Python glossary of terms

Adapted from InteractivePython.org for DHcode.org

Basic terms about computing

algorithm
A general step by step process for solving a problem.

bug
An error in a program.

close
When you are done with a file, you should close it.

constant
Fixed values, either numbers, letters or strings, that do not change.

debugging
The process of finding and removing any of the three kinds of programming errors.

documentation
A place where you can go to get detailed information about the programming language.

execute
The process of performing a series of instructions written in the program code. Also called "run."

iteration
A basic building block for algorithms (programs). It allows steps to be repeated.

open
You must open a file before you can read its contents.

parse
To examine a program and analyze the syntax or the formatted structure of the code.

pattern recognition

A way of identifying sequences, similarities and differences in data that helps to establish new ways of interpreting information.

problem solving
The process of formulating a problem, evaluating different options, and expressing a solution.

program
A sequence of instructions that directs certain actions and computations to be performed by a computer.

read
A process by which a program will read the entire contents of a file as a single string. This is often used as a way for Python to know and reference the data inside a spreadsheet.

reserved words
Keywords that are used by Python to initiate some action. Reserved words should not be used as variables or other identifiers (classes, functions, etc.). Reserved words include: *and*, *as*, *assert*, *break*, *class*, *continue*, *def*, *del*, *elif*, *else*, *except*, *False*, *finally*, *for*, *from*, *global*, *if*, *import*, *in*, *is*, *lambda*, *None*, *nonlocal*, *not*, *or*, *pass*, *raise*, *return*, *True*, *try*, *while*, *with*, *yield.*

sequence
Any of the data types that consist of an ordered collection of elements, with each element identified by an index.

syntax
The set of rules that determine how the code is formatted that creates the structure of a program.

write
The process of adding code to a programming file.

Python commands we'll be learning

assignment statement
The process by which the computer assigns a value to the variable. To the left of the assignment token (= sign) is a name. To the right is the value, an expression which is evaluated by the Python interpreter and then assigned to the variable name.

```
var = some value that is assigned by the code, an input, etc.

var1 = “Wendy”
var2 = 12345
```

assignment token
The equal sign (=) is Python’s assignment token, which should not be confused with the mathematical comparison operator using the same symbol.

block
A group of consecutive statements with the same indentation.

body
The statements inside a loop.

boolean expression
A logical statement that only can have a value of either True or False. The Boolean expression is the reply to the Boolean function.

boolean function
A function is the snippet of code that queries, or asks a question of some data. It can only return a value of True or False.

```
name = input(“What is your name?  ”)
What is your name? Wendy
[computer searches database for name “Wendy”]
“Wendy” ← True this name is in the database
    print(“Hello, Wendy”)
“Wendy” ← False this name is not in the database
    print(“Sorry, I don’t know you.”)
```

comment
Information in a program that is meant for other programmers (or anyone reading the source code) and has no effect on the execution of the program.

```
# this is a comment
```

comparison operator
One of the operators that compares two values:

`==`	`equals`		`!=`	`not equal to`
`>`	`greater than`		`<`	`less than`
`>=`	`greater than or equal to`		`<=`	`less than or equal to`

condition
A condition is a trigger to execute code that depends on whether a Boolean expression is True or False.

conditional statement
A statement that controls the flow of execution depending on some condition. In Python the keywords if, elif, and else are used for conditional statements.

```
name = input(“What is your name?  ”)
password = input(“What is your password? ”)
if name == “Wendy” and password == “skobuffs” ← this is True
    print(“Hello, Wendy”)
elif name == “Wendy” and password == “cat345” ← this is False
else:
    print(“Sorry, your password is wrong. Try again”)
```

control flow
Different statement types, such as conditionals (if, elif and else), loops (while), etc., that direct Python to ignore the top-to-bottom sequence of code and react to True/False responses from inputs, data queries, etc.

counter
A variable used to count something, that usually starts at zero and increases incrementally in the body of a loop.

data structure
An organization of data for the purpose of making it easier to use.

A way to store and organize data to make it easier to use. In Python, data can be structured as a list, dictionary, tuple or set.

data type
A data type is a way to classify certain kinds of data and how it can be used in expressions and statements. Strings, integers and floating-point numbers are examples of data types.

dictionary
A dictionary is a collection of key-value pairs that maps together data types. A key can be thought of as a category while the values are the different ways to describe, count or respond to that category. The key is immutable which means it must have a unique name that is different than any other variable name in your code. The values can be any data type, e.g., string, integer or Boolean. It is often abbreviated as *dict*

```
dict = {key: value}

# This dictionary lists the total number of items on a menu
menu_dict = {“sandwiches”: 6, “wraps”: 4, “soup”: 1}

# This dictionary lists the elements of an Instagram user
account
user_dict = {“name”: “Wendy”, “active”: False, “followers”: 2}
```

element
One of the values in a list (or other sequence).

escape
The backslash character \ is used by Python to designate a nonprintable character.

```
# Add a escape to tell Python to ignore the apostrophe
name = ‘Wendy\’s dog’
print(name)
Wendy’s dog
```

expression
An expression is a representation of a value that can be stored for use later. An expression is a combination of values, variables, operators, and calls to functions. This is different than a statement because the expression is simply a value while a statement is an instruction to Python to do something.

```
#The var1 expression is the sum of the two numbers
var1 =  12 + 34
var 1 = 46

#The var2 expression is the number of letters that are in the
word “hello”
var2 = len(“hello”)
var2 = 5
```

float
A Python data type which stores *floating-point* numbers. Floating-point numbers are stored internally in two parts: a *base* and an *exponent*. When printed in the standard format, they look like decimal numbers. In Python, floats round down, so 1.25 would round to 1.2 not 1.3, as expected.

```
float = 5.0
float = -1.25
float = 587643291.001
```

for loop
A statement in Python that is useful for working through a list of items. The loop repeats a statement over and over until it has *iterated* through or touched all the possible items.

```
menu = [“wraps”, “sandwiches”, “soup”, “salad”]
for i in menu:
    print(i)
```

index
An integer variable or value that indicates each element (including blank spaces) of a string or a list. Indexing allows you to manipulate the data more easily, e.g,. find the second to last value in a long list. Remember that counting always starts at 0.

```
# A string index of a name
```

w	e	n	d	y		n	o	r	r	i	s

0	1	2	3	4	5	6	7	8	9	10	11

```
# A list index of menu items
```

wrap	salad	sandwich	soup
0	1	2	3

infinite loop
A loop in which the terminating condition is never satisfied.

int
A Python data type that holds positive and negative **whole** numbers.

```
int = 5
int = -1
int = 587643291
```

input prompt string
Provides the user with hints about the type of value to enter for the variable called *name*.

```
name = input("What's your name?  ")
```

key
A data item that is *mapped to* a value in a dictionary. Keys are used to look up values in a dictionary.

key-value pair
One of the pairs of items in a dictionary. Values are looked up in a dictionary by key.

```
dict = {key: value}
dict = {"sandwiches": 6} ← the number of sandwiches on the menu
dict = {"sandwiches" : "vegetarian"} ← type of sandwich by ingredients
```

keyword
A reserved word that is used by a program to parse (understand) the code; you cannot use keywords like *if*, *def*, *input*, and *while* as variable names or key names.

len
This is an abbreviation of the word "length" and tells Python to count some distance. It could be how many letters are contained in a word. Or the number of items in an object.

```
# how many letters are in the word "hello"?
var1 = len("hello")
var1 = 5

# how many items are in this list of greeting words?
var2 = ["hi", "hello", "ola"]
var2 = 3
```

list
A data structure that is a collection of objects (items).

```
menu_items = ["wrap", "salad", "sandwich", "soup"]
```

logical operator
One of the operators that combines boolean expressions: *and*, *or*, and *not*.

loop
A statement or group of statements that execute repeatedly until a terminating condition is satisfied.

nested list
A list that is an element of another list.

nested loop
A loop inside the body of another loop.

operator
A special symbol that represents a simple computation like addition (+), multiplication (*), subtraction (-), division (/) or string concatenation (+).

parameter
A parameter is a special type of variable used to pass data to a function. There can be multiple parameters inside a function.

print function
A function used in the code that causes Python to display a command.

```
print(“Hello!  ” + name. “How are you?”)
Hello! Wendy. How are you?
```

range
A built-in function in Python that generates a sequence of integers. It is especially useful when we need to write a for loop that executes a fixed number of times.

reassignment
Assigning different values to the same variable during the execution of a program.

return value
The value provided as the result of a function call.

```
def greet():
    return “Hello”
print(greet(), “Wendy”)
Hello Wendy
```

slice
A part of a string (substring).

statement
An instruction that the Python interpreter can execute.

tuple
A data type that contains a sequence of items of any type, like a list, but the contents cannot be changed.

```
# a list index of menu items ranked by daily special
```

wrap	salad	sandwich	soup

0	1	2	3

```
# a **tuple** index of menu items ranked in fixed order
```

salad	sandwich	soup	wrap
0	1	2	3

type conversion function
A function that can convert a data value from one type to another.

```
# Change the input to an integer

age = (int)input("How old are you now?  " )
How old are you now? 25
print("How old will you be next year?)
print(age + 1)
```

value
A string (series of characters) or an integer (number) that can be stored in a variable.

```
var = "value "
var = "thank you"
var = 12345
```

variable
A variable is a label that Python stores in memory to retrieve and use later. Here, a variable must always begin with a letter and should be written in lowercase. Multiple words should be separated by an underscore. In best programming practice, variable names should be chosen so that they describe their use in the program, making the program *self documenting*. Also see: Reserved words.

```
var = "value "
name = "wendy"
lucky_number = 7
best_pet = "porcupine" ← example of bad variable name that
```

```
                                   doesn’t match the intent of the value
```

while loop (sometimes referred to as indefinite iteration)
A loop where we keep going until some TRUE or FALSE condition is met. While loops and Booleans go together.

```
my_lucky_number = 7

guess = int(input(“Guess my lucky number! I think it is: ”))

while my_lucky_number != guess:
    guess = int(input(“Oops! Not it. Try again: ”))

print(“Congrats! You guessed it.”)
```

whitespace
Any of the characters that move the cursor without printing visible characters.