rows and n in the number of columns.

Example

```
M = [[1,2],[4,5],[7,8]]

m = len(M)
n = len(M[0])

print("The size of M is ",m," x ",n)
```

Now let's make this into two functions

Example

```
def numrows(M):
    return len(M)

def numcols(M):
    return len(M[0])

M = [[1,2],[4,5],[7,8]]

print("The size of M is ",numrows(M)," x ",numcols(M))
```

20.1 Adding Matrices

We can add two matrices if they have the same size; in this case we just add the corresponding entries in each row and column. Before we do this, we need to create a "sum" matrix of the same size to store the sum.

Example

```
M = [[7,8],[0,1]]
N = [[1,2],[3,4]]

sum_matrix = [[0,0],[0,0]]

for i in range(2):
  for j in range(2):
      sum_matrix[i][j] = M[i][j] + N[i][j]
print("M =   ", M)
print("N =   ", N)
print("M+N = ", sum_matrix)
```

We can get Python to generate a sum matrix filled with zeros with the following function.

Example

```
def creatematrix(numrows,numcols):
    # create a matrix filled with zeros
    Matrix = []
    for i in range(numrows):
        row = []
        for j in range(numcols):
            row.append(0)
        Matrix.append(row)
    return Matrix

sum_matrix = creatematrix(2,2)
print(sum_matrix)
```

Now we can use this to write the following general code for adding two matrices.

Example

```
def numrows(M):
    return len(M)

def numcols(M):
    return len(M[0])

def creatematrix(numrows,numcols):
    # create a matrix filled with zeros
    Matrix = []
    for i in range(numrows):
        row = []
        for j in range(numcols):
            row.append(0)
        Matrix.append(row)
    return Matrix

M = [[7,8],[0,1]]
N = [[1,2],[3,4]]

sum_matrix  = creatematrix(numrows(M),numcols(M))

if (numrows(M) == numrows(N)) and (numcols(M) == numcols(N)):
    for i in range(numrows(M)):
        for j in range(numcols(M)):
           sum_matrix[i][j] = M[i][j] + N[i][j]
    print("M =   ", M)
    print("N =   ", N)
    print("M+N = ", sum_matrix)
else:
    print("Incompatible size for sum")
```

Now let's make the function `addmatrices()`

Example

```
def numrows(M):
    return len(M)

def numcols(M):
    return len(M[0])

def creatematrix(numrows,numcols):
    NEW = []
    for i in range(numrows):
        row = []
        for j in range(numcols):
            row.append(0)
        NEW.append(row)
    return NEW

def addmatrices(M,N):
    SUM = creatematrix(numrows(M),numcols(M))
    for i in range(numrows(M)):
        for j in range(numcols(M)):
           SUM[i][j] = M[i][j] + N[i][j]
    return SUM

M = [[7,8],[0,1]]
N = [[1,2],[3,4]]

if (numrows(M) == numrows(N)) and (numcols(M) == numcols(N)):
    SUM = addmatrices(M,N)
    print("M =   ", M)
    print("N =   ", N)
    print("M+N = ", SUM)
else:
    print("Incompatible size for sum")
```

20.2 Multiplying Matrices

We multiply two matrices M and N by multiplying each row of M into each column of N. For this to work, we need

$$\text{number of columns}(M) = \text{number of rows}(N)$$

Example

```
def numrows(M):
    return len(M)

def numcols(M):
    return len(M[0])

def creatematrix(numrows,numcols):
    NEW = []
    for i in range(numrows):
        row = []
        for j in range(numcols):
            row.append(0)
        NEW.append(row)
    return NEW

def multiplymatrices(M,N):
    PRODUCT = creatematrix(numrows(M),numcols(N))
    for i in range(numrows(M)):
        for j in range(numcols(N)):
            PRODUCT[i][j] = 0
            for k in range(numcols(M)):
               PRODUCT[i][j] = PRODUCT[i][j] + M[i][k]*N[k][j]
    return PRODUCT

M = [[7,8],[0,1]]
N = [[1,2],[3,4]]

if (numcols(M) == numrows(N)):
    PRODUCT = multiplymatrices(M,N)
    print("M =   ", M)
    print("N =   ", N)
    print("MxN = ", PRODUCT)
else:
    print("Incompatible size for product")
```

20.3 Copying Matrices

There is a `copy` method for lists, but it only **copies one level deep** and we need to copy two levels deep (a lists of lists).
Fortunately, there is a `deepcopy` command inside the `import copy` module.

Example

```
import copy

M = [[7,8],[0,1]]
N = [[1,2],[3,4]]

NEW = copy.deepcopy(N)

print("N = ",N)
print("NEW = copy.deepcopy(N) = ",NEW)

print("Changing NEW")
NEW[0][0] = 0

print("changed NEW = ",NEW)
print("N = ",N)
```

Note

If we just write `NEW = N` then `NEW` and `N` will point to the same list and so changing `NEW` will make the same change to `N`

Chapter 21

Print Format

Python supports the C method of printing (and formating) strings.

Example

```
print("This %s computer has %d cores" % ("windows", 4))
```

where

```
%s stands for a string to be inserted
%d stands for a number to be inserted
%f stands for a float  to be inserted
```

We can use extra agruments to make sure that our numbers fit in columns. Notice that the following numbers do not line up in columns.

Example

```
print("Three numbers: %f %f %f" % (1, -42, 3))
print("Three numbers: %f %f %f" % (45, -42, 300))
```

Now let's print the numbers to 2 decimal places, with a minimum width of 10 characters.

Example

```
print("Three numbers: %10.2f %10.2f %10.2f" % (1, -42, 3))
print("Three numbers: %10.2f %10.2f %10.2f" % (45, -42, 300))
```

We can use this to print out a matrix with correctly aligned columns.

First note that we can print without ending with a new line by using `print(text,end="")`

Example

```
print("Hello ",end="")
print("there",end="")
```

21.1 Printing a Matrix

Now we can make a function to print a matrix.

Example

```
def numrows(M):
    return len(M)

def numcols(M):
    return len(M[0])

def printmatrix(M):
    for i in range(numrows(M)):
        print("[",end="")
        for j in range(numcols(M)):
            print("%10.2f" % M[i][j], end="")
        print("]")


M = [[-7.03,800],[0.001,-1]]
printmatrix(M)
```

21.2 format

Python also has its own formating method for formating strings called `.format`

Example

```
print("This {0} computer has {1} cores {2}".format("windows", 4, "Sam"))
```

This makes it easy to repeat and mix parameters

Example

```
print("This {0} computer has {0} cores {0}".format("windows", 4, "Sam"))
```

and the `.format` method will return a string

Example

```
s = "This {0} computer has {0} cores {0}".format("windows", 4, "Sam")
print(s)
```

and we can still specify the number of decimal places and minimum width

Example

```
print("Three numbers: {0:10.2f} {1:10.2f} {2:10.2f}".format(1, -42, 3))
print("Three numbers: {0:10.2f} {1:10.2f} {2:10.2f}".format(45, -42, 300))
```

Chapter 22

Breaking Out of a Loop

22.1 break

The `break` command breaks out of a loop. This command is not needed in most programs and is even considered "bad programming" by some programmers, but many people still use it. It replaces some of the `goto` statements used in BASIC programs many years ago.

Example

```
for i in range(100):
  if i == 20:
    break
  print(i)

print("i = ",i)
```

Note that the `break` command only breaks out of the current loop. If it is in a 'double loop' then the break command breaks into the outermost loop.

Example

```
while True:
 for i in range(100):
   if i == 20:
     break
   print(i)
 print("in while loop")

print("i = ",i)
```

The `break` command will also break out of a `while` loop.

Example

```
j = 1
while True:
 j = j + 1
 for i in range(100):
   if i == 20:
     break
   print(i)
 print("in while loop")
 if j == 7:
     break

print("i = ",i)
print("j = ",j)
```

22.2 continue

The command `continue` jumps to the top of the enclosing loop without running the following commands in the loop, and so `continue` also replaces some of the `goto` statements used in BASIC programs many years ago.

Example

The following program will print the numbers 1 to 9 excluding 3.

```
for i in range(1,10):
  if i==3:
      continue
  print(i)
```

Chapter 23

Files

Variables give us **temporary** storage (all values are lost when the program ends).

Files give us **permanent** storage, since the files are stored in our C: drive (solid state drive).

Example

To make a file called `test.txt`, we can use the following code.

```
myfile = open("test.txt","w")
myfile.write("This is a test\n")
myfile.write("Second line of a test of files\n")
myfile.close()
print("file test.txt written to SSD")
```

Note

- `"w"` means "write to the file"
- `\n` means "start a new line"
- The file `test.txt` will be located in the same directory as our Python file.

To read the file, we use `readline()`

Example

```
myfile = open("test.txt","r")
line1 = myfile.readline()
line2 = myfile.readline()
myfile.close()
print("Line1:",line1)
print("Line2:",line2)
```

Notice that the data is permanently stored on our C: drive.

Note

When specifying a path we should use a raw string:

```
open(r"C:\Mydir\myfile.txt")
```

The `r` in front of `r"C:\Mydir\myfile.txt"` tells Python to read the string exactly as it is, without escape characters like `\n`

We can also use a `for loop` to read all the strings in a file.

Example

```
myfile = open("test.txt","r")
for line in myfile:
 print(line)
myfile.close()
```

Now let's access each word

Example

```
myfile = open("test.txt","r")
for line in myfile:
  mywords = line.split()
  print(mywords)
myfile.close()
```

and let's store all words in one list.

Example

```
myfile = open("test.txt","r")
allwords = []
for line in myfile:
  mywords = line.split()
  for w in mywords:
      allwords.append(w)
myfile.close()
print(allwords)
```

Finally, let's use a `dictionary` to store the words together with the number of times (frequency) that the word appears in the file.

Example

```
myfile = open("test.txt","r")
allwords = []
for line in myfile:
  mywords = line.split()
  for w in mywords:
      allwords.append(w)
myfile.close()
print(allwords)

dict = {}
for w in allwords:
  dict[w] = allwords.count(w)
print(dict)
```

The word `w` is the `key` of our dictionary `dict` and the corresponding `value` of each key is the number of times that the word appears.

Finally, let's store the frequencies and words in a list of strings so that we can sort it to find the most common words.

Example

```
myfile = open("test.txt","r")
allwords = []
for line in myfile:
  mywords = line.split()
  for w in mywords:
      allwords.append(w)
myfile.close()
print(allwords)

dict = {}
for w in allwords:
  dict[w] = allwords.count(w)
print(dict)

countlist = []

for key in dict:
  countlist.append("{0:10.0f}".format(dict[key])+":"+key)

countlist.sort()
countlist.reverse()
for w in countlist:
   print(w)
```

Chapter 24

Try: Except: Else:

We can "capture" exceptions to our running code to avoid the program stopping unexpectedly.

A common way for an exception to occur is when the user is supposed to enter an integer, but enters something else. We can prevent the program from "crashing" by capturing the exception.

Example

```
try:
    n = int(input("Input your integer: "))
except:
    print("You did not enter an integer!")
else:
    print("Your number is: ",n)
```

In general, the code looks like this

```
try:
   # code that might raise an exception
except ExceptionType:
   # code to handle the exception
else:
   # code to run if there is no exception
```

Some of the exceptions that you can use are:

- **Exception (or blank)**: This covers all types of exception.
- **ValueError**: Raised when a built-in operation or function receives an argument that has the right type but an inappropriate value.
- **TypeError**: Raised when an operation or function is applied to an object of inappropriate type.
- **IndexError**: Raised when an index is not found in a sequence or list.
- **KeyError**: Raised when a dictionary key is not found in the set of existing keys.
- **ZeroDivisionError**: Raised when division or modulo by zero takes place for any numeric type.
- **AttributeError**: Raised when an attribute reference or assignment fails because the named attribute does not exist.
- **IOError**: Raised when an I/O operation (such as reading or writing to a file) fails for an I/O-related reason.

Example

```
L = ["a","b"]

try:
    d = L[2]
except:
    print("Index error")
else:
    print(d)
```

Chapter 25

Objects

An **object** is a structure that contains variables and functions. The functions inside the object are called **methods**.

We use a **class** to build a template for an object.

Then we create (many) instances of the object from the class.

25.1 Classes

We can define a class `Quadratic` with three instance variables a, b and c as follows.

Example

```
class Quadratic:  # create a class for ax^2+bx+c
      def __init__(self,a=0,b=0,c=0):
          self.a = a   # store coefficients with instance variables a,b,c
          self.b = b
          self.c = c
```

25.2 Object instances

We can make an object instance `q` of the class by just writing `q = Quadratic(1,2,3)`

Then `q` is an object that contains variables $a = 1, b = 2, c = 3$.

If we just write `q = Quadratic()` then the variables with have the default values $a = 0, b = 0$ and $c = 0$.

Example

```
class Quadratic:  # create a class for ax^2+bx+c
      def __init__(self,a=0,b=0,c=0):
          self.a = a   # store coefficients  with instance variables a,b,c
          self.b = b
          self.c = c

q = Quadratic(1,3,1)  # q is an instance of the class Quadratic
t = Quadratic()  # t is another instance of the class Quadratic

print(q.a)
print(q.b)
print(q.c)

print(t.a)
print(t.b)
print(t.c)
```

Note

In the definition of the class, `self` refers to the actual object instance that we are creating, and so when we write `q = Quadratic(1,2,3)` we have `self = q`

25.3 Methods

We can attach a method (function) to the object, as follows:

Example

```
class Quadratic:  # create a class for ax^2+bx+c
      def __init__(self,a=0,b=0,c=0):
          self.a = a   # store coefficients  with instance variables a,b,c
          self.b = b
          self.c = c
      def display(self):
          print(self.a,"x^2 +",self.b,"x +",self.c)

q = Quadratic(1,3,1)  # q is an instance of the class Quadratic
q.display()
```

Example

The function definition `def display(self):` includes the parameter `self` so that we can access the object instance variables `self.a`, etc.

Finally, let's add methods for solving the quadratic equation.

Example

```
class Quadratic:  # create a class for ax^2+bx+c
      def __init__(self,a=0,b=0,c=0):
          self.a = a   # store coefficients  with instance variables a,b,c
          self.b = b
          self.c = c
      def display(self):
          print(self.a,"x^2 +",self.b,"x +",self.c)
      def root1(self):
          return (-1*self.b + (self.b*self.b-4*self.a*self.c)**0.5)/(2*self.a)
      def root2(self):
          return (-1*self.b - (self.b*self.b-4*self.a*self.c)**0.5)/(2*self.a)
      def solve(self):
          print("x = ",self.root1(), "  or   x = ", self.root2())

q = Quadratic(1,3,1)  # q is an instance of the class Quadratic
t = Quadratic(1,3,2)  # t is another instance of the class Quadratic

print("q = ")
q.display()

print("First root of q")
print(q.root1())

print("Solutions to q = 0")
q.solve()

print("t = ")
t.display()

print("Solutions to t = 0")
t.solve()
```

25.4 Class (static) variables

A class variable keeps the same value for all object instances. In other languages, the variable is called **static**. Let's define a class variable called `degree` that stores the degree of all the quadratics.

Example

```
class Quadratic:  # create a class for ax^2+bx+c
      degree = 2
      def __init__(self,a=0,b=0,c=0):
          self.a = a   # store coefficients  with instance variables a,b,c
          self.b = b
          self.c = c
      def display(self):
          print(self.a,"x^2 +",self.b,"x +",self.c)

q = Quadratic(1,3,1)  # q is an instance of the class Quadratic
t = Quadratic(1,3,2)  # t is another instance of the class Quadratic

print(q.degree)
print(t.degree)

Quadratic.degree = 3  # changes both q.degree and t.degree

print(q.degree)
print(t.degree)
```

25.5 Operator Overloading

We can change the default methods (like printing) for objects. This is called **operator overloading**. To change printing, we override the `__str__` method.

Example

```
class Quadratic:  # create a class for ax^2+bx+c
      degree = 2
      def __init__(self,a=0,b=0,c=0):
          self.a = a   # store coefficients  with instance variables a,b,c
          self.b = b
          self.c = c
      def display(self):
          print(self.a,"x^2 +",self.b,"x +",self.c)
      def __str__(self):
          return str(self.a)+"x^2 +"+str(self.b)+"x +"+str(self.c)

q = Quadratic(1,3,1)  # q is an instance of the class Quadratic
t = Quadratic(1,3,2)  # t is another instance of the class Quadratic

print(q)
print(t)
```

We can even overload the + operator, so that we can add quadratics!

Example

```
 class Quadratic:  # create a class for ax^2+bx+c
      degree = 2
      def __init__(self,a=0,b=0,c=0):
          self.a = a   # store coefficients  with instance variables a,b,c
          self.b = b
          self.c = c
      def display(self):
          print(self.a,"x^2 +",self.b,"x +",self.c)
      def __str__(self):
          return str(self.a)+"x^2 +"+str(self.b)+"x +"+str(self.c)
      def __add__(self,other):
            return Quadratic(self.a+other.a,self.b+other.b,self.c+other.c)

q = Quadratic(1,3,1)  # q is an instance of the class Quadratic
t = Quadratic(1,3,2)  # t is another instance of the class Quadratic
s = q+t
print(s)
```

25.6 Inheritance

Objects can "inherit" the variables and methods of other objects.

Let's define a "parent" class called `Polynomial` for the class `Quadratic`.

Example

```
class Polynomial:
      def __init__(self,a=0,b=0,c=0,d=0,e=0,f=0):
          self.a = a   #  instance variables a,b,c,d,e,f
          self.b = b
          self.c = c
          self.d = d
          self.e = e
          self.f = f

class Quadratic(Polynomial):  # create a class for ax^2+bx+c
      degree = 2
      def __str__(self):
          return str(self.a)+"x^2 +"+str(self.b)+"x +"+str(self.c)

q = Quadratic(1,2,3)  # q is an instance of the class Quadratic
print(q)
```

The "constructor" `__init__` is defined in `Polynomial` and so `Quadratic` **inherits** this "constructor" from `Polynomial`. We don't need to define it again in `Quadratic`.

We still define a separate print method (`__str__`) for `Quadratic` since this depends on the type of polynomial.

Now we can create another class `Linear` from `Polynomial`.

Both `Linear` and `Quadratic` are called **children** of `polynomial`.

Example

```
 class Polynomial:
      def __init__(self,a=0,b=0,c=0,d=0,e=0,f=0):
          self.a = a   # store coefficients  with instance variables a,b,c
          self.b = b
          self.c = c
          self.d = d
          self.e = e
          self.f = f

class Quadratic(Polynomial):  # create a class for ax^2+bx+c
      degree = 2
      def __str__(self):
          return str(self.a)+"x^2 +"+str(self.b)+"x +"+str(self.c)

class Linear(Polynomial):  # create a class for ax+b
      degree = 1
      def __str__(self):
          return str(self.a)+"x +"+str(self.b)


q = Quadratic(1,2,3)  # q is an instance of the class Quadratic
print(q)

z = Linear(7,8)
print(z)
```

Note that `print` chooses the correct version of `__str__` to print the quadratic and linear functions correctly.

Note

We can even write

```
for obj in (q,z):
      print(obj)
```

The fact that the correct version of `__str__` gets choosen for different objects is called **polymorphism**. Python knows what type of object it is looking at, and so chooses the version of `__str__` that matches the type.

Chapter 26

Tkinter

Tkinter is used to make a program that looks and functions like a Microsoft Windows program.

For a full list of commands, have a look at

`https://www.tcl.tk/man/tcl8.6.11/TkCmd/contents.html`

26.1 Hello World

Example

```
from tkinter import *
mainwin = Tk()
mainwin.geometry("400x200")
button1 = Button(mainwin,text="Hello World")
button1.place(x=170,y=100)
mainwin.mainloop()
```

Note

Don't call your source file tkinter.py since that will stop `import tkinter` from working!!

26.2 Textbox

A textbox is used to input and display text.

Example

```
from tkinter import *
mainwin = Tk()
mainwin.geometry("400x200")
button1 = Button(mainwin,text="Hello World")
button1.place(x=170,y=100)

textbox1 = Text(mainwin,width=18,height=10)
textbox1.place(x=10,y=10)
mainwin.mainloop()
```

Note that `width` is the width of the textbox in characters (letters), and similarly for `height`.

26.2.1 Inserting text into a textbox

Use the `insert` command to put text inside a textbox.

Example

```
from tkinter import *
mainwin = Tk()
mainwin.geometry("400x200")
button1 = Button(mainwin,text="Hello World")
button1.place(x=170,y=100)

textbox1 = Text(mainwin,width=18,height=10)
textbox1.place(x=10,y=10)
textbox1.insert(INSERT,"hello there")
mainwin.mainloop()
```

26.2.2 Reading text from a textbox

Use the `get` command to store the text inside a variable.

Example

```
from tkinter import *
mainwin = Tk()
mainwin.geometry("400x200")
button1 = Button(mainwin,text="Hello World")
button1.place(x=170,y=100)

textbox1 = Text(mainwin,width=18,height=10)
textbox1.place(x=10,y=10)
textbox1.insert(INSERT,"hello there")

mytext = textbox1.get("1.0","end-1c")  # start line 1, and char 0, and stop at end
print(mytext)
mainwin.mainloop()
```

Note

To read the lines one by one, convert the string to a list of lines as follows:

```
mylines = textbox1.get("1.0","end-1c").splitlines()
for line in mylines:
    print(line)
```

26.3 Buttons

We use the `command` parameter to tell the button which function to call when the button is pressed.

Example

```
from tkinter import *
mainwin = Tk()
mainwin.geometry("400x200")

textbox1 = Text(mainwin,width=18,height=10)
textbox1.place(x=10,y=10)
textbox1.insert(INSERT,"hello there")

mytext = ""
def readtext():
    global mytext
    mytext = textbox1.get("1.0","end-1c")  # start line 1, and char 0, and stop at end
    print(mytext)

button1 = Button(mainwin,text="Hello World", command=readtext)
button1.place(x=170,y=100)
mainwin.mainloop()
```

26.4 Drawing on a canvas

Example

```
from tkinter import *
mainwin = Tk()
mainwin.geometry("400x200")

canvas1 = Canvas(mainwin, width=400, height = 100, background="black")
canvas1.place(x=0,y=100)
canvas1.create_line(10,10,400,100,fill="red",width=3)
mainwin.mainloop()
```

26.5 Drawing Text

Example

```
from tkinter import *
from time import *
mainwin = Tk()
mainwin.geometry("800x400")
font1 = ("Arial",20)
canvas1 = Canvas(mainwin, width=800, height=400)
canvas1.place(x=0,y=0)
mytext = canvas1.create_text(100,100,text="Some Text",fill="blue",font=font1)
mainwin.mainloop()
```

26.5.1 Changing Text

Example

```
from tkinter import *
from time import *
mainwin = Tk()
mainwin.geometry("800x400")
font1 = ("Arial",20)
canvas1 = Canvas(mainwin, width=800, height=400)
canvas1.place(x=0,y=0)
mytext = canvas1.create_text(100,100,text="Some Text",fill="blue",font=font1)
canvas1.itemconfigure(mytext, text="hello")
mainwin.mainloop()
```

26.5.2 Text Alignment

Example

```
from tkinter import *
from time import *
mainwin = Tk()
mainwin.geometry("800x400")
font1 = ("Arial",20)
canvas1 = Canvas(mainwin, width=800, height=400)
canvas1.place(x=0,y=0)
mytext = canvas1.create_text(100,100,text="Some Text nw", anchor="nw")
mytext = canvas1.create_text(100,200,text="Some Text se", anchor="se")
mytext = canvas1.create_text(100,200,text="Some Text")
mainwin.mainloop()
```

26.6 Timer(clock)

Example

```
from tkinter import *
from time import *
mainwin = Tk()
mainwin.geometry("800x400")

def timer1():
  t = localtime()
  mysec  = str(t.tm_sec)
  mymin  = str(t.tm_min)
  myhour = str(t.tm_hour)
  canvas1.itemconfigure(mytext,text=myhour+":"+mymin+":"+mysec)
  mainwin.after(100,timer1)

canvas1 = Canvas(mainwin, width=800, height=400)
canvas1.place(x=0,y=0)
mytext = canvas1.create_text(100,100,text="time",fill="blue")
timer1()

mainwin.mainloop()
```

26.7 Example: Clock with Reminder Text

```
from tkinter import *
import os
import sys
import time
import datetime

def timer1():
       t  = time.localtime()
       mysec = str0(t.tm_sec)
       mymin = str0(t.tm_min)
       myhour = str0(t.tm_hour)
       canvas.itemconfigure(mt, text=myhour+":"+mymin+":"+mysec  )
       today = datetime.date.today()
       canvas.itemconfigure(mt3,text=today.strftime("%d %B %Y"))
       mainwin.after(100, timer1)

def str0(num):
       if num <= 9:
          return "0"+str(num)
       else:
          return str(num)

mainwin = Tk()
mainwin.configure(bg="black")
mainwin.geometry("400x600")

font1 = ("Arial",24)
font2 = ("Arial",60)

canvas = Canvas(mainwin, width=400, height = 400, bg = "black")
canvas.place(x=0,y=0)   # we draw on this canvas
mt=canvas.create_text(200,34,text="time", font=font2, fill = "yellow")
mt3=canvas.create_text(200,80,text="date", font=font1, fill = "yellow")

textbox = Text(mainwin, width =22, height=12, bg="black", fg="white",\
font = font1, insertbackground = "yellow")
textbox.place(x=0,y=100)
```

```
# load reminder text from file
try:
   myfile = open(os.path.join(sys.path[0], "Rem2.txt"),"r")
      # sys.path[0] is local folder directory
except:
    textbox.insert(INSERT,"empty")
else:
    for myline in myfile:
            textbox.insert(INSERT,myline)
    myfile.close()

def onclickChangeMade(event):
      buttonSave.config(text="Save")

def onclickSave():
      myfile = open(os.path.join(sys.path[0], "Rem2.txt"),"w")
          # sys.path[0] is local folder directory
      myfile.write(textbox.get(1.0,"end-1c"))
      myfile.close()
      buttonSave.config(text="Saved")

buttonSave = Button(mainwin,text="Save",command=onclickSave, bg="grey",\
fg ="white", font = font1)
buttonSave.pack(side = BOTTOM, fill = X)
# place button at bottom of window filling the bottom

mainwin.bind("<Key>",onclickChangeMade)
mainwin.after(100, timer1)
mainwin.mainloop()
```

26.8 Drawing external images (png)

Example

```
from tkinter import *
mainwin = Tk()
mainwin.geometry("800x400")
mypic = PhotoImage(file="image1.png")
canvas1 = Canvas(mainwin, width=mypic.width(), height = mypic.height())
canvas1.create_image(0,0,anchor=NW,image=mypic)
canvas1.place(x=120,y=0) # move whole canvas to move image :)
mainwin.mainloop()
```

26.9 Animation

Example

```
from tkinter import *
from time import *
mainwin = Tk()
mainwin.geometry("800x400")
mypic = PhotoImage(file="image1.png")
mypic2 = PhotoImage(file="image2.png")
canvas1 = Canvas(mainwin, width=800, height=400)
sprite  = canvas1.create_image(0,0,anchor=NW,image=mypic)
canvas1.place(x=0,y=0)
for i in range(100):
  canvas1.move(sprite,1,0)
  canvas1.update()
  sleep(0.01)
  if i == 50:
    canvas1.itemconfigure(sprite,image=mypic2)
  print(canvas1.coords(sprite)[0])
mainwin.mainloop()
```

Chapter 27

Copy/Paste from clipboard

Tkinter supports copy/paste to/from clipboard.

Example

```
from tkinter import *

mainwin = Tk()

text_box = Text(mainwin, height=10, width=50)
text_box.pack()


def copy():
    selected_text = text_box.get(SEL_FIRST, SEL_LAST)
    mainwin.clipboard_clear()
    mainwin.clipboard_append(selected_text)

def paste():
    clipboard_text = mainwin.clipboard_get()
    text_box.insert(INSERT, clipboard_text)

copy_button = Button(mainwin, text="Copy", command=copy)
copy_button.pack()


paste_button = Button(mainwin, text="Paste", command=paste)
paste_button.pack()

mainwin.mainloop()
```

Chapter 28

Reading/Writing csv spreadsheets

When reading and writing simple spreadsheets with Python, we would normally use the "csv" format. Most spreadsheet programs have an "export to csv" function which writes the spreadsheet as a text file where columns are separated by a comma (,) and each row starts on a new line.

Example

The spreadsheet

Excel				
	A	B	C	D
1	1	2	3	
2	4	5	6	
3	7	8	9	
4				

can be exported to a csv file which looks like

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

Suppose that we have a csv file called `test.csv` in the **same** directory (location) as our python program, and suppose that `test.csv` has the 3 lines

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

28.1 Reading a Spreadsheet

To read the above spreadsheet we write

```
import csv

csvfile = open('test.csv','r')
csvreader = csv.reader(csvfile)
for row in csvreader:
  print(row)
csvfile.close()
```

Note that `csvreader` is a list of rows, where each row is another list (of entries in the row).

28.2 Reading Individual Entries

The following example shows how to read individual entries.

Example

```
import csv

csvfile = open('test.csv','r')
csvreader = csv.reader(csvfile)
csvmatrix = list(csvreader)
rownum = 1  # starts at row 0
colnum = 2  # starts at col 0
print(csvmatrix[rownum][colnum])
csvfile.close()
```

28.3 Writing a Spreadsheet

Example

```
import csv

data = [[10,11,12],[13,14,15],[16,17,18]]

csvfile = open('test2.csv','w', newline='')
# newline='' stops blank rows from being inserted
#            after each row
csvwriter = csv.writer(csvfile)
csvwriter.writerows(data)
csvfile.close()
```

Chapter 29

Starting Python from a Terminal

We can start a Python program from the command line (Terminal, or PowerShell).

Type

```
powershell <Enter>
```

in the address bar of an open folder to open a command line at the current folder directory.

Then type

```
python myprogram.py <Enter>
```

to start myprogram.py

Example

Save the following program as `first.py`

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

Then type

```
python first.py <Enter>
```

in the command line to run the program.

Running Python programs from the command line is very helpful when the program needs to read the contents of a file. We simply enter

```
python myprogram.py test.txt <Enter>
```

to start myprogram.py with input "test.txt"

Example

Save the following program as `first.py`

```
import sys
myfilename = sys.argv[1] # this is the name we input in the command line

print("Filename = ", myfilename)

myfile = open(myfilename,"r")
for line in myfile:
    print(line)
myfile.close()
```

Save the program and type (in the command line)

```
python first.py first.py
```