Week 1.1
Asynchronous Session

Learning Vocabulary: Glossaries and Cognates

Explanation

1. What are glossaries?

A **glossary** is an alphabetical list of words, phrases, and abbreviations with their definitions. **Glossaries are** important for students to learn the specific language (words, phrases, and abbreviations) from a specific discipline or technology area. Some glossaries provide the pronunciation of the term.

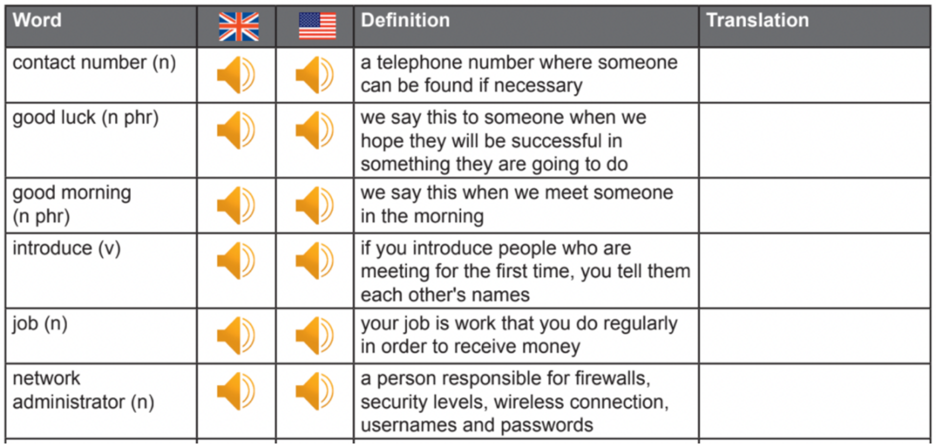
Glossary entry

Argument

In coding, **argument** refers to **extra information** which the computer uses to perform commands.

Typing cartoons illustration made by Storyset: https://storyset.com/illustration/typing/bro

Many course books contain glossaries. Look at the **example below**:



Word	🇬🇧	🇺🇸	Definition	Translation
contact number (n)	🔊	🔊	a telephone number where someone can be found if necessary	
good luck (n phr)	🔊	🔊	we say this to someone when we hope they will be successful in something they are going to do	
good morning (n phr)	🔊	🔊	we say this when we meet someone in the morning	
introduce (v)	🔊	🔊	if you introduce people who are meeting for the first time, you tell them each other's names	
job (n)	🔊	🔊	your job is work that you do regularly in order to receive money	
network administrator (n)	🔊	🔊	a person responsible for firewalls, security levels, wireless connection, usernames and passwords	

Example Glossary from an IT book

2. Organizing your glossaries

The best way to learn specific coding vocabulary is by organizing **your own glossary (tu propio glosario)**. To create them, you need the term, definition and translation. In coding, commands and codes are usually **NOT TRANSLATED**.

Term	Definition	Translation (if any)
Arguments	Extra information which the computer uses to perform commands.	*argumentos*
Block	Section of code which is grouped together	*bloque*
Argparse	Argparse is a parser for command-line options, arguments and subcommands.	*Es un comando propio del programa. No se traduce.*
Assert	Used during debugging to check for conditions that ought to apply	*Es un comando propio del programa. No se traduce.*

3. Alphabet

Programmers often use **acronyms**. To use acronyms, you first need to learn the **alphabet**. Follow the link below and explore the material on alphabet. Pay special attention to:

1. The **sound of each letter** when it is pronounced separately
2. The **pronunciation** of specific language **examples** related to Python.

https://view.genial.ly/607cf25b05992b0cfe9a0e9d

Application

Part 1. Look at the words in the box. Match them with their definitions from the glossary.

(a) conditional statement (b) class (c) debugging (d) def

________	The process of finding and removing programming errors or bugs
________	Defines a function or method
________	Statement that contains an "if" or "if/else".
________	A template for creating user-defined objects.

Activity 2.Practice

Part 2. Reading Strategy Cognates

Cognates are words **in two languag**es that are similar and usually have similar meaning.

Interesting Fact

30-40% of all words in English have a cognate in Spanish. For Spanish speakers, cognates are **an obvious bridge** to the English language.

Table below gives you some examples of cognates in the text you are going to read. As you read, mark all cognates you see. How are they pronounced?

English	Spanish
visually	visualmente
special	especial
logical	lógico
important	importante
incorrectly	incorrectamente

Be careful with pronouncing cognates! Look them up in a dictionary.

Part 3. Activating Prior Knowledge. Guess the answers. Then read the text and check if you were right.

1. Assert is one of the 35 keywords in Python.   T F
2. A string is an invalid argument to a mathematical function expecting a number. T F
3. Docstrings are comments that help people understand the coding block. T F
4. Indentations can damage the code. T F
5. Python only uses 0 and 1 to code. T F
6. Continue is one of the built-on functions in Python. T F

Part 4. While Reading. You are going to read a text about Python. As you read, try to find the information you answered in Part 1.
You will find **a glossary of terms** at the end of this document. Use it to understand the text.

What is Python?

Programming isometric Semi Flat Illustrations made by Story <a href="https://storyset.com/web">Illustration by Freepik Storyset</a>

Python is a programming language. It integrates dynamic semantics for web and app development. It is very attractive because it offers dynamic typing and dynamic binding options.

Python is simple and easy to learn. It uses a syntax that focuses on readability. Developers can **read** and translate Python code much easier than other languages.

How is Python Used?

Python is easy to read. Its formatting is visually simple, and it often uses English keywords-other programming languages use punctuation. Python does not use curly brackets to delimit **blocks**, and semicolons after statements are allowed but are rarely used. It has fewer syntactic **exceptions** and special cases than C or Pascal.

Keywords

Python has 35 keywords or reserved words; they cannot be used as identifiers.

`and`	`break`	`elif`	`for`	`in`	`not`	`True`
`as`	`class`	`else`	`from`	`is`	`or`	`try`
`assert`	`continue`	`except`	`global`	`lambda`	`pass`	`while`
`async`	`def`	`False`	`if`	`None`	`raise`	`with`

await	del	finally	import	nonlocal	return	yield

Indentation

Python uses whitespace to delimit control flow **blocks** (following the off-side rule). Python borrows this feature from its predecessor ABC: instead of punctuation or keywords, it uses indentation to indicate the run of a block.

Incorrectly indented code could be misread by a human reader differently than it would be interpreted by a **compiler** or interpreter. This example illustrates an error introduced by incorrect indentation:

```
def foo(x):
    if x == 0:
        bar()
        baz()
    else:
        qux(x)
    foo(x - 1)
```

Can you notice the error in the `if`/`else` **conditional statement**? This process is called **debugging**, which is an important process in software development because it helps the programmer to find errors.

Data structures

Python is a dynamically typed language. Python values, not variables, carry type information. This has implications for many aspects of the way the language functions.

All variables in Python hold references to objects, and these references are passed to functions; a function cannot change the value of variable references in its calling function (but see below for exceptions)

Among dynamically typed languages, Python is moderately type-checked.

Literals & Operators

Python has various kinds of strings literals, for example: Normal Strings, Multi-line strings, Raw strings, Concatenation of adjacent strings.

Python can use numbers, like, `0`, `-1`, `3.4`, `3.5e-8` and also list, tiples, sets, dictionaries, for example:

```
    a_list = [1, 2, 3, "a dog"]
    a_dictionary = {"key 1": "value 1", 2: 3, 4: []}
```

It also uses Arithmetic operators for example: `+`, `-`, `*`, `/` comparison operators such as: `==`, `!=`, `<`, `>`, `<=` and logical operators like: `and` `or`

Important: In Python, the left operand is always evaluated before the right operand. That also applies to function **arguments**. This is known as evaluation order.

Functional Programming

In Python, a function is a group of related statements that performs a specific task.

Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized.

Python also offers the following built-in functions:

`abs()`	`delattr()`	`hash()`	`memoryview()`	`set()`
`all()`	`dict()`	`help()`	`min()`	`setattr()`
`any()`	`dir()`	`hex()`	`next()`	`slice()`
`ascii()`	`divmod()`	`id()`	`object()`	`sorted()`
`bin()`	`enumerate()`	`input()`	`oct()`	`staticmethod()`
`bool()`	`eval()`	`int()`	`open()`	`str()`
`breakpoint()`	`exec()`	`isinstance()`	`ord()`	`sum()`
`bytearray()`	`filter()`	`issubclass()`	`pow()`	`super()`
`bytes()`	`float()`	`iter()`	`print()`	`tuple()`
`callable()`	`format()`	`len()`	`property()`	`type()`
`chr()`	`frozenset()`	`list()`	`range()`	`vars()`

`classmethod()`	`getattr()`	`locals()`	`repr()`	`zip()`
`compile()`	`globals()`	`map()`	`reversed()`	`__import__()`
`complex()`	`hasattr()`	`max()`	`round()`	

Exceptions

Python supports (and extensively uses) **exception handling** as a means of testing for error conditions and other "exceptional" events in a program. Indeed, it is even possible to trap the exception caused by a syntax error.

Python style calls for the use of exceptions whenever an error **condition** might arise. Rather than testing for access to a file or resource before actually using it, it is conventional in Python to just go ahead and try to use it, catching the **exception** if access is rejected.

Comments and docstrings

Comments are descriptions that help programmers better understand the intent and functionality of the program. They are completely ignored by the Python interpreter.

In Python, we use the hash symbol `#` to write a single-line comment.

Docstrings (documentation strings) are strings that are located alone without assignment as the first indented line within a module, class, method, or function, intended to store a human-readable description of the object's purpose, behavior, and usage.

Example:

```
>>> print "Hello World"  # print is a statement
Hello World
>>> from __future__ import print_function
>>> print "Hello World"
  File "<stdin>", line 1
    print "Hello World"
                      ^
```

```
SyntaxError: invalid syntax
>>> print("Hello World") # print become a function
Hello World
```

Part 5. After Reading. Read the article again and compare your answers in Part 1. Were you right?

Bring your answers to class and discuss them with your teacher and classmates.

	Self-Assessment	✓	✗
1	I understand why glossaries are important.		
2	I know 20 new words related to Python.		
3	I can understand basic texts about Python with a use of glossaries.		
4	I can pronounce Python words from this lesson correctly.		
5	I can spell Python related vocabulary correctly.		
My goal:			
I should study….			

Python Glossary 0-20

This glossary is meant to be a quick reference guide to Python. Use it in your coding classes.

Term	Definition	Translation (write your own)
abs	Return the absolute value of a number.	
argument	Extra information which the computer uses to perform commands.	
argparse	Argparse is a parser for command-line options, arguments and subcommands.	
assert	Used during debugging to check for conditions that ought to apply	
assignment	Giving a value to a variable.	

block	Section of code which is grouped together	
break	Used to exit a for loop or a while loop.	
class	A template for creating user-defined objects.	
compiler	Translates a program written in a high-level language into a low-level language.	
continue	Used to skip the current block, and return to the "for" or "while" statement	
conditional statement	Statement that contains an "if" or "if/else".	
debugging	The process of finding and removing programming errors.	
def	Defines a function or method	
dictionary	A mutable associative array (or dictionary) of key and value pairs. Can contain mixed types (keys and values). Keys must be a hashable type.	
distutils	Package included in the Python Standard Library for installing, building and distributing Python code.	

docstring	A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition.	
__future__	A pseudo-module which programmers can use to enable new language features which are not compatible with the current interpreter.	
easy_install	Easy Install is a python module (easy_install) bundled with setuptools that lets you automatically download, build, install, and manage Python packages.	
evaluation order	Python evaluates expressions from left to right. Notice that while evaluating an assignment, the right-hand side is evaluated before the left-hand side.	
exceptions	Means of breaking out of the normal flow of control of a code block in order to handle errors or other exceptional conditions	

Answer Key

Part 1. **Look at the words in the box. Match them with their definitions from the glossary.**

(a) conditional statement (b) class (c) debugging (d) def

____C____	The process of finding and removing programming errors or bugs
___D_____	Defines a function or method
___A_____	Statement that contains an “if” or “if/else”.

_____B____	A template for creating user-defined objects.

Part 1 Cognates posible answers:
Delimit, human, comments, value, variable, dynamically, etc.

Part 2. Activating Prior Knowledge. Guess the answers. Then read the text and check if you were right.

1. Assert is one of the 35 keywords in Python. **T**
2. A string is an invalid argument to a mathematical function expecting a number. T
3. Docstrings are comments that help people understand the coding block. T F
4. Indentations can damage the code. T
5. Python only uses 0 and 1 to code. F
6. Continue is one of the built-on functions in Python. F