Week 4: Conditional statements and strings

Intended learning outcomes

By the end of this class, you will be able to:

- write if and else statements to perform conditional operations;
- interpret and resolve type errors;
- define, interrogate, concatenate, and print strings.

Note. By default, Google Colab uses 2 spaces for indentation, not 4 spaces (as used in these worksheets). To fix this, select "Tools → Settings → Editor" and set "Indention width in spaces" to 4 .

1 The conditional statement: `if` and `else`

In English we have the if-then construction: "*if* the soil is dry *then* I water my plants". I check to see whether not the statement "the soil is dry" is true, and in the case that it is, I perform the operation "water my plants".

Concept 1. A **conditional statement** in a program is a statement that only performs some operations when some condition is met.

In Python we use the `if` command for conditional statements. Type in the following program, taking care to notice the colon and the indentation. What do you think the program will do? (Remember that `a % b` is the remainder when a is divided by b.)

```
# A program to see if a number is odd.
number = int(input("Enter an integer: "))
if number % 2 == 1:
    print("That is odd.")
    print("But not too odd.")
print("Thank you.")
```

Run this program a few times and understand which print statements are printed when. Note how the indentations make a difference.

In general, a Python `if` statement has the following form:

```
if boolean_expression:
    do_this
    and_do_this
carry_on_here
```

The indented lines are executed only if the Boolean expression is `True`, otherwise they are ignored. There can be more than two indented lines.

If you want to do something different in the case that the Boolean expression is `False`, you can use the `else` statement. Add two lines to the middle of your program so that it now looks like this (taking care with colons and indentations):

```
# A program to see if a number is odd or even.
number = int(input("Enter an integer: "))
if number % 2 == 1:
    print("That is odd.")
    print("But not too odd.")
else:
    print("That is even.")
print("Thank you.")
```

Again, run it a few times to see what happens.

Exercise 4.1. Write a program using `if` and `else` (but not using `max()`) that inputs two numbers and prints out the larger of the two.

2 The string type: `str`

A string is basically a piece of text, but it can have other symbols in it.

```
>>> menu_item = "Spam & eggs"
>>> print(menu_item)
>>> type(menu_item)
```

A string is denoted by having quotation marks at each end, *either* using double quotation marks (`"`) *or* a single quotation mark (`'`), but with the same at each end of the string. Use the up-arrow to change the first line above to

```
>>> menu_item = 'Spam & eggs'
```

Then re-execute the next two lines, make sure that it gives the same answer.

To join together, or *concatenate*, two strings you use the + operator.

```
>>> special_menu_item = menu_item + " with extra spam"
>>> print(special_menu_item)
```

We can use + on numbers and on strings. What might happen if you add a number to a string? Guess what the following would output, then type it in.

```
>>> string = "6"
>>> number = 5
>>> print(number + string)
```

Python doesn't know what to do: this is called a *type error*. We need to make sure that they are both strings, or both numbers. We can try to convert a string to a number using `float()` or `int()`, and can convert a number to a string using `str()`. Which of the following commands will Python not like? Try and see.

```
>>> int("7")
>>> float("7")
>>> int("7.5")
>>> str(7)
>>> float("parrot")
```

We can now 'add' the number and string in two different ways.

```
>>> print(str(number) + string)
>>> print(number + int(string))
```

Make sure you understand why you get two different answers.

There is a Python function `len()` that tells you the length of a string.

```
>>> print(len("Hello"))
```

Exercise 4.2. Use the functions `str()` and `len()` to find out how many digits there are in 2^{100}. (On the last worksheet you had to count them by hand.) Why did you have to use `str()`?

String indices

Each character in a string has an index that denotes its position. Typing `string[i]` gives the character at index `i`. However it is important to note the following important convention:

Python starts counting from zero!

Counting from zero is a common feature of computer languages and there are good reasons for it, however, it often trips up beginners from time to time. This means that the index of the first character in a string is 0 and the index of the final character is one less than the length of the string.

```
>>> bird_name = "parrot"
>>> print(bird_name[0])
>>> print(bird_name[1])
>>> print(bird_name[4], bird_name[5])
```

What do you think would happen if you type in `bird_name[10]`? Try it!

Type in the following program, replacing `Your Name` by your own name. See if you can guess what it will do, then run it.

```
my_name = "Your Name"
index = 0
while index < len(my_name):
    print(index, my_name[index])
    index = index + 1
```

You can also count from the *end* of a string by using negative indices. In that case, an index of -1 corresponds to the last character in a string.

```
>>> "spam"[-1]
>>> "spam"[-3]
```

Print separation

When you use `print()` with a list of items, it places a space between each item.

```
>>> print("spam", "egg", "beans")
```

You can change this behaviour by adding `sep=str` to the end of the print statement, where `str` is the string you want to separate the items with. [You don't put spaces around this equals sign as it's not an assignment; we'll return to this in two weeks' time.] Change the last line you typed in to the following.

```
>>> print("spam", "egg", "beans", sep=":")
```

You can remove all separation between the items by using the empty string: `sep=""`. Try it in the above example.

Exercise 4.3. Write a program that takes a string as input. If the length of the string is odd it prints out the middle character. If the length is even it prints out the middle two characters.

Special characters

There are certain special characters you can include in a string by using the backslash '\', to form an *escape sequence*.

Escape sequence	Meaning
`\n`	newline
`\t`	tab
`\"`	"
`\'`	'
`\\`	\

A new line is obtained with `\n`.

```
>>> print("Hello\nHello\nHello")
```

A tab (like pressing the tab key in a word processor) is obtained with `\t`. This can be useful for aligning text. A tab will move the cursor to the next *tab stop* and these are located every eight characters.

```
>>> print("Hello\tHello\tHello\nBye\tBye\tBye\tBye")
```

Using the escape characters for inverted commas and apostrophes allows us to form strings like: `He said "Don't do that!"` This would otherwise not be possible. Try the following and then try it without the backslashes.

```
>>> print("He said \"Don\'t do that!\"")
```

Print ending

At the end of a `print` statement there is normally a newline. You can change this behaviour by adding `end=str` to the print statement. Try the following.

```
i = 0
while i < 100:
    print(i, end="\n")
    i = i + 1
```

Try replacing `end="\n"` with, e.g., `end="\t"`, `end=" "`, `end=""`, or `end=":"`.

Homework

1. Finish off the worksheet.

2. (*Quick review.*) Answer these without Python. Then **check** your answer. Ask, e.g. on the discussion board, if you do not understand what is happening.

(i) What *precisely* will the following output?

```
string = "hello"
i = 0
print(string)
while i < len(string):
    print(string[i], end=":")
    i = i + 1
print(string)
```

(ii) Which of the following will work as a password in the program below: curt, crowd, credit, cost, herring, point, pant? (At least one will work.)

```
attempt = input("Enter the password: ")
if attempt[2] == "password"[-2]:
    print("You may enter.")
else:
    print("Go away!")
```

3. (*Assessed homework.*) This week's homework involves writing a program to compute the **Hamming distance** between two strings. To access the homework, you must navigate to the following webpage:

`https://aim.shef.ac.uk/moodle/mod/quiz/view.php?id=425`

Your code will be run with some test strings. If it gives the correct answer you will get a mark, if it does not then you will get no mark. You can submit your homework anytime from now til 2pm on Monday of Week 5 (25th October).

4. (*Optional.*) In Python the `in` comparison operator can be used to test if a character is contained in a string. So `"a" in "spam"` gives `True`, whereas `"i" in "team"` gives `False`.

In the game **Fizz Buzz** you count out loud. When you reach a number divisible by 5, or containing the digit 5, you say 'fizz'. When you reach a number divisible by 7, or containing the digit 7, you say 'buzz'. If both these conditions are met, you say 'fizz buzz'. Counting in this way begins as follows:

1, 2, 3, 4, fizz, 6, buzz, 8, 9, fizz, 11, . . .

Write a program which counts upwards from one to a specified number, using the rules of Fizz Buzz.