Python Programming:

An Introduction to Computer Science

**Chapter 1. Computer and programs**

Objectives

* To understand the respective roles of hardware and software in computing systems
* To learn what computer scientist’s study and the techniques they use.
* To understand the basic design of modern computers
* To understand the form and function of computer programming languages
* To begin using the Python programming language
* To learn about chaotic models and their implications in computing
  1. The Universal machine

Computers – are machines that manipulate information

Computer program – Detailed, step by step set of instructions telling a computer exactly what to do.

* 1. Program power

Software (programs) rules the hardware (the physical machine)

1.3: What is computer science

Computer scientists use numerous techniques of investigation to answer this question. The three main ones are design, analysis and experimentation.

Algorithms are step-by-step process for achieving the desired result.

One weakness of design is that it can only answer the question “What is computable?” in the positive. If I can devise an algorithm, then the problem is solvable. However, failing to find an algorithm does not mean than a problem is unsolvable. It may mean that I’m not smart enough, or I haven’t hit upon the idea yet.

Analysis is the process of examining algorithms and problems mathematically.

Experimentation is often needed in order to verify and refine the analysis.

1.4: Hardware basics

The central processing unit (CPU) is the “brain” of the machine. This is where all the basic operations of the computer are carried out. The CPU can perform simple arithmetic operations like adding two numbers and can also do logical operations like testing to see if two numbers are equal.

The memory stores programs and data.

The CPU can directly access only information that is stored in main memory (called RAM for random access memory).

Main memory is fast, but it is also volatile. That is when the power is turned off, the information in the memory is lost.

In modern personal computer, the principal secondary memory I typically an internal hard disk drive (HDD) or a solid state drive (SSD). An HDD stores information as magnetic patterns or a spinning disk, while an SSD employs electronic circuits known as flash memory.

Input/output are in regards to a mouse and keyboard.

Typically the CPU follows a process called the *fetch-execute cycle.* The first instruction is retrieved from the memory, decoded to figure out what it represents and the appropriate action carried out.

1.5: Programming languages

Every structure in a programming language has a precise form (its syntax) and a precise meaning (its semantics). A programming language is something like a code for writing down instruction that a computer will follow.

A high level language can either be *compiled* or *interpreted.*

A *compiler* is a complex computer program that takes another program written in a high level language that translates it into an equivalent program in the machine language of some computer.

A high level program is called source code, and the resulting machine code is a program that the computer can directly execute.

An *interpreter* is a program that simulates a computer that understands a high – level language. Rather than translating the source program into a machine language equivalent, the interpreter analyses and executes the source code instructions by instruction as necessary.

Compiled programs tend to be faster since the translation is done once and for all, but for interpreted languages lend themselves to a more flexible programming environment as programs can be developed and run interactively.

1.6: The magic of Python

Commands can have changeable parts called parameter (also called arguments) that are placed within the parentheses.

Programs are usually created by typing definitions into a separate file called a *module* or *script*. This file is saved in secondary memory so that it can be used over and over again.

A module file I just a file of text, and you can create one using any application for editing text, such as notepad or a word processor, provided you save your program as a “plain text” file.

IDE specifically designed to help programmers write programs and includes features such as automatic indenting, colour highlighting, and interactive development IDLE is a good example.

All written in module in Python

Summary

A computer is a universal information-processing machine. It can carry out any process that can be described in sufficient detail. A description of the sequence of

Of steps for solving a particular problem is called an algorithm. Algorithm can be turned into software (programs) that determines what the hardware (physical machine) can and does accomplish. The process of creating software is called programming.

Computer science is the study of what can be computed. Computer scientists use the technique of design, analysis, and experimentation. Computer science is the foundation of the broader field of computing which includes areas such as networking, databases, and information management systems, to name a few.

A basic functional view of computer system comprises a central processing unit (CPU), main memory, secondary memory, and input and output devices. The CPU is the brain of the computer that performs simple arithmetic and logical operations. Information that the CPU acts on (data and programs) is stored in main memory (RAM). More permanent information is stored on secondary memory devices such as magnetic disks, flash memory, and optical devices. Information is entered into the computer via input devices, and output devices display the results.

Programs are written using a formal notation knows as a programming language. There are many different languages, but all share the property of having a precise syntax (form) and semantics (meaning). Computer hardware understands only a very low level language knows as machine language. Programs are usually written using human-orientated, high-level languages such as Python. A high-level language must either be compiled or interpreted in order for the computer to understand it. High-level languages are more portable than machine language.

Python is an interpreted language. One good way to lean about Python is to use an interactive shell for experimentation. The standard Python distribution includes a program called IDLE that provides a shell as well as facilities for editing Python programs.

A Python program is a sequence of commands (called statements) for the Python interpreter to execute. Python includes statements to do thing such as print output to the screen, get input from the user, calculate the value of a mathematical expression, and perform a sequence of statements multiple times (loop)

A mathematical model is called chaotic if very small changes in the input lead to large changes in the results, making them seem random or unpredictable. The models of many real-world phenomena exhibit chaotic behaviour, which places some limits on the power of computing.

**Review questions**

True/False

1. Computer science is the study of computer? FALSE
   1. Computer science is the study of what can be computed. Computer scientists use the technique of design, analysis, and experimentation. Computer science is the foundation of the broader field of computing which includes areas such as networking, databases, and information management systems, to name a few.
2. The CPU is the brain of the computer. TRUE
   1. The central processing unit (CPU) is the “brain” of the machine. This is where all the basic operations of the computer are carried out. The CPU can perform simple arithmetic operations like adding two numbers and can also do logical operations like testing to see if two numbers are equal.
3. Secondary memory is also called RAM. FALSE
   1. Secondary memory which can provide more permanent storage e.g. memory sticks. Where RAM (Random access memory) is the main memory, which is fast but volatile.
4. All information that a computer is currently working on is stored on its main memory. TRUE
   1. Some information can be stored on a secondary device which can be a memory stick however most memory is store on its internal hard disk drive (HDD) or solid state drive (SSD).
5. The syntax of a language is it’s meaning, and semantic is its form. FALSE
   1. Syntax – Precises form
   2. Semantics – Precise meaning
6. A function definition is a sequence of statement that defines a new command. TRUE
   1. A function in python is the creation of a sequence of statements
7. A programming environment refers to a place where programmers work. FALSE
   1. A programming environment is a special computer program that provides facilities to make programming easier. IDLE (in the standard Python distribution) is an example of a simple programming environment.
      1. IDE specifically designed to help programmers write programs and includes features such as automatic indenting, colour highlighting, and interactive development IDLE is a good example
8. A variable is used to give a name to a value so it can be referred to in other places. TRUE
   1. A variable is an identifier that labels a value for future reference. The value of a variable can be changed through assignment.
9. A loop is used to skip over a section of a program. FALSE
   1. A control construct for executing portion of a program multiple times.
10. A computer can’t compute a chaotic function. FALSE
    1. Chaotic function refers to the very small differences in the initial value can lead to a large differences in the result as the formula is repeatedly applied. It can be computer but the result may not be useful.

Result: 8/10

Multiple choices

1. What is the fundamental question of computer science? B.
   1. Computer science is the study of what can be computed. Computer scientists use the technique of design, analysis, and experimentation.
2. An algorithm is like a? D.
   1. Algorithms are step-by-step process for achieving the desired result
3. A problem is intractable when? D.
   1. Intractable means too difficult to be solved in practice, usually because it would take too long or require too much value to be of practical value.
4. Which is not an example of a secondary memory? A.
   1. RAM
5. Computer languages designed to be used and understood by humans are? B.
   1. Computer hardware can understand only a very low level language known a machine language. High-level language on the other hand is precise and designed to be used and understood by humans.
6. A statement is? D. Actually B.
   1. A single command in a programming language
      1. A complete computer command
7. One difference between a compiler and an interpreter is? B. Actually C
   1. Compiler – A complex program that translates a program written in a high-level language into the machine language that can be executed by a particular computer.
   2. Interpreter – A computer program that simulates the behaviour of a computer than understands a high-level language. It executes the lines of source one by one and carries out the operations.
   3. A compiler is no longer needed when a program is translated
8. By convention, the statements of a program are often placed in a function called? B.
   1. Functions are a subprogram within a program. Functions take parameters as input and can return values. However it is customary to place the instructions that comprise a program inside of a function called main.
9. Which of the following is *not* true of comments? A.
   1. Comments are intended for human readers of the program and are ignored by Python. The Python interpreter always skips any text from the pound sign (#) through the end of a line.
10. The item listed is the parentheses of a function definition are called? D.
    1. An argument is an actual parameter. A parameter is a special variable in a function that is initialized at the time of call with information passed from the caller.

Discussion

1. Compare and contrast the following pairs of concepts from the chapter:
   1. Hardware vs. Software
      1. Hardware relates to the physical components of a computing system. If it goes “crash” when you toss it out the window, that’s hardware.
      2. Software relates to computer programs.
   2. Algorithm vs Program
      1. Algorithms are detailed sequence of steps for carrying out some process. A recipe.
      2. Programs are a detailed set of instructions for a computer to carry out
   3. Programming Language vs Natural Language
      1. Programming languages is a notation for writing computer programs. Usually used to refer to high-level languages such as Python, Java, C++, etc
      2. Natural languages refer to the human language, which is fraught with ambiguity and imprecision. For example, if I say “I saw the man in the park with the telescope,” did I have the telescope, or did the man?? And who was in the park?? We understand each other most of the time only because all humans share a vast store of common knowledge and experience.
   4. High-level language vs Machine language
      1. High-level language refers to programming languages that are designed to be used and understood by humans.
      2. Machine languages are designed to be understood by computers.
   5. Interpreter vs Compiler
      1. Interpreter is a program that simulates a computer that understands a high level language. Rather than translating the source program into a machine language equivalent, the interpreter analyses and executes the source code instruction by instruction as necessary.
      2. Compiler is a complex computer program that takes another program written to a high-level language and translates it into an equivalent program in the machine language of some computer.
         1. The difference between the two is that compiling is a one shot translation; once a program is compiled, it may be run over and over again without further need for the compiler or the source code. In the interpreted case, the interpreter and the source are needed every time the program runs. Compiled program tend to be faster, since the translation is done once and for all, but interpreted languages lend themselves to amore flexible programming environment as the program can be developed and run interactively.
            1. Even more simplified is that an interpreter translates code into machine language line by line whereas a compiler does it in one shot.
   6. Syntax vs Semantics
      1. Syntax reflects the form of a language
      2. Semantics reflects the meaning of the construct
2. List and explain in your own words the role of each of the five basic functional unit of a computer depicted in Figure 1.1
   1. CPU – The brains of the machine where all the basic operations of the computer are carried out.
   2. Main memory – The great memory that gives your CPU speed, which is volatile.
   3. Secondary memory – Secondary source of memory e.g. CD’s and memory sticks
   4. Output/input devices - Mouse, keyboard that helps you direct the computer which may result in a program arising in your monitor
3. Write a detailed algorithm for making a peanut butter and jelly sandwich (or some everyday activity). You should assume that you are talking to somebody, who is conceptually able to do the task, but has never actually done it before. For example, you might be telling your young child.
   1. Grab 2 pieces of some motherfucking sourdough bread
   2. Lather crunchy peanut butter on one side of the sourdough bread
   3. Lather strawberry Jam on the other
4. As you will learn in later chapter, many of the numbers stored in a computer are not exact values, but rather close approximations. For example, the value of 0.1 might be stored as 0.1000000000555. Usually, such small differences are not a problem; however, given what you have learned about chaotic behaviour in chapter 1, you should realise the need or caution in certain situations. Can you think of example where this might be a problem? Explain.

A problem of where decimal places can have a dramatic impact on results is for when calculating discounted cash flow models for businesses. A decimal place can have a dramatic effect on the valuation of a single company that will have a weighting impact on the decision of purchasing or selling the company.

1. Trace through the chaos program from Section 1.6 by hand using 0.15 as the input value. Show the sequence of output that results.

**Chapter 2. Writing simple programs**

Objectives

* To know the steps in an orderly software development process
* To understand programs following the input, process, output (IPO) pattern and be able to modify them in simple ways
* To understand the rules for forming valid Python identifiers and expressions
* To be able to understand and write Python statement to output information to the screen, assign values to variable, get information entered from the keyboard, and perform a counted loop

2.1 The software development process

The process of creating a program is often broken down into stages according to the information that is produced in each phase. In a nutshell, here’s what you should do:

**Analyse the Problem**

Figure out exactly what the problem to be solved is. Try to understand as much as possible about it. Until you really know what the problem is, you cannot begin to solve it.

**Determine Specifications**

Describe exactly what your program will do. At this point, you should not worry about *how* your program will work, but rather about deciding exactly *what* it will accomplish. For simple programs this involves carefully describing what the inputs and outputs of the program will be and how they relate to each other.

**Create a design**

Formulate the overall structure of the program. This is where and *how* the programs get worked out. The main task is to design the algorithm(s) that will meet the specifications.

**Implement the design**

Translate the design into a computer language and put it into the computer. In this book, we will be implementing out algorithms as Python programs.

**Test/Debug the program**

Try out your program and see whether it works as expected. If there are any error (often called *bugs*), then you should go back and fix the. The process of locating and fixing errors is called *debugging* a program. During the debugging phase, your goal is to find errors, so you should try everything you can think of that might “break” the program. It’s good to keep in mind the old maxim: “Nothing is fool proof because fools are too ingenious.”

**Maintain the program**

Continue developing the program in response to the needs of your users. Most programs are never really finished; they keep evolving over the years of use.

2.2 Example program: Temperature converter

IPO – Input, Process, Output

Pseudocode – Precise English that describes what a program does. It is meant to communicate algorithms without all the extra mental overhead of getting the details right in any particular programming language.

2.3 Elements of Programs

Names.

We give names to modules (e.g. convert) and to the function within modules (e.g. main). Variables are used to give names to values (e.g., Celsius and Fahrenheit). Technically, all these names are called *indetifiers.*

Every identifier must begin with a letter or underscore (the “\_” character) which may be followed by any sequence of letters, digits, or underscores. This implies that a single identifier cannot contain any spaces and are case\_sensitive.

Example:

X

Celsius

Spam

Spam2

SpamAndEggs

Spam\_and\_Eggs

However, be wary of that some identifiers are part of Python itself. These names are called reserved words or keywords and cannot be used as ordinary identifiers.

Expressions.

The fragments of program code that produce or calculate new data values are *expressions*.

The simplest kind of expression is a *literal*. A literal is used to indicate a specific value

String is a sequence of printable characters. A string literal is indicated in Python by enclosing the characters in quotation marks (“ ”).

Evaluation is the process of turning an expression into an underlying data type. When you type an expression into a Python shell, the shell evaluates the expression and print out a textual representation of the result. Consider this small interaction:

>>> 32

32

>>> “Hello”

“Hello”

>>> ”32”

“32”

When an identifier appears as an expression, its value is retrieved to provide a result for the expression. Here is an interaction with the Python interpreter that illustrates the use of variable as expressions.

>>> x = 5

>>> x

5

>>> print(x)

5

>>> print (spam)

Traceback (most recent call last):

File “<stdin>”, line 1, in <module>

NameError: name “spam” is not defined

NameError means that there is no value with that name.

Concatenation is the result of “gluing” the strings together.

2.4 Output statements

Python has a precise set of rules for the syntax (form) and semantics (meaning) of each statement.

Computer scientists have developed sophisticated notations called *meta-languages* for describing programming languages.

print(<expr>, <expr>, …, <expr>)

This examples angle bracket notation (<>) in the template is used to indicate “slots” that are filled in by other fragments of Python code. The name inside the brackets indicates what is missing; expr stands for an expression. The ellipsis (….) denotes an indefinite series (of expression, in this case).

print( )

This example shows that it’s also legal to have a print without an expressions to print.

Print statements (semantics) displays information in textual form.

As an example, this sequence of print statements:

print(3 + 4)

print(3, 4, 3+4)

print( )

print(“The answer is”, 3 + 4)

Produces the out:

7

3 4 7

The answer is 7

A template for the print statement including the keyword parameter to specify the ending text looks like this:

print(<expr>, <expr>, …, <expr>, end=”\n”)

One common use of the end parameter in print statements is to allow multiple prints to build up a single line of output. For example:

print( “The answer is”, end = “ “)

print( 3 + 4)

produces the single line of output:

The answer is 7

Notice the output from the first print statement ends with a space (“ “) rather than an end–of–line character. The output from the second statement appears immediately following the space.

2.5 Assignment statements

Simple assignment.

The basic assignment statement has this form:

<variable> = <expr>

Here variable is an identifier and expr is an expression. The semantics of the assignment is that the expression on the right side is evaluated to produce a value, which is then associated with the variable named on the left side.

Here are some of the assignments we’ve already seen:

x = 3.9 \* x \* (1 – x)

fahrenheit = 9/5 \* celcius + 32

x = 5

A variable can be assigned many times. It always retains the value of the most recent assignment. Here is an interactive Python session that demonstrates the point:

>>> myVar = 0

>>> myVar

0

>>> myVar = 7

>>> myVar

7

>>> myVar = myVar + 1

>>> myVar

8

The last assignment statement shows how the current value of a variable can be used to update its value.

Assigning Input.

The purpose of an input statement is to get some information from the user of a program and store it into a variable.

<variable> = input(<prompt>)

<prompt> - A string expression that is used to prompt the user for input; the prompt is almost always a string literal (i.e., some text inside of quotation marks.)

input – it prints the prompt on the screen

Python then pauses and waits for the user to type some text and press the <Enter> key. Whatever the user types is then stored as a string. Consider the simple interaction:

>>> name = input(“Enter your name: “)

Enter your name: JJ Takhar

>>> name

“JJ Takhar”

When the user input is a number, we need a slightly more complicated form of input statement:

<variable> = eval(input(<input>))

The important thing to remember is that you need to eval the input when you want a number instead of some raw text (a string).

Consider the following interaction with the Python interpreter

>>> ans = eval(input (“Enter an expression: “))

Enter an expression: 3 + 4 \* 5

>>> print (ans)

23

>>>

Beware of eval functions as they can be exploited to retrieve information on the database.

Simultaneous Assignment.

There is an alternative form of the assignment statement that allows us to calculate several values all at the same time. It looks like this:

<var1>, <var2>, …, <varn> = <expr1>, <expr2>, …, <exprn>

This is called simultaneous assignment. Semantically, this tells Python to evaluate all the expression on the right-hand side and then assign these values to the corresponding variables named on the left-hand side. Here’s an example:

sum, diff = x+y, x-y

Here sum would get some of x and y, and diff would get the difference.

Example:

Suppose you have two variables x and y, and you want to swap the values. That is, you want the value currently stored in x to be in y and the value that is currently in y to be stored in x. At first, you might think this could be done with two simple assignments:

X = y

Y = x

This doesn’t work. We can trace the execution of these statements step by step to see why.

Suppose x and y start with the value 2 ad 4. Let’s examine the logic of the program to see how the variables change. The following sequence uses comments to describe what happens to the variables as these two statements are executed:

#variables x y

#initial values 2 4

X = y

# now 4 4

Y = x

#final 4 4

See how the first statement clobbers the original value of x by assigning to it the value of y? When we assign x to y in the second step, we just end up with two copies of the original y value.

One way to make the swap work is to introduce an additional variable that temporarily remembers the original value of x.

Temp = x

X = y

Y= temp

Let’s walk through this sequence to see how it works.

#variables x y temp

#initial values 2 4 no value yet

Temp = x

# 2 4 2

X = y

# 4 4 2

Y = temp

#final 4 2 2

As you can see from the final values of x and y, the swap was successful in this case.

Simultaneous assignment statement offers and elegant alternative. Here is a simpler Python equivalent:

X, y = y, x

Simultaneous assignment can also be used to get multiple numbers from the user in a single input. Consider this program for averaging exam scores.

See Anaconda platform

Definite loop

A definite loop is a loop that will execute a definite number of times. That is, at the point in the program when the loop begins, Python knows how many times to go around (or iterate) the body of the loop. For example, the chaos program in Chapter 1 used a loop that always executed exactly ten times.

for i in range(10):

x = 3.9 \* x \* (1 - x)

print(x)

This particular loop is a counted loop, and it is built using a Python for statement. Before considering this example in detail, lets take a look at what for loops are all about.

A Python for loop has this general form:

For < var > in < sequence >:

< body >

The body of the loop can be any sequence of Python statements. The extent of the body is indicated by its indentation under the loop heading (the for <var> in <sequence>: part.

The variable after the keyword for is called the loop index. It takes on each successive value in the sequence, and the statements in the body are executed once for each value. Often the sequence portion consists of a list of values. Lists are very important concept in Python, and you will learn more about it them in upcoming chapters. For now, it’s enough to know that you can create a simple list by placing a sequence of expressions in square brackets. Some interactive example to help to illustrate the point:

>>> for i in [0, 1, 2, 3]:

Print (i)

0

1

2

3

>>> for odd in [1, 3, 5, 7, 9]:

Print(odd \* odd)

1

9

25

49

81

The body of the loop is executed using each successive value in the list. The length of the loop is executed each successive value in the list.

The expression range(10) produces the sequence of numbers 0 through 9. The loop using range(10) is equivalent to one using a list of those numbers.

for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:

for <variable> in range (<expr>):

The value of the expression determines how many times the loop executes. The name of the index variable doesn’t really matters much; programmers of use I or j as the loop index variable for counted loops.

The interesting and useful thing about loops is the way that they alter the “flow of control” in a program. Usually we think of computers as executing a series of instructions in strict sequence. Introducing a loop causes Python to go back do some statements over and over again. Statements like the for loop are called control structures because they control the execution of other parts of the program.

Example program: Future value

**Chapter summary**

This chapter has covered a lot of ground laying out both the process that is used to develop programs and the details of Python that are necessary to implement simple programs.

Here is a quick summary of some of the key points:

* Writing programs requires a systematic approach to problem solving and involves the following steps:

1. Problem analysis: Studying the problem to be solved
2. Problem specification: Deciding exactly what the program will do.
3. Design: Writing an algorithm in pseudocode
4. Implementation: Translating the design into a programming language.
5. Testing/Debugging: Finding and fixing errors in the program
6. Maintenance: Keeping the program up to date with evolving needs.

* Many simple programs follow the input, process, output (IPO) pattern.
* Programs are composed of statements that are built from identifiers and expressions.
* Identifiers are names; they begin with an underscore or letter which can be followed by a combination of letter, digit, or underscore characters. Identifiers in Python are case – sensitive.
* Expressions are the fragments of a program that produced data. An expression can be composed of the following components:
  + **Literals** – A literal is a representation of a specific value. For example, 3 is a literal representing the number three.
  + **Variables** – A variable is an identifier that stores a value.
  + **Operators** – Operators are used to combine expressions into more complex expressions. For example, in x + 3 \* y the operators + and \* are used.
* The Python operators for numbers include the usual arithmetic operations of addition (+), subtraction (-), multiplication (\*), division (/), and exponentiation (\*\*).
* The Python output statement print displays the values of a series of expressions to the screen.
* In Python, assignment of a value to a variable is indicated using the equal sign (=). Using assignment, programs can get input from the keyboard. Python also allows simultaneous assignment, which is useful for getting multiple input values with a single prompt.
* The eval function can be used to evaluate user input, but it is a security risk and should not be used with input from unknown or untrusted sources.
* Definite loops are loops that execute a known number of times. The Python for statement is a definite loop that iterates through a sequence of values. A Python list is often used in a for loop to provide a sequence of values for the loop.
* One important use of a for statement is in implementing a counted loop, which is a loop designed specifically for the purpose of a repeating some portion of the program a specific number of times. A counted loop in Python is created by using the built-in range function to produce a suitability sized sequence of numbers.

**Exercises**

Review questions

True/False

1. The best way to write a program is to immediately type in some code then debug it until it works. **FALSE**
   1. The best way to program is to approach it in a systematic way which involves:
      1. Program analysis: Studying the problem to be solved.
      2. Program specifications: Deciding exactly what the program will do.
      3. Design: Write an algorithm in pseudocode
      4. Implementation: Translating the design into a programming language
      5. Testing/Debugging: Finding and fixing errors in the program.
      6. Maintenance: Keeping the program up to date with evolving needs.
2. An algorithm can be written without using a programming language. **TRUE**
   1. That is correct – pseudocode is an example of this
3. Programs no longer require modification after they are written and de-bugged. **FALSE**
   1. Under programming in a systematic way - in regards to maintenance. As a programmer we will continue developing the program in response to the needs of users. Most programs are never really finished; they keep evolving over years of use.
4. Python identifiers must start with a letter or underscore. **TRUE**
   1. Python identifiers are names that are given to program entities.
      1. Every identifier must must begin with a letter or underscore (the “\_”) which may be followed by an sequence of letters, digits, or underscores.
5. Keywords make good variable names. **FALSE**
   1. Keywords are a part of identifiers inbuilt into Python themselves and cannot be used a ordinary identifiers.
6. Expressions are built from literal, variable, and operators. **TRUE**
   * 1. Expressions - Apart of a program that produces data.
     2. Literal – A notation for writing a specific value in a programming language. For example, 3 is an int literal and “Hello” is a string literal.
     3. Variable – An identifier that labels a value for future reference. The value of a variable can be changed through assignment.
     4. Operators – A function for combining expressions into more complex expressions.
7. In Python, x = x + 1 is a legal statement. **True**
   1. Sometimes its helpful to think of a variable as a sort of named storage location in computer memory, a box that we can put a value in. When the variable changes, the old value is erased and a new one written in. Figure 2.1 shows how we might picture the effect of x = x + 1using this model. This is exactly the way assignment works in some computer languages.
8. Python does not allow the input of multiple values with a single statement. **False**
   1. Drawn from p. 41 on simultaneous assignment and p. 39 on input: Since Python allows simultaneous assignment, the built-in input function can be used to prompt the user for multiple values which are simultaneously assigned to variables.
9. A counted loop is designed to iteral a specific number of times. **True**
   1. A definite/counted loop is a loop that executes a known number of times. The Python for statement is a definite loop that iterates through a sequence of values.
10. In a flowchart, diamonds are used to show statements sequences, and rectangles are used for decision points. **False**
    1. Diamond shaped box in the flow chart represents a decision in the program.
       1. Flow chart
          1. A flowchart is a diagram that uses boxes to represent different parts of a program
       2. Statement sequences
          1. In computer programming, a statement is a syntactic unit of an imperative programming language that expresses some action to be carried out. A program written in such a language is formed by a sequence of one or more statements
       3. Decision points
          1. Anticipation of conditions occurring while execution of the program and specifying actions taken according to conditions.

**10/10**

Multiple choice

1. Which of the following is not a step in the software development process? **C. – Fee setting**
   1. The software development process is as follows:
      1. Problem analysis – Studying the problem to be solved
      2. Program specification – Deciding exactly what the program will do
      3. Design – Writing an algorithm in pseudocode
      4. Implementation – Translating the design into a programming language
      5. Testing/Debugging – Finding and fixing errors in the program
      6. Maintenance – Keeping the program up to date with evolving needs
   2. What is the correct formula for converting Celsius to Fahrenheit? **A F = 9/5(C) + 32**
      1. The rest of the formula are mathematically incorrect
2. The process for describing exactly what a computer program will do to solve a problem is called **C. Programming WRONG – D. Specifications**
   1. Design – The process of developing a system that can solve some problem. Also, the product of that process
   2. Implementation – Translating the design into a programming language
   3. Programming – The process of creating a computer program to solve some problem
   4. Specifications – Describes a product intended capabilities, appearance, and interactions with users in detail for software developer.

1. Which of the following is not a legal identifier? **C. 2spam**
   1. Identifiers are names; they begin with an underscore or letter which can be followed by a combination of a letter, digit, or underscore character. Identifiers in Python are case-sensitive.
2. Which of the following are *not* used in expressions? **B. Statements**
   1. Expressions are the fragments of a program that produce data. An expression can be composed of the follow components:
      1. Literals – A literal is a representation of a specific value. For example, 3 is a literal representing the number three.
      2. Variables – A variable is an identifier that stores a value
      3. Operators – Operators are used to combine expressions into more complex expressions. For example, in x + 3 \* y the operators + and \* are used.
3. Fragments of code that produce or calculate new data values are called:

**B. Expressions**

* 1. Identifiers – Identifiers are names; they begin with an underscore or letter which can be followed by a combination of a letter, digit, or underscore character. Identifiers in Python are case-sensitive.
  2. Expressions – Expressions are the fragments of a program that produce data. An expression can be composed of the follow components:
     1. Literals – A literal is a representation of a specific value. For example, 3 is a literal representing the number three.
     2. Variables – A variable is an identifier that stores a value
     3. Operators – Operators are used to combine expressions into more complex expressions. For example, in x + 3 \* y the operators + and \* are used.
  3. Productive clauses – ???
  4. Assignment statements – The process of giving a value to a variable

1. Which of the following is *not* part of the IPO pattern?

**B. Program**

* 1. Many simple programs follow the input, process, output (IPO) pattern.

1. The template for <variable> in range (<expr>) describes **D. counted loop**
   1. General for loop – Used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
   2. An assignment statement – The process of giving a value to a statement
   3. A flow chart – A flow chart is a diagram that uses boxed to represent different parts of a program and arrows between the boxes to show the sequence of events when the program is running.
   4. A counted loop - A definite/counted loop is a loop that executes a known number of times. The Python for statement is a definite loop that iterates through a sequence of values.
2. Which of the following is the most accurate model of assignment in Python? **B. Assignment statements – A. Sticky note**
   1. Sticky note -
   2. An assignment statement – The process of giving a value to a statement
   3. Simultaneous – There is an alternative form of the assignment statement that allows us to calculate several values all at the same time.
   4. Plastic scale -
3. In Python, getting user input is done with a special expression called. D. input
   1. For - Used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
   2. Read -
   3. Simultaneous assignment – There is an alternative form of the assignment statement that allows us to calculate several values all at the same time.
   4. input – The purpose of the input statement is to get some information from the user of a program and store it in a variable.

**6/10**

Discussion

1. List and describe in your own words the six steps in the software development process.
   1. Problem analysis – Studying the problem to solve
   2. Program specifications – Deciding exactly what the program will do.
   3. Design – Writing an algorithm in pseudocode
   4. Implementation – Translating the design into a programming language
   5. Testing/debugging – Finding and fixing errors in the program
   6. Maintenance – Keeping the program up to date with evolving needs
2. Write out the chaos.py program (Section 1.6) and identify the parts of the program as follows:
   1. Circle each identifier
      1. Identifiers are names; they begin with an underscore or letter which can be followed by a combination of letter, digit, or underscore characters. Identifier in Python are case sensitive
   2. Underline each expression
      1. Expressions are the fragments of a program that produce data. An expression can be composed of the following components:
         1. Literals – is a representation of a specific value. For example, 3 is a literal representing the number 3.
         2. Variables – a variable is an identifier that stores a value
         3. Operators – Are used to combine expressions into more complex expressions. For example, in x + 3 \* y the operators + and \* are used.
   3. Put a comment at the end of each line indicating the type of statement on that line (Output, Assignment, Input, loop, etc.)
      1. Output – Output statements are like print statements that give an immediate effect
      2. Assignment – The semantics of an assignment that that the expressions usually on the right side is evaluated to produce a value, which is then associated with the variable named on the left side.
      3. Input – Input statements are there to get some information from the user of a program and store it into a variable.
      4. Loop – Loops are created to execute a sequence of statements multiple times in succession.
      5. SEE CODE IN PYTHON PROGRAMMING CODE
3. Explain the relationships among the concepts: definite loop, for loop, and counted loop
   1. Definite loops – Executes a loop a definite number of times, that is at the point in the program when the loops begin.
   2. Counted loops – Is built using a Python for statement which executes a loop an exact period time.
   3. For loops – A Python for loop has the general form
      1. For <var> in <sequence>
      2. <body>
      3. The body of the loop can be any sequence of Python statements. The extent of the body is indicated by its indentation under the loop heading (the for <var> in <sequence>: part).
   4. Therefore, the relationship between all the above is the Definite loop is the high level version of what a loop is where a counted loop gives the exact amount of times a loop is executed. The for loop of the relationship is the exact code the is executed under the sequence of the Python code.
4. Show the output of the following codes:
   1. SEE CODE IN PYTHONG PROGRAMMING CODE
5. Why is it a good idea to first write out an algorithm in pseudocode rather than jumping immediately to Python code?
   1. Pseudocode is the precise notation for writing algorithms using natural language, instead of a computer language.
      1. Pseudocode is written to describe what a program does and communicate algorithms without all the mental extra overhead of getting the details right in any particular programming language.
6. The Python print function supports other keyword parameters besides end. One of these other keyword parameters is sep. What do you think the sep parameter does? Hint: sep is short for separator. Test your idea either by trying it interactively or by consulting Python documentation.
   1. Sep function in programming print functions separates the output into separate columns.
7. What do you think will happen when the following code is executed?
   1. The code will output “start” and then not print hello as the loop will note execute. It will then print “end”.

Programming exercise

1. A user-friendly program should print an introduction that tells the user what the program does. Modify the convert.py program (section 2.2 to print an introduction.)
   1. LOOK AT PYTHON CODE
2. On many systems with Python, it is possible to run a program by simply clicking (or double clicking) on the icon of the program file. If you are able to run a convert.py program this way, you may discover another usability issue. The program starts running in a new window, but as soon as the program has finished, the window disappears so that you cannot read the results. Add an input statement at the end of the program so that it pauses to give the user a change the read the results. Something like this should work:
   1. Input(“Press the <Enter> key to quit.”) on the last line of code before defining the code again \*\* LOOK AT GITHUB FOR ANSWERS
3. Modify the avg2.py program (section 2.5.3) to find the average of three exam scores.
   1. LOOK AT PYTHON CODE
4. Modify the convery.py program (Section 2.2) with a loop so that it executes 5 times before quitting. Each time through the loop, the program should get another temperature from the user and print the converted value.
   1. LOOK AT PYTHON CODE
5. Modify the convert.py program (Section 2.2) so that it computes and prints a table of Celsius temperatures and the Fahrenheit equivalents every 10 degrees from 0c to 100c.
   1. LOOK AT PYTHON CODE – **NEEDED HELP**
6. Modify the futval.py program (Section 2.7) so that the number of years for the investment is also a user input. Make sure to change the final message to reflect the correct number of years.
   1. LOOK AT PYTHON CODE
7. Suppose you have an investment plan where you invest a certain fixed amount every year. Modify futval.py to compute the total accumulation of your investment. The inputs to the program will be the amount to invest each year, the interest rate, and the number of years for the investment.
   1. LOOK AT PYTHON CODE – **NEEDED HELP**
8. As an alternative to APR, the interest accrued on an account is often described in terms of a nominal rate and the number of compounding periods. For example, if the interest rate is 3% and the interest is compounded quarterly, the account actually earns ¾% interest every 3 months.
   1. LOOK AT PYTHON CODE – **NEEDED HELP**
9. Write a program that converts temperatures from Fahrenheit to Celsius.
10. Write a program that converts distances measured in kilometres to mils. One kilometre is approximately 0.62 mils.
11. Write a program to perform a unit conversion of your own choosing. Make sure that the program prints an introduction that explains what it does.
12. Write an interactive Python calculator program. The program should allow the user to type a mathematical expression, and then print the value of the expression. Include a loop so that the user can perform many calculations (say, up to 100). Note: to quit early, the user can make the program crash by typing a bad expression or simply closing the window that the calculator program is running in. You’ll learn better ways of terminating interactive programs in later chapters.

**Chapter 3. Computing with numbers**

Objectives

* To understand the concept of data types
* To be familiar with the basic numeric data types in Python
* To understand the fundamental principles of how numbers are represented on a computer
* To be able to use the Python math library
* To understand accumulator program patterns
* To be able to read and write programs that process numerical data

3.1 Numeric Data Types

Data is the information that is stored and manipulated by computer programs. Different kinds of data will be stored and manipulated in different ways.

General rule of thumb:

If you don’t need fractional values, use an int.

The data types of an object determined what values it can have and what operations can be performed on it.

* Integer (int)
  + integers are whole numbers that can be positive or negative and are distinguished by not containing a decimal point.
* Floating point (float)
  + Numbers that can have fractional parts and decimal points.

You may be wondering why there are two different data types for numbers.

One reason has to do with program style. Value that represent counts can’t be fractional; we can’t have 3 ½ quarters, for example. Using an Int value tells the reader of a program that the value can’t be a fraction.

The second reason has to do with the efficiency of various operations. The underlying algorithms that perform computer arithmetic are simpler, and can therefore be faster,for ints that the more general algorithms required for float values.

Another difference between ints and floats is that the float type can only represent approximations to real numbers. As we will see, there is a limit to the precision, or accuracy, of the stored values. Since float values are not exact, while ints always are, your general rule of thumb should be: If you don’t need fractional values, use an int.

The computer hardware performs a floating-point addition, whereas with ints the computer performs an integer addition.

Example on Notebook\*

In the case of division, however, things get a bit more interesting. The usual symbol (/) is used for “regular” division and get double slash (//) is used to indicate integer division. The best way to get a handle on the difference between these two is to try them out.

Example in Notebook\*

Remember floating point numbers are always approximations.

To get a division that returns an integer result, you can use the integer division operation //. Integer division always produces an integer. Also, data types are always dependent on the operands.

3.2 Type conversions and rounding

In a mixed-type expression, Python will automatically convert ints to floats and perform floating -point operation to produce a float result.

Explicit type – Performing a type conversion ourselves

Example in Notebook\*

Built in functions

>>> int(4.5)

4

>>>float(4)

4.0

>>> round(3.14)

3

Note that floats are approximations; what we really get is a value that’s very close that what we requested.

The type conversion functions int and float can also be used to convert strings of digits into numbers.

>>> int(”32”)

32

>>>float(“32)

32.0

>>>float(“9.8”)

9.8

\*Example in Notebook\*

Using int instead of eval in the input statements ensures that the user may only enter valid whole numbers. Any illegal (non-int) inputs will cause the program to crash with an error message, thus avoiding the risk of a code injection attack. A side benefit is that this version of the programs emphasizes that the inputs should be whole numbers.

The only downside to using numeric type conversions in place of eval is that it does not accommodate simultaneous input (getting multiple values in a single input), as the following example illustrates.

>>> # simultaneous input using eval

>>> x,y = eval(input (“Enter (x,y): “))

Enter (x,y): 3,4

>>>x

3

>>>y

4

>>> # does not work with float

>>> x,y = flat(input(“Enter (x,y): “))

Enter (x,y): 3,4

Traceback (most recent call last):

File “<stdin>”, line 1, in <module>

ValueError: could not convert string to flat: “3,4”

As a matter of good practice, you should use appropriate type conversion functions in place of eval wherever possible.

3.3 Using the Math Library

A library contains useful definitions. Our next program illustrates the use of this this library to compute the roots of quadratic functions.

Let’s write a program that can find the solution to a quadratic equation. The input to the program will be the values of the coefficient a, b, and c. The outputs are the two values given by the quadratic formula. Here’s a program that does the job:

\*Examples in Notebook\*

3.4 Accumulating Results: Factorials\*\*

Suppose you have a root beer sampler pack containing dix different kinds of root beer. Drinking the various flavours in different orders might affect how good they taste. If you wanted to try out every possible ordering how many different orders would there be? It turns out the answer is a surprisingly large number, 720. Do you know where this number comes from? The value 720 is the factorial of 6.

Input number to take factorial of, n

Compute factorial of nm fact

Output fact

In computing the factorial of 6, we first multiply 6(5) = 30. Then we take that result and do another multiplication: 30(4) = 120. This result is multiplied by 3: 120(3) = 360. Finally, this result is multiplied by 2: 360 (20) = 720. According to the definition, we then multiply this result by 1, but that won’t change the final value of 720.

Accumulator – We build up, or accumulate, a final value piece by piece.

To accomplish this in a program, we will an accumulator variable and a loop structure.

The general pattern looks like this:

Initialize the accumulator variable

Loop until final result is reached

Update the value of accumulator variable

Range (start, n) produces a sequence that starts with the value start and continues up to, but does not include n.

The differing version being range (start, n, step) is like the two-parameter version, except that it uses step and the increment between numbers.

See notebook for examples.

3.5 Limitations of Computer Arithmetic

Each bit can represent 2^n different values.

Typically, PC’s use 32 – 64 bit. For a 32-bit CPU, means that there are 2^32 values.

A sequence of bits can be used to represent more possibilities. With two bits, we can represent four things:

|  |  |
| --- | --- |
| Bit 2 | Bit 1 |
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

Three bits allow us to represent eight different values by adding a 0 or 1 to each of the four two-bit patterns:

|  |  |  |
| --- | --- | --- |
| Bit 3 | Bit 2 | Bit 1 |
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

Each extra bit doubles the number of distinct patterns. Each extra bit doubles the number of distinct patterns. In general, n bits can represent 2^n different values.

3.6 Chapter Summary

This chapter has filled in some important details concerning programs that do numerical computations. Here is a quick summary of some key concepts.

* The way a computer represents a particular kind of information is call data type. The data type of an object determines what values it can have and what operations it supports.
* Python has several different data types for representing numberic values, including int and float.
* Whole numbers are generally represented using the int data type, and fractional values are represented floats. All of the Python numeric data types support standard, built-mathmatical operations: addition (+), subtraction (-), multiplication (\*), division (/), integer division (//), remainder (%), exponentiation (\*\*), and absolute values (abs(X)).
* Python automatically converts numbers from one data type to another in certain situations. For example, in a mixed-type expression involving ints and floats, Python first converts the ints into floats and then uses float arithmetic.
* Programs may also explicitly convert on data type into another using the functions float(), int(), and round(). Type conversion functions should generally be used in place of eval for handling numeric user inputs.
* Additional mathematical functions are defined in the math library. To use these functions, a program must first import the library.
* Numerical results are often calculated by computing the sum or product of a sequence of values. The loop accumulator programming pattern is useful for this sort of calculation.
* Both ints and floats are represented on the underlying computer using a fixed-length sequence of bits. This imposes certain limits on these representations. Hardware ints must be in the range -23^31…. \*2^31 – 1) on a 32-bit machine. Floats have a finite amount of precision and cannot represent most numbers exactly.
* Python’s ints data type may be used to store whole numbers of arbitrary size. Int values are automatically converted to longer presentations when they become too large for the underlying hardware int. Calculations involving these long ints are less efficient that those that use only small ints.

Exercises

Review questions

True/false

1. Information that is stored and manipulated by computers is called data

True – The information that a computer program manipulates.

1. Since floating-point numbers are extremely accurate, they should generally be used in ints

False – The difference between ints and floats is that the float type can only represent approximations to real number. Since float values are not exact, while ints always are, your general rule of thumb should be: if you don’t need fractional values, us an int.

1. Operations like addition and subtraction are defined in the math library

False – operators like addition, subtraction, multiplication, float division, exponentiation, absolute value, integer division and remainder in the Python built-in numeric operations

1. The number of possible arrangements of n items is equal to n!.

True - 0 That’s correct as it will be n factorial

1. The sqrt function computes the sqrt of a number

True – sqrt(x) = The square root of X

False – The Sqrt function from the **Math library** computes the square root of the number.

1. The float data type is identical to the mathematical concept of a real number

True – however due to the limitations of Python it won’t give the full approximation and will make it 2+32

1. Computers represent numbers using base-2 (binary) representations.

Binary – The base-2 numbering system in which the only digits are 0 and 1.

True – All data in a computer stores in Binary, or have two representations, and this includes numbers.

1. A hardware float can represent a larger range of value than a hardware int

True as floats will do an e^10. This means that the trade off is that int can be stored accuratly, while floats can’t.

1. Type conversion functions such as float are safe alternative to eval for getting a number as user input.

True - Rule of thumb is to use an integer rather than a float. Integers however are useful as it prevents code injection.

1. In Python, 4+5 produces that same result type as 4.0 + 5.0

True but the float type will give a .0

8/10 – 80%

Multiple choice

1. Which of the following is not a built-in Python data type?

c). Rational – Is not a native data type in python. All mathematical rational numbers are represented using wither intger or floating point numbers.

1. Which of the following is not a built-in operation?

c). Abs

d). +, % are nuilt in opertaors in the python language, and abs is a built-in function. However, sqrt is included from the math library.

1. In order to use functions in the math library, a program must include

d). Import statement – To use the functions from the math library, we first import it using the import math statements, and then use the functions e.g. math.sqrt (5)

Import statements are a statement that makes an external library module available for use within a program.

1. The value of 4! Is

b). 24 = 4 x 3 x 2 x 1

1. The most appropriate data type for storing the value of pi is

b). float as it has multiple decimal points – Irrational is not a data type in Python. Pi is an irrational number, and hence it’s value can be stored to appropriate number of decimal places using float.

1. The number of distinct values that can be represented using 5 bits is:

Formula: 2\*\*n

c). 32 – Given 5 bits, since each bit can take value 0 or 1 i.e. 2 values per bit. Since the bits vary independently, the total number of possible combinations of bits is 2 \* 2 \* 2 \* 2 \* 2 = 2^5 = 32

1. In a mixed-type expression involving ints and floats, Python will convert

d). Ints to floats and Python will try to preserve the accuracy of numbers

1. Which of the following is not a Python type conversion function?

d). abs

1. The pattern used to compute factorials is

b). input, process, output – A common programming pattern. The program prompts for input, processes it, and outputs a response.

a). Accumulator – A common programming pattern in which a final answer is built a piece at a time in a loop.

1. In modern Python, an int value that grows larger than the underlying hardware int

a). Causes an overflow

b). Converts to float – Precision is maintained by Python in this case by converting the underlying integer into a float, this is done automatically so the programmer does not even need to think about it.

7/10 – 70%

Discussion

1/.

5/6 = 83%

2/.

4/5 = 80%

3/.

5/5 = 100%

4/.

4/4 = 100%

5/.

0/1 = 0%

6/.

5/5 = 100%

Programming exercises

1/. 3/5 = 75%

2/. 1/5 = 20%

3/. 4/5 = 80%

4/. 3/5 = 60%

5/. 3.5/5 = 70%

6/. 3/5 = 60%

7/. 5/5 = 100%

8/. 4.5/5 = 90%

9/. 5/5 = 100%

10/. 4.5/5 = 90%

11/. 2/5 = 40%

12/. 3/5 = 60% - made it a square function not cubed

13/. 0/5 = 0%

14/. 5/5 = 100%

13/. – 14/. – important loop to know

for i in range(1, n + 1):

num = float(input("Enter number: "))

total = (total + num)

answer = total/n

15/. 0/5 – need to understand how to derive your own formulas to solve problems

**Chapter 4. Objects and Graphics**

Objectives

* To understand the concept of objects and how they can be used to simplify programming.
* To become familiar with the various objects available in the graphics library.
* To be able to create objects in programs and call appropriate methods to perform graphical computations
* To understand the fundamental concepts of computer graphics, especially the role of coordinate systems and coordinate transformations.
* To understand how to work with both mouse- and text-based input in a graphical programming concept
* To be able to write simple interactive graphics programs using the graphics library.

4.1 Overview

Graphical user interface (GUI)

Provides visual elements like windows, icons (representative pictures), buttons and menus.

4.2 The Object of objects

The basic idea of object-orientated development to view a complex system as the interaction of simpler objects. The word objects is being used here in a specific technical essence.

You can think of an OO object as a sort of active data type that combines both data and operations. To put simply objects, know stuff (they contact data), and they can do stud (they have operations). Objects interact by sending each other messages.

A message is simply a request for an object to perform one of its operations.

4.3 Simple graphics programming

**IDE**

Integrated Development Environment dedicated to software development.

Check Chapter 4.py for examples

4.4 Using Graphical Objects

Definitions

* Objects combine data with operations.
* A call to constructor is an expression that creates a brand-new object
* Methods are the set of messages that an object responds to. You can think of methods as function that live inside the object.
* Accessors allow us to access information from the instance variables of the object
* Mutators are methods that change the state of an object
* Aliasing is the situation where two variables refer to the same object.

Often a constructor is used on the right side of an assignment statements, and the resulting object is immediately assigned to a variable on the left side that is then used to manipulate the object.

Example.

p = Point (50,60)

The constructor for the point class requires two parameters giving the x and y coordinates for the new point. These variables are stored as *instance* variables.

This creates an instance point having x value of 50 and a y value of 60. The resulting point is then assigned to the variable p.

A method is invoked using dot-notation.

<object>.<method-name>(<param1>,<param2>,…..)

An example of parameter less methods, consider these two expressions:

p.getX( )

p.getY( )

The getX and getY methods return x and y values of a point, respectively.

Other methods change the values of an object’s instance variables, hence changing the state of the object. All of the graphical objects have a move method.

Here is a specification:

move (dx, dy) : Moves the object dx units in the x direction and dy units in the y direction.

To move the point p to the right 1 unit, we could use this statement:

p.move (10,0)

This moves the x instance variable of p by adding 10 units.

Example. Drawing a circle.

Let examine a sequence of commands that does this:

Circ = Circle (Point(100, 100), 30)

Win = GraphWin( )

Circ.draw (win)

The first line creates a Circle with a centre located at the Point (100, 100) and a radius of 30. Notice that we used the Point constructor to create a location for the first parameter to the Circle constructor.

The second line creates a GraphWin.

The third line is a request for the Circle object circ to draw itself into the GraphWin object win.

The visible effect of this statement is a circle in the GraphWin centred at (100, 100) and having a radius of 30.

It is possible for two different variables to refer to exactly the same object; changes made to the object through one variable will also be visible to the other.

For example, we are trying to write a sequence of code that draws a smiley face. We want to create two eyes that are 20 units apart. Here is a sequence of code intended to draw eyes.

## Incorrect way to create two circles

leftEye = Circle(Point (80, 50) , 5)

leftEye.setFill(“yellow”)

leftEye.setOutline(“red”)

rightEye = leftEye

rightEye.move(20, 0)

The basic idea is to create the left eye and the copy that into a right eye, which is then moved over 20 units.

This doesn’t work. The problem here is that online one Circle object is created. This assignment

rightEye = leftEye simply makes the rightEye refer the very same circle as leftEye.

One solution to this problem would be to create a separate circle for each eye:

## A correct way to create two circles.

leftEye = Circle(Point (80, 50) , 5)

leftEye.setFill(“yellow”)

leftEye.setOutline(“red”)

rightEye = Circle(Point (80, 50) , 5)

rightEye.setFill(“yellow”)

rightEye.setOutline(“red”)

Above is cumbersome.

## Correct way to create two circles, using clone.

leftEye = Circle(Point (80, 50) , 5)

leftEye.setFill(“yellow”)

leftEye.setOutline(“red”)

rightEye = leftEye.clone() # tight Eye is an exact copy of the left

rightEye.move (20, 0)

4.5 Graphing Future Value (GO OVER THIS AGAIN SLOWLY)

Let’s use a concrete example for illustration.

Here is a rough design for the program:

Print an introduction

Get value of principal and apr from user

Create a GraphWin

Draw a scale labels on left side of window

Draw bar at position 0 with height corresponding to principal

For successive years 1 through 10

Calculate principal = principal \* (1 + apr)

Draw a bar for this year having a height corresponding to principal

Wait for user to press Enter

The pause created by the last step is necessary to keep the graphics window displayed so that we can interpret the results. Without such a pause, the program would end, and the GraphWin would vanish with it.

Now we need to flesh out the size of the graph.

Create a320 x 240 GraphWin titled “Investment growth chart”

This is translated to Python in the following code:

Win = GraphWin (“Investment Growth Chart”, 320, 240)

When creating Text, we specify the anchor point (the point the text is centred on) and the string to use as the label.

Draw label “0.0k” at (20, 230)

Draw label “2.5k” at (20, 180)

Draw label “5.0k” at (20, 130)

Draw label “7.5k” at (20, 80)

Draw label “10.0k” at (20, 30)

Now the size of the bars.

We can fill in the details for drawings the first bar into our algorithm

Draw a rectangle from (40, 230) to (65, 230 – principal \* 0.02)

Lets figure out where the lower left corner of each bar is going to be located.

For year running from a value of 1 up through 10:

Calculate principal = principal \* (1 + apr)

Calculate x11 = 25 \* year + 40

Calculate height = principal \* 0.02

Draw a rectangle from (x11, 230) to (x11 + 25, 230 – height)

The variable x11 stands for x lower left – the x value of the lower-left corner of the bar.

Putting this all together produces the detailed algorithm shown below:

Print an introduction

Get value of principal and apr from user

Create a 320 x 240 GraphWin titled “Investment Growth Chart”

Draw label “0.0k” at (20, 230)

Draw label “2.5k” at (20, 180)

Draw label “5.0k” at (20, 130)

Draw label “7.5k” at (20, 80)

Draw label “10.0k” at (20, 30)

Draw a rectangle from (40, 230) to (65, 230 – principal \* 0.02)

For year running from a value of 1 up through 10:

Calculate principal = principal \* (1 + apr)

Calculate x11 = 25 \* year + 40

Calculate height = principal \* 0.02

Draw a rectangle from (x11, 230) to (x11 + 25, 230 – height)

Wait for user to press Enter

The translation is straightforward using objects from the graphics library.

# Futval\_graph.py

from graphics import \*

LOOK AT EXAMPLE IN ANACONDA

All graphical objects support methods for chaning colour. I have set the background colour of the window to white:

Win.setBackground (“white”)

The following link asks the bar to colour its interior green

Bar.setFill (“green”)

You can also change the colour of a shapes outline using the setOutline method. In this case, I have chosen to leave the outline the default black so that the bars stand out from each other. To enhance this effect, this code makes the outline wider (two pixels instead of the default one):

Bar.setWidth (2)

You might also have noted the economy of notation in drawing the labels. Since we don’t ever change the labels, assigned them to a variable is unnecessary. We can just create a Text object, tell it do draw itself, and be done with it.

Here is an example:

Text(Point(20,230, ‘ 0.0k’).draw (win)

Finally, take a close look at the use of the year variable in the loop:

For year in range (1,11):

The expression range (1,11) produces a sequence of int 1-10. The loop index variable year marches through this sequence on successive iteration of the loop. So the first time through year 1, then 2, then 3, etc., up to 10. The value of year is then used to compute the proper position of the lower-left corner of each bar.

X11 = year \* 25 + 40

4.6 Choosing Coordinates

The method requires four parameters specifying the coordinates of the lower left and upper right corners, respectively. You can then use this coordinate system to place graphical objects in the window.

Example in Notebook

4.7 Interactive Graphics

Graphical interfaces can be used for input as well as output. In a GUI environment, users typically interact with their application by clicking on button, choosing items from menus, and typing information into on-screen text boxes.

These applications use a technique call event-driven programming.

An event is an object that encapsulates data about what just happened.

Example

* A click on a button might product an event button
* This event would be passed to the button handling code
* Perform the appropriate action corresponding to that button.

4.7,1 Getting mouse clicks

When getMouse is invoked on a GraphWin, the program pauses and waits for the user to click the mouse somewhere in the graphics window.

The spot where the user clicks is returned to the program as a Point.

Check the code in Notebook.

4.7,2 Handling Textual Input

The GraphWin object provides a getKey() method that works very much like the getMouse method. Here’s an extension of the simple clicking program that allow the user to label positions in a window by typing a single keypress after each mouse click.

Code found in notebook.

Notice what happens in the loop body. First it waits for a mouse click and the resulting Point is saved as the variable p. Then the program waits for the user to type a key on the keyboard. The key that is pressed is returned as a string and saved as the variable key. For example, if the users presses g on the keyboard, then key will be the string “g”. The point and string are then used to create a text object (called label) that is drawn in the window.

Here is a version of the temperature conversion program from Chapter 2 with a graphical user interface.

Code found in notebook.

When run, this produces a window with an entry box for typing in a Celsius temperature and a “button” for doing the conversion. The button is just for a show.

4.8 Graphics Module Reference

This section provides a complete reference to the objects and functions provided in graphics.

The set of objects and functions that are provided by a module is sometimes called and Applications Programming Interfaces, or API.

You will probably want to refer back to this section often when you are writing your own graphical programs.

One of the biggest hurdles in learning an API is familiarising yourself with the various data types that are used. As you read through the reference, pay close attention to the types of the parameters and return values of the various methods.

For example, when creating a CIRCLE, it’s essential that the first parameter you supple must be a Point object (for the center) and the second parameter must be a number (the radius). Using incorrect types will sometimes give an immediate error message, but other times problems may not crop up until later, say when the object is drawn.

4.8.1 GraphWin Objects

A GraphWin object represents a window on the screen where graphical images may be drawn. A program may define any number of GraphWin.

4.8.2 Graphics Objects

The modules provides the following classes of drawable objects: Point, Line, Circle, Oval, Rectangle, Polygon, and text. All objects are initially created unfilled with a black outline.

4.8.3 Entry Objects

Objects of the type Entry displayed as text entry boxes that can be edited by the user of the program. Entry objects support the generic graphics methods move (),draw (graphwin), undraw(), setFill(color), and clone().

4.8.4 Displaying images

The graphics module also provides minimal support for displaying and manipulating images in a GraphWin. Most platforms will support at least PPM and GIF images. Display is done with an image obect. Images support the generic methods move (dx, dy), draw (graphwin), undraw (), and clone ().

4.8.5 Generating Colours

Colours are indicated by strings. Most normal colours such as “red”, “purple”, “green”, “cyan”, etc, should be available. Many colours come in various shades, such as “red1”, “red2”, “red3”, “red4”, which are increasingly darker shades of red. For a full list, look up X11 colour names on the web.

The graphics module also provides a function for mixing your own colours numerically. The function colour\_rgb(red, green, blue) will return a string representing a colour that is a mixture of the intensities of red, gree, and clue specified. These should be ints in the range 0-255. Thus color\_rgb(255, 0, 0) is example a.Circle.setFill(colour\_rgb(130, 0, 130)).

4.8.6 Controlling Display updates(Advanced)

Under some circumstances, for example when using the graphics library inside some interactive shells, it may be necessary to force the window to update in order for changes to be seen. The update () function is provided to do this.

Update () Causes any pending operating to be carried out and the results displayed.

Chapter summary

This chapter introduced computer graphics and object0based programming. Here is a summary of some of the important concepts:

* An object is a computational entity that combines data and operations. Objects know stuff and can do stuff. An objects data is stored in instance variables, and its operations are called method.
* Every object is an instance of some class. It is the class that determines what methods an object will have. An instance is created by calling a constructor method.
* An objects attributes are accessed via dot notation. Generally computations with objects are performed by calling on an objects methods. Accessor methods return information about the instance variables of an object. Mutator methods change the value(s) of instance variables.
* The graphics module supplied with this book provides a number of classes that are useful for graphics programming. A GraphWin is an object that represents a window on the screen for displaying graphics. Various graphical objects such as Point, Line, Circle, Rectangle, Oval, Polygon< and Text may be drawn in a GraphWin. Users may interact with a GraphWin by clicking the mouse or typing into an Entry box.
* An important consideration in graphical programming is the choice of an appropriate coordinate system. The graphics library provides a way of automating certain coordinate transformations.
* The situation where two variables refer to the same object is called aliasing. Aliasing can sometimes cause unexpected results. Use of the clone method in the graphics library can help prevent these situations.

4.10 Exercises

Review questions

True/false

1. Using graphics.py allows graphics to be drawn in Python shell window.

True.

The graphics library makes it easy to experiment with graphics interactively and write a simple graphics program.

False.

The graphics are drawn in their own window, not the Python shell window.

2. Traditionally, the upper-left corner of a graphics window has coordinates (0,0)

True.

We define a point by supplying x and y coordinates (x,y). The x value represents the horizontal location of the point, and the y value represents the vertical.

Traditionally graphics programmers locate the point (0, 0) in the upper left corner of the window. Thus, x values increase from left to right, and y values increase from top to bottom. In the default 200 x 200 GraphWin, the lower-right corner has the coordinates (199, 1999).

Height – bottom-up y coordinate

3. A single point on a graphics screen is called a pixel.

True.

A pixel is a single dot on a graphical display.

4. A function that creates a new instance of a class is called an accessor.

True.

The accessor methods are a method that returns the value of one or more of an object’s instance variable(s), but does not modify the object.

Also, a class describes a set of related objects. The class mechanism in Python is used as a “factory” to produce objects.

False.

To create a new instance of a class, you use a constructor. An accessor is a function that returns a value stored in a object (like the getX Point accessor used in the chapter).

5. Instance variables are used to store data inside an object.

True.

An instance variable is a piece of data stored inside an object.

6. The statement myShape.move(10,20) moves myShape to the point (10, 20).

True.

Move(dx,dy) moves the object dx units in the x direction and dy units in the y direction. If the object is currently drawn, the image is adjusted to the new position.

Example: someObject.move (10, 15.5)

False.

The arguments of move are how far the object is moved, not where the object is moved to. So myShape.move (10,20) is used to move myShape 10 units to the right and 20 units up – it doesn’t move it to the point (10, 30) unless it’s already at point (0,0)

7. Aliasing occurs when two variables refer to the same object.

True.

Aliasing is the situation in which two or more variables refer to exactly the same object. If the object is mutable, then changes made through one variable will be seen by the others.

Var2 = var1

This statement DOES NOT create a copy of var1, but instead aliases var1 to var2. That means that bot var1 and var2 refer to exactly the same elements in memory and hence changing once will automatically change the other.

Sometimes aliasing can be what you want in a program, but more often (at least for beginners) it’s the cause of errors.

8. The copy method is provided to make a copy of a graphics object.

True.

The copy() method in Python returns a copy of the Set. We can copy a set to another set using the = operator, however copying a set using = operator means that when we change the new set the copied set will also be changed, if you do not want this behaviour then use the copy() method instead of = operator.

False.

The method to create a copy of graphics object is **clone**.

9. A graphics window always has the title “Graphics Window”

False.

Graph (title, width, height) constructs a new graphics window for drawing on the screen. The parameters are optional; the default title is “Graphics Window”, and the default size is 200 x 200 pixels.

10. The method in the graphics library used to get a mouse click is readMouse.

True.

getMouse ( ) pauses for the user to click a mouse in the window and returns where the mouse was clicked as a Point object.

Example: clickPoint = win.getMouse ( )

False.

The graphics library method is called getMouse. It causes the program to wait for a mouse click and then returns a Point object containing the coordinate of where the mouse clicked.

5/10 = 50%

Multiple choice

1. A Method that returns the value of an object’s instance variable is called a (n)

a. Mutator

A method that changes the state of an object (i.e., modifies one or more of the instance variables).

b. Function

A subprogram within a program. Functions take a parameter as input and can return values.

c. Constructor

A function that creates a new object. In a Python class, it is the \_ \_init\_\_ method.

d. Accessor

A Method that returns the value of one or more of an object’s instance variable(s), but does not modify the object.

2. A method that changes the state of an object is called (n)

a. Stator

-

b. Mutator

A method that changes the state of an object (i.e., modifies one or more of the instance variables).

c. Constructor

A function that creates a new object. In a Python class, it is the \_\_init\_\_ method.

d. Changor

-

3. What graphics class would be best for drawing a square

a. Square

Not a thing for the graphic.py module

b. Polygon

Polygon (point1, point2, point3, …) constructs a polygon with the given points as vertices. Also accepts a single parameter that is a list of the vertices.

Example:

Example:

c. Line

line (point 1, point2) constructs a line segment from point1 to point 2.

Example: aLine = Line (Point (1,3), Point (7,4))

d. Rectangle

Rectangle (point1, point2) constructs a rectangle having opposite corners at point1 and point2.

Example: aRectangle = Rectangle(Point (1, 3), Point (4,7))

4. What command would set the coordinates of win to go from (0,0) in the lower-left corner to (10,10) in the upper-right?

a. win.setcoords (Point(0,0), Point(10,10))

b. win.setcoords ((0,0), (10,10))

c. win.setcoords (0, 0, 10, 10)

d. win.setcoords (Point(10,10), Point(0,0))

5. What expression would create a line from (2,3) to (4,5)?

a. Line (2, 3, 4, 5)

b. Line ((2, 3), (4, 5))

c. Line (2, 4, 3, 5)

d. Line (Point(2, 3), Point (4, 5))

6. What command would be used to *draw the graphics object shapes* in the *graphics window win*?

a. win.draw(shape)

b. win.show(shape)

c. shape.draw( )

d. shape.draw (win)

The draw method is an object operation without the graphics object, not within the GraphWin object. So we’ll definitely have shape.draw. As for the argument, it determines which graphics window to draw the object in. In this case we want to draw it in win, so the statement would be shape.draw (win).

7. Which of the following computes the horizontal distance between points p1 and p2?

a. abs(p1 – p2)

b. p2.get( ) – p1.get( )

c. abs(p1.getY( ) – p2.getX( ))

d. abs(p1.getX( ) – p2.getX( )

In this case, we want the distance between the x coordinates of p1 and p2. So we’ll need to use the getX operations to retrieve the x coordinates and then calculate the distance.

Misread the HORIZONTAL distance.

8. What kind of object can be used to get text input in a graphics window?

a. Text

Constructs a text object that displays textstrings centred at anchorPoint.

b. Entry

Constructs an Entry having the given centre point and width. The width is specified in number of characters of text that can be displayed.

We use the Entry object to get text input from the graphics window. It creates a text box where the user can type in information.

c. Input

Prompts for input, processes it and outputs a response

d. Keyboard

9. A user interface organized around visual elements and user actions is called a (n)

a. GUI

A style of interaction with a computer application that involves heavy use of graphical components such as windows, menus and buttons.

b. Application

c. Windower

d. API

A specification of the functionality provided by a library modules. A programmer needs to understand the API to be able to use a module.

10. What colour is color\_rgb(0, 255, 255)?

a. yellow

b. cyan

c. magenta

d. orange

8/10 = 80%

Discussion

1. Pick an example of an interesting real-world object and describe it as a programming object by listing its data (attributes, what is “knows”) and its methods (behaviours, what it can “do”)

The clock.

Attributes

* Circular
* Has 2 progressively smaller circles
* Brown filling between the first 2 circles
* Lighter brown/yellow between the next circles
* White filling for the rest
* Smaller circle in the middle
* Filled grey
* Different birds at 60-degree intervals
* 3 arrows from the middles

Behaviours

* One arrow moving at every second
* One arrow moving every 60 seconds
* One arrow moving every 3600 seconds

2. Describe in your own words the object produces by each of the following operations from the graphics module. Be as precise as you can. Be sure to mention such things as the size, position, and appearance of the various objects. You may include sketches if that helps.

1. Point (130, 130)

Size: 1 Pixel

Position: Bottom right quadrant near the middle. Specifically, coming from the top left corner. Point x is 130 and Point Y is 130.

Appearance of various objects: a point in black.

1. C = Circle (Point (30, 40), 25)

C .setFill (“blue”)

c.setOutline (“red”)

Size: Circle with the radius of 30

Position: Top left corner at the co-ordinates 30,40

Appearance of various objects: Blue with a red outline

1. R = Rectangle (Point (20,20), Point (40, 40))

r.setFill (color\_rgb (0, 255, 250))

r.setWidth (3)

Size: Square’s area is 20 by 20

Position: Top left corner

Appearance of various objects: Square has a black outline with a Fluro green filling.

1. L = Line (Point (100,100), Point (100, 200))

l.setOutline(“red4”)

l.setArrow(“first”)

Size: 100 pixels line with an arrow head

Position: Line pulls from the bottom/middle of the page straight down the middle

Appearance of various objects: Red arrow head

1. Oval (Point (50,50), Point (60,100)

Size: Relatively small/thin oval on the upper left quadrant of the graph window

Position: Middle upper left corner

Appearance of various objects: The oval has a black outline with no filling

1. Shape = Polygon (Point (5, 5), Point ( 10, 10), Point (5,10). Point (10,5))

Shape.setFill (“orange”)

Size: 5 pixels worth X

Position: Top left corner

Appearance of various objects: X on the top left corner

1. T = Text (Point ( 100, 100), “Hello World!”)

T.setFace (“courier”)

t.setSize (16)

t.setStyle (“italic”)

Size: text size 16 – so relatively large

Position: middle of the graphics window

Appearance of various objects: Hello world! In the middle of the graphics object

3. Describe what happens when the following interactive graphics program runs:

A red circle is created with a radius of 10.

Where ever you click the circle will move up to 10 times in which the program will then close.

Programming exercises

1. After the program from the last discussion question in the following ways:

1. Make it draw squares instead of circles.

Check

1. Have each successive click draw an additional square on the screen (rather than moving the existing one).
2. Print a message on the window “Click again to quit” after the loop, and wait for a final click before closing the window.

Check the program

* Program needs to have a clone subject before move
* Have to create a second variable when creating additional squares
* Have to have a .getMouse at the end for the final click

2. An archery target consists of a central circle of yellow surrounded by the concentric rings of red, blue, black and white. Each ring has the same width which is the same as the radius of the yellow circle. Write a program that draws such a target.

*Hint:* Objects draw later will appear on top of objects drawn earlier.

Check code.

3. Write a program that draws some sort of face

Check code.

4. Write a program that draws a winder scene with a Christmas tree and a snowman.

Check code.

5.

**Chapter 5. Sequences: Strings, Lists, and Files.**

Objectives

* To understand the string data type and how strings are represented in the computer.
* To become familiar with various operations that can be performed on strings through built0in functions and string methods.
* To understand the basic idea of sequences and indexing as they apply to Python strings and lists.
* To be able to apply string formatting to produce attractive, informative program out.
* To understand basic file-processing concepts and techniques for reading and writing text files in Python.
* To understand basic concepts of cryography.
* To understand and write programs that process textual information.

5.1 The string data type.

Slicing is a way of indexing a range of positions in the string. Slicing takes the form <string>[<start> : <end>].

A slice produces a substring starting at the positions given by start and running up to, but not including positions end.

Indexing and slicing are useful operations for chopping strings into smaller pieces. The string data type also supports operations for putting strings together.

Two handy operators are concatenation (+) and repetition (\*).

Concatenation builds a string by “gluing” two strings together. Whilst Repetitions builds a string by multiple concatenations fo a string with itself.

Another useful function is len, which tells how many characters are in a string.

5.2 Simple String Processing

\n inputs a space between your code when the code is entered.

5.3 Lists as Sequences

Strings are always sequencing of characters, whereas lists can be sequences of arbitrary objects. You can create a list of numbers or a list of strings. In fact, you can even mix it up and create a list that contains both numbers and strings.

Lists are mutable. This means that the value of an item in a list can be modified with an assignment statement. Strings, on the other hand, cannot be changed “in place”.

5.4 String Representation and Message Encoding

5.4.1 String Representation

Python provides a couple of built in functions that allow us to switch back and further between characters and the numeric values used to represent them in strings. The ord functions returns the numeric (“ordinal) code of a single character string, while chr fors the other direct.

5.5. String Methods

5.5.1 Programming a Decoder

Decoding algorithm:

Get the sequence of numbers to decode

Message = “ “

For each number in the input:

Covert the number to the corresponding Unicode character

Add the character to the end of message

Print message

The accumulator variable message is initialised to be an empty string; that is, a string that contains no characters (“”).

\*Problem with the above code is that how do we decode the sequence of numbers

Here is an example of the correct code

Get the sequence of numbers as a string, inStrings

Split inString into a sequence of smaller strings

Message = “ “

For each of the smaller strings:

Change the string of digits into the number it represents

Append the Unicode character for that number to message

Print message

Split methods split a string into a List of substrings.

5.7 From Encoding to Encryption

Each character of the original message, called the plaintext, is replaced by a corresponding symbol in our case number) from a cipher alphabet. The resulting code is called the *ciphertext*.

Encryption approaches come in two flavours: private key and public key.

In a private key (also called a *share key*) system, the same key is used for encrypting and decrypting messages. All parties that wish to communicate need to know and key, but it must be kept secret from the outside world. This is the usual system that people think of when considering secret codes.

In public key systems, there are separate but related keys for encrypting and decrypting.

5.8 Input/Output as String Manipulation

5.8.1 Example application: Date Conversion

Often it is also necessary to turn a number into a stirng. In Python, most data types can be converted into strings using the str function.

Check chapter 5 code

By turning value into a string, we can use string concatenation to put a period at the end of a sentence. If we didn't first turn value into a string, Python would interpret the + as a numerical operation and produce an error, because "." is not a number.

Table 5.2 summarizes these four Python type conversion functions.

Function

|  |  |
| --- | --- |
| Function | Meaning |
| float(<expr>)  int(<expr>)  str(<expr>)  eval(<expr>) | Convert expr to a floating-point value.  Convert expr to an integer value.  Return a string representation of expr.  Evaluate string as an expression. |

5.8.2 String Formatting