Decision Structures and Boolean Logic

1. The if Statement

The simplest is the **sequence structure** where statements are executed in the order in which they appear.

In a **decision structure** statements are only executed under certain circumstances. The simplest decision structure is the `if` statement.

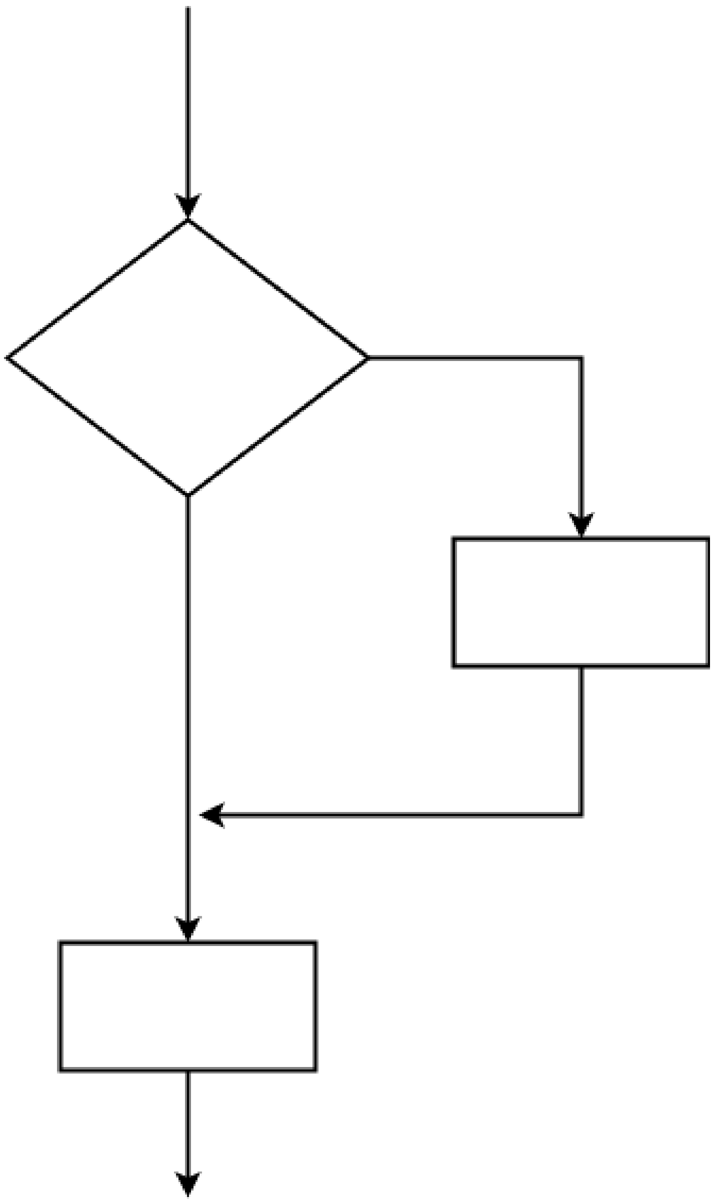


Figure 1: Example of an `if` structure

In Python syntax[1] an `if` statement is written as

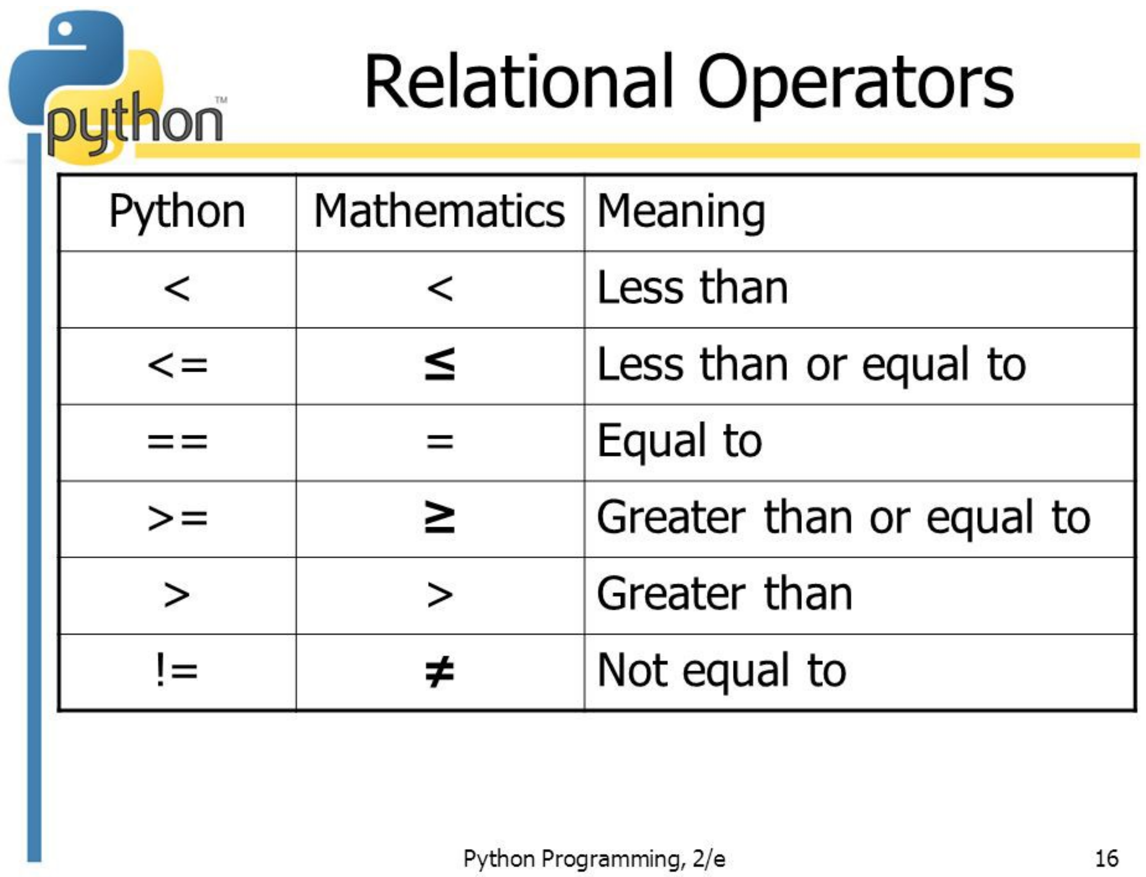
```
if condition:
     statement
     statement
    …
statement
```

2. Boolean Expressions and Relational Operators

In the above the `condition` tested by the `if` statement is either `True` or `False`.

We say that a condition which is either true or false is a **Boolean expression[2].** A Boolean expression is made up of relational and logical operators. We will deal with logical operators later. A list of relational operators is given below.

[1] Remember, **syntax** is just a word meaning the structure of statements in a programming language.
[2] Named after the mathematician George Boole (1815-1864), who was the first professor of mathematics at Queen's College Cork (or, as it is now known, University College Cork).



Relational Operators

Python	Mathematics	Meaning
<	$<$	Less than
<=	$\leq$	Less than or equal to
==	$=$	Equal to
>=	$\geq$	Greater than or equal to
>	$>$	Greater than
!=	$\neq$	Not equal to

Python Programming, 2/e 16

Example

The Python interpreter can be used in interactive mode to evaluate expressions, and display the value as either `True` or `False`.

```
>>> x=3   Enter

>>> y=2   Enter

>>> x>y   Enter

True
```

Example

`test_average.py` (Program 3-1 from Gaddis Starting Out with Python)

```
# This program gets three test scores and displays
# their average.  It congratulates the user if the
# average is a high score.

# The high score variable holds the value that is
# considered a high score.
high_score = 95

# Get the three test scores.
test1 = int(input('Enter the score for test 1: '))
test2 = int(input('Enter the score for test 2: '))
test3 = int(input('Enter the score for test 3: '))

# Calculate the average test score.
average = (test1 + test2 + test3) / 3

# Print the average.
print('The average score is', average)

# If the average is a high score,
# congratulate the user.
if average >= high_score:
    print('Congratulations!')
    print('That is a great average!')

# end of if structure
```

3. The `if-else` Statement

The `if-else` statement allows for decisions with two possible paths; one if the condition is `True` and the other if it is `False`. This is called a dual alternative decision structure.

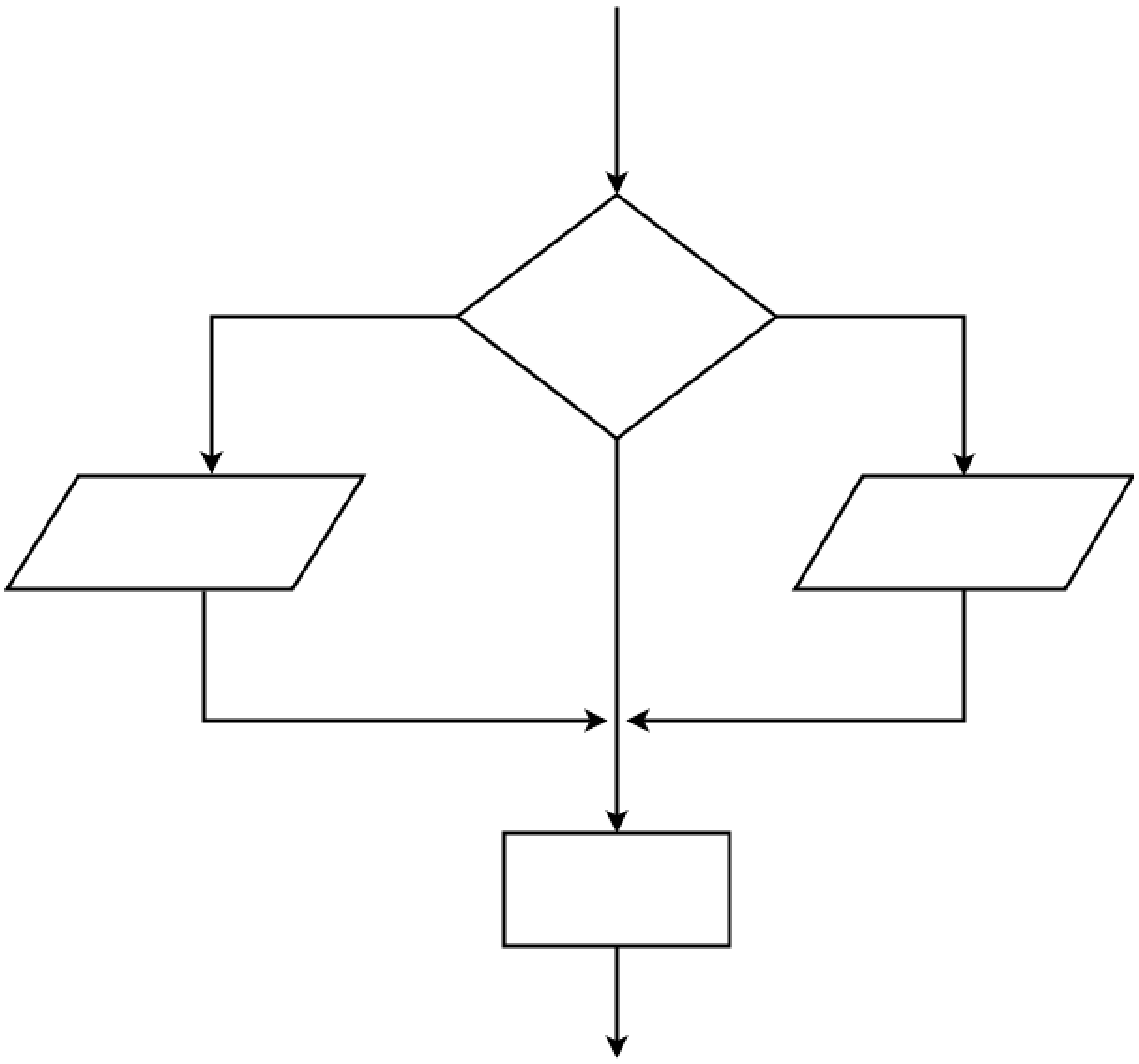


Figure 2: Example of an `if-else` Structure

In Python the syntax of the `if-else` statement is

```
if condition:
    statement
    statement
    …
else:
    statement
    statement
    …
statement
```

Example

What is the python code corresponding to the flow chart in Figure 2?

4. The **`if-elif-else`** Statement

It is not unusual to have a whole series of conditions to test, with a different outcome depending on which condition is true. In principle such a scenario can always be dealt with by using multiple `if-else` statements, but the `if-elif-else` statement makes life easier. The Python key word `elif` is short for the English phrase else if.

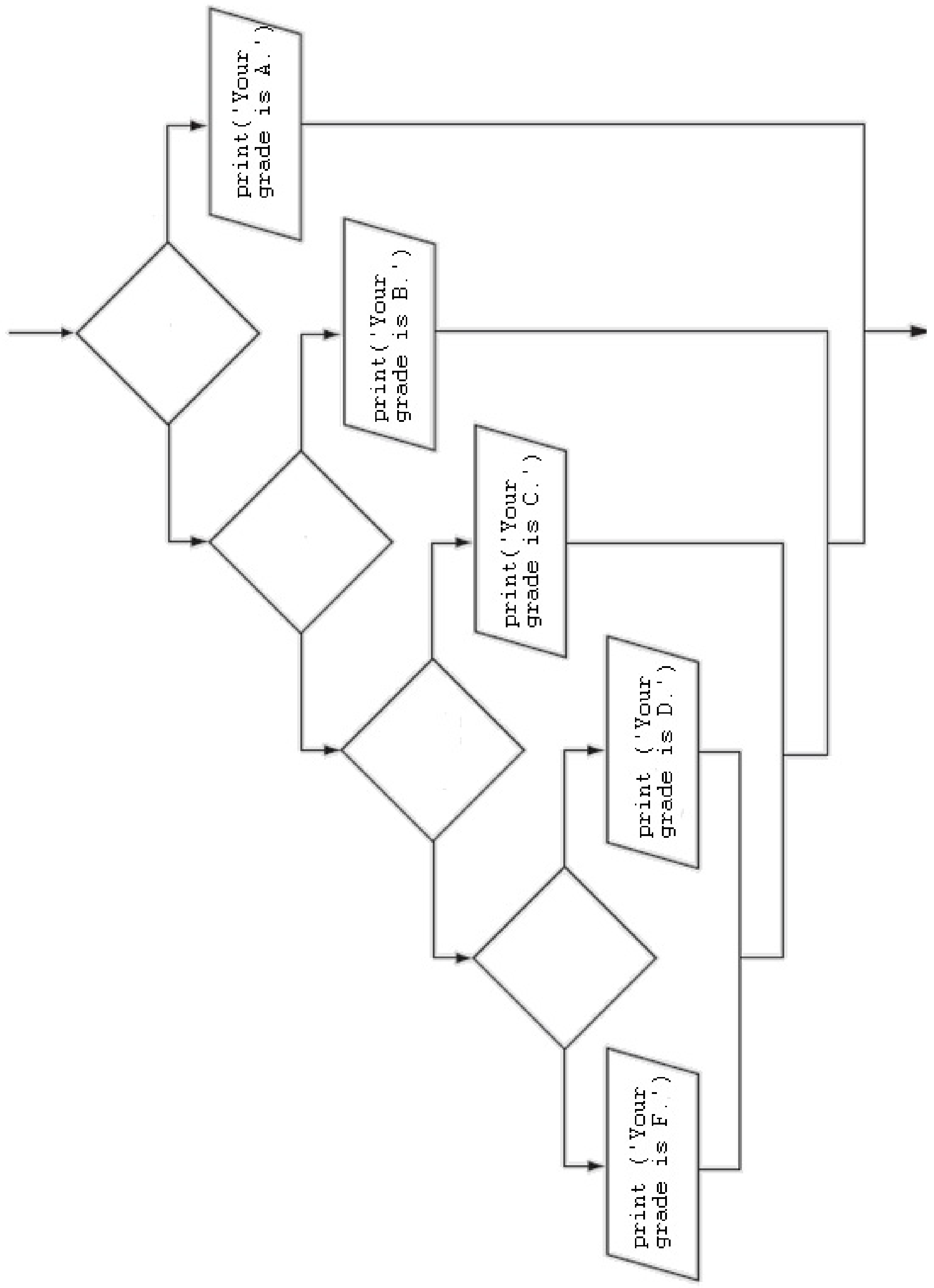

print('Your grade is A.')
print('Your grade is B.')
print('Your grade is C.')
print ('Your grade is D.')
print ('Your grade is F.')

Figure 3 An `if-elif-else` Ladder

The Python syntax for the `if-elif-else` statement is

```
if condition_1:

     statement
     statement
     …

elif condition_2:

     statement
     statement
     …







else:

     statement
     statement

statement
```

Example `grader.py`

```python
# This program gets a numeric test score from the
# user and displays the corresponding letter grade.

# Variables to represent the grade thresholds
A_score = 90
B_score = 80
C_score = 70
D_score = 60

# Get a test score from the user.
score = int(input('Enter your test score: '))

# Determine the grade.
if score >= A_score:
    print('Your grade is A.')
elif score >= B_score:
    print('Your grade is B.')
elif score >= C_score:
    print('Your grade is C.')
elif score >= D_score:
    print('Your grade is D.')
else:
    print('Your grade is F.')
# End of if-elif structure
```

5. Comparing Strings

So far we have been using Boolean expressions to compare numbers, but Python will also allow us to compare strings. The Python interpreter can be used in interactive mode to compare, and display the value as either `True` or `False`.

Example

```
>>> ‘a’ == ‘a’   Enter
```

```
>>> apple = ‘fruit’   Enter

>>> pear = ‘fruit’    Enter

>>> pear != apple     Enter
```

```
>>> ‘a’ < ‘b’
```

This last answer may look a little surprising! However what Python is actually checking is the ASCII[3] code for the characters. In ASCII

Characters A-Z are represented by numbers 65-90

Characters a-z are represented by numbers 97-122

Characters 0-9 are represented by numbers 48-57

Blank space is represented by the number 32

[3] ASCII stands for American Standard Code for Information Interchange.

This means that

Example

Which of the below will return a `True`?

```
 'Mary' > 'Mark'

 'mary' > 'Mark'

 'Mar' > 'Mark'

'MarK' > 'Mark'
```

M	a	r	k
77	97	114	107

M	a	r	y
77	97	114	121

M	a	r
77	97	114

m	a	r	y
109	97	114	121

M	a	r	K
77	97	114	75

Example

password.py

```
# This program compares two strings.
# Get a password from the user.

#set password
PASSWORD = 'thegrail'

user_password = input('Enter the password: ')

# Determine whether the correct password
# was entered.

if user_password == PASSWORD:
    print('Password accepted.')
else:
    print('Sorry, that is the wrong password.')
# end of if-else structure
```

6. Logical Operators and Truth Tables

The Logical operators `and`, `or` and `not` allow us to connect two logical expressions together. Using combinations of `and`, `or` and `not` we can construct more complex statements which are called **compound propositions**.

Consider two logical expressions A and B, each of which can be either `True` or `False`. The truth tables below show how they are combined in the `and`, `or` and `not` operators. Note that these definitions follow very closely the ‘common sense’ meaning of the words.

Example

What is the truth value of the statement ‘either the moon is made of green cheese **and** Henry VIII had six wives **or** it is **not** true that the Dodo is extinct’?

7. Logical Equivalence

Example

Show that **not(P and Q)** is logically equivalent to **(not P) or (not Q).**

A few examples of these in Python are given in the following three code fragments.

```
if temperature <= 20 and time >= 30:

    print('The temperature needs to be adjusted.')
```

```
if temperature  < 0 or temperature > 100:

    print('The temperature levels are unsafe.')
```

```
if not temperature==100:

    print('The water has not boiled yet.')
```

Note that though all of the above are perfectly accurate Python code it is wise to place brackets around the statements as follows to aid readability:

```
if (temperature <= 20) and (time >= 30):
```

```
if (temperature  < 0) or (temperature > 100):
```

```
if not (temperature==100):
```

8. Boolean Variables

So far you have met three types of variables in Python: `int`, `float` and `str`, i.e. numbers which are integers, are floating point decimals, or are character strings. One more is the `bool` data type, which can be either `True` or `False`.

Boolean variables are often used as flags. A flag is a variable which signals if a particular condition is met (`True`) or not (`False`). Thus a program could contain:

```
if temperature < 0:

    freezing_temp = True

else:

    freezing_temp = False
```

and later in the code it could test this flag via

```
if  freezing_temp:

    print('You need to use some de-icer.')
```

9. Turtle Graphics

`turtle.heading()` returns the turtle's heading in degrees
`turtle.setheading(56)` sets the turtle heading to 56 degrees

`turtle.pencolor()` returns the color of the pen
`turtle.pencolor('blue')` sets the pen colour to `blue`.
`turtle.fillcolor()` returns the fill color
`turtle.fillcolor('white')` sets the fill colour to `white`
`turtle.bgcolor()` returns the background colour of the turtle window
`turtle.bgcolor('gray')` sets the background colour to `gray`.

Note there are several hundred possible colours to choose from when using turtle! All the ones common ones you would guess are present (`green, black, red, yellow` etc) but there are also more esoteric colours such as `lavender blush`, `dark sea green` and `plum3`.[4]

`turtle.pensize()` returns the thickness of the pen
`turtle.pensize(5)` sets the thickness of the pen to 5

`turtle.speed()` returns the speed of the turtle animation
`turtle.speed(5)` sets the animation speed to 5

Note the animation speed can be in the range 1-10 (from slowest to fastest). Setting the speed to 0 means the turtle moves 'instantaneously'.

`Turtle.isdown()` returns `True` if the pen is down or `False` otherwise.
`Turtle.isvisible()` returns `True` if the turtle is visible or `False` otherwise.

[4] See www.tcl.tk/man/tcl8.4/TkCmd/colors.htm, or Appendix D of the textbook *Starting Out with Python* by Tony Gaddis, for the complete list.

Supportive Reading

Tony Gaddis, *Starting Out with Python*, 4/e Pearson, 2019, Chapter 3.