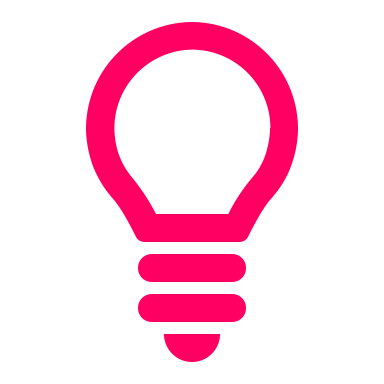


**Week 2 Class 2**

**Student Worksheet**

# **Before you read**

# **Predicting**

Look at the Word Cloud of the most frequent words in the text you are going to read. Based on the vocabulary, can you predict what the text is about?

If you do not know the words, use a dictionary.

Text

Description automatically generated

Keywords

**Which words do you:**

1. **know, can pronounce and can use in a sentence?**
2. **Do not know and have to look up in a dictionary?**

Check the dictionary and write down the words you do not know. You will need to, later when you read the text.

1. **Skimming**

Skim the text and match the headings with the sections.

|  |  |  |
| --- | --- | --- |
| Section 1 | \_\_\_\_\_\_\_\_\_ | 1. Additional types supported by Python |
| Section 2 | \_\_\_\_\_\_\_\_\_ | 1. An Informal Introduction to Python |
| Section 2.1 | \_\_\_\_\_\_\_\_\_ | 1. Using Python as a Calculator |
| Section 3 | \_\_\_\_\_\_\_\_\_ | 1. Numbers |

**II. While you read**

**You are going to read an article about coding in Python and numbers. Before you read, decide whether the statements below are ‘T’ (true) or ‘F’ (false). Then read the text on the next page to confirm or correct your answers.**

|  |  |  |
| --- | --- | --- |
|  | **True** | **False** |
| 1. Comments in Python start with a **#**. |  |  |
| 1. A comment can appear inside a string literal. |  |  |
| 1. The expression syntax in Python is similar to other languages because it uses similar symbols. |  |  |
| 1. There are 2 types of integer numbers: Int and float. |  |  |
| 1. Python only supports int and float numbers. |  |  |

# **Section 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

In the following examples, input and output are distinguished by the presence or absence of prompts ([>>>](https://docs.python.org/3/glossary.html#term-0) and […](https://docs.python.org/3/glossary.html#term-1)): to repeat the example, you must type everything after the prompt, when the prompt appears; lines that do not begin with a prompt are output from the interpreter. Note that a secondary prompt on a line by itself in an example means you must type a blank line; this is used to end a multi-line command.

Many of the examples in this manual, even those entered at the interactive prompt, include comments. Comments in Python start with the hash character, #, and extend to the end of the physical line. A comment may appear at the start of a line or following whitespace or code, but not within a string literal. A hash character within a string literal is just a hash character. Since comments are to clarify code and are not interpreted by Python, they may be omitted when typing in examples.

Some examples:

*# this is the first comment*

spam = 1 *# and this is the second comment*

*# ... and now a third!*

text = "# This is not a comment because it's inside quotes."

## **Section 2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Let’s try some simple Python commands. Start the interpreter and wait for the primary prompt, >>>. (It shouldn’t take long.)

### Section 2.1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The interpreter acts as a simple calculator: you can type an expression **at** it and it will write the value. Expression syntax is straightforward: the operators +, -, \* and / work just like in most other languages (for example, Pascal or C); parentheses (()) can be used for grouping. For example:

>>>

**>>>** 2 + 2

4

**>>>** 50 - 5\*6

20

**>>>** (50 - 5\*6) / 4

5.0

**>>>** 8 / 5 *# division always returns a floating point number*

1.6

The integer numbers (e.g. 2, 4, 20) have type [int](https://docs.python.org/3/library/functions.html#int), the ones with a fractional part (e.g. 5.0, 1.6) have type [float](https://docs.python.org/3/library/functions.html#float). We will see more about numeric types later in the tutorial.

Division (/) always returns a float. To do [floor division](https://docs.python.org/3/glossary.html#term-floor-division) and get an integer result (discarding any fractional result) you can use the // operator; to calculate the remainder you can use %:

>>>

**>>>** 17 / 3 *# classic division returns a float*

5.666666666666667

>>>

**>>>** 17 // 3 *# floor division discards the fractional part*

5

**>>>** 17 % 3 *# the % operator returns the remainder of the division*

2

**>>>** 5 \* 3 + 2 *# result \* divisor + remainder*

17

With Python, it is possible to use the \*\* operator to calculate powers [1](https://docs.python.org/3/tutorial/introduction.html#id3):

>>>

**>>>** 5 \*\* 2 *# 5 squared*

25

**>>>** 2 \*\* 7 *# 2 to the power of 7*

128

The equal sign (=) is used to assign a value to a variable. Afterwards, no result is displayed before the next interactive prompt:

>>>

**>>>** width = 20

**>>>** height = 5 \* 9

**>>>** width \* height

900

If a variable is not “defined” (assigned a value), trying to use it will give you an error:

>>>

**>>>** n *# try to access an undefined variable*

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'n' is not defined

There is full support for floating point; operators with mixed type operands convert the integer operand to floating point:

>>>

**>>>** 4 \* 3.75 - 1

14.0

In interactive mode, the last printed expression is assigned to the variable \_. This means that when you are using Python as a desk calculator, it is somewhat easier to continue calculations, for example:

>>>

**>>>** tax = 12.5 / 100

**>>>** price = 100.50

**>>>** price \* tax

12.5625

**>>>** price + \_

113.0625

**>>>** round(\_, 2)

113.06

This variable should be treated as read-only by the user. Don’t explicitly assign a value to it — you would create an independent local variable with the same name masking the built-in variable with its magic behavior.

**Section 3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

In addition to [int](https://docs.python.org/3/library/functions.html#int) and [float](https://docs.python.org/3/library/functions.html#float), Python supports other types of numbers, such as [Decimal](https://docs.python.org/3/library/decimal.html#decimal.Decimal) and [Fraction](https://docs.python.org/3/library/fractions.html#fractions.Fraction). Python also has built-in support for [complex numbers](https://docs.python.org/3/library/stdtypes.html#typesnumeric), and uses the j or J suffix to indicate the imaginary part (e.g. 3+5j).

**III. After you read**

1. **Vocabulary**
2. The Word Cloud below show the most frequent verbs in the text. Find them and highlight them in the text. Do you know what they mean? **If not, look them up in a dictionary.**

<https://www.wordreference.com/>

Text

Description automatically generated

1. **Common Adjectives in Python language**

In Python literature, it is common to find adjectives. Look at the word cloud below. **Do you know all these adjectives?**

Text

Description automatically generated

Adjectives usually go with nouns and form collocations: 2 or 3 words that usually go together. Here are some common collocations with the adjectives from the Cloud. Write your own definition for each one. You can ask your coding instructor to help you.

|  |  |
| --- | --- |
| **Collocation** | **Definition/translation** |
| **Floating** point |  |
| **Interactive** prompt |  |
| **Fractional** part |  |
| **Primary** prompt |  |

Some of the adjectives can also be nouns. Check the following words in the text and decide if they are adjective- describen algo, or they are noun- denominan un objeto o una persona.

|  |  |  |
| --- | --- | --- |
| **Word** | **Adjective or Noun?** | **What does it mean?** |
| **Variable** |  |  |
| **prompt** |  |  |
| **Decimal** |  |  |
| **Tutorial** |  |  |

1. **Prepositions and connectors**

In your class, you learnt some prepositions to talk about place and time. The prepositions in the Word Cloud below are used in the text. Find them and organize them in the table below based on what they express in the text. You can also add your own category.

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Time |  |
| Place |  |
| Sequence |  |
| Company |  |
| Comparison |  |
| Add your category |  |
| Add your category |  |

**Answer Key**

1. **Before you read**
2. **Skimming**

Skim the text and match the headings with the sections.

|  |  |  |
| --- | --- | --- |
| Section 1 | \_\_\_\_\_b\_\_\_\_ | 1. Additional types supported by Python |
| Section 2 | \_\_\_\_\_c\_\_\_\_ | 1. An Informal Introduction to Python |
| Section 2.1 | \_\_\_\_\_d\_\_\_\_ | 1. Using Python as a Calculator |
| Section 3 | \_\_\_\_\_a\_\_\_\_ | 1. Numbers |

**II. While you read**

1. Comments in Python start with a **#**. T
2. A comment can appear inside a string literal.  **F**
3. The expression syntax in Python is similar to other languages because it uses similar symbols**. T**
4. There are 2 types of integer numbers: Int and float. **T**
5. Python only supports int and float numbers. **F**
6. **After you read**

|  |  |
| --- | --- |
| **Word** | **Adjective or Noun?** |
| **Variable** | Noun |
| **prompt** | Noun |
| **Decimal** | noun |
| **Tutorial** | noun |

|  |  |
| --- | --- |
| Time | / |
| Place | In, within |
| Sequence | After, before |
| Company | with |
| Comparison | like |
| Add your category  Reason | Because of  Since – debido a que |
| Add your category |  |