

Lecture 22

408/508 *Computational Techniques for Linguists*

Today's Topics

- Homework 9 okay?
 - That was installing Python 3
- Homework 10 today:
 - install `nltk` (natural language toolkit)
 - see `nltk.org` install

Today's Topics

- More on Python:
 - range()
 - example of use in calculating compound interest
 - print(' *format string* '.format())
 - formatted printing
 - type coercion:
 - converting numbers and strings
 - def
 - defining functions
 - strings and lists
 - sys.argv
 - command line argument vector (ARGV)

Python: range()

• range(): equivalent to producing a list of numbers (*sequence*)

• range(n)

[0, 1, .., n-1] ← note: excludes n!

• range(start, n)

[start, start+1, .., n-1]

• range(start, n, step)

[start, start+step, ..., last]

• last: $start + k * step < n$

• range(n, stop, -step)

counts down to stop

range(Start, End, Step) (*integers only!*)

```
>>> range(1,10,2)
```

```
range(1, 10, 2)
```

```
>>> list(range(1,10,2))
```

```
[1, 3, 5, 7, 9]
```

note: list() converts sequence into a list

Python: range()

```
python
>>> for odd in range(1,9,2):
...     print(odd*odd)
...
1
9
25
49
>>> for odd in range(9,1,2):
...     print(odd*odd)
...
81
49
25
9
>>> for odd in range(9,1,-2):
...     print(odd*odd)
...
81
49
25
9
```

← note: hit ENTER

← note: 81 not printed!

← note: 1 not printed!

Python Program: using range()

- File: futval.py

```
1 print("This program calculates the future value of a 10 year investment.")  
2 principal = float(input("Enter initial principal: "))  
3 apr = float(input("Enter annual interest rate, e.g. 0.03 (3%): "))  
4  
5 for year in range(10):  
6     principal = principal * (1 + apr)  
7  
8 print("Value in 10 years is: ", principal)
```

float() converts string to a floating point number

```
(base) ling508-22$ python futval.py  
This program calculates the future value of a 10 year investment.  
Enter initial principal: 1000  
Enter annual interest rate, e.g. 0.03 (3%): 0.05  
Value in 10 years is: 1628.8946267774422  
(base) ling508-22$
```

can use int() to convert to dollars. How about to 2 decimal places?

Python numbers

- explicit type coercion:

1. `float()`

2. `int()`

3. `long()`

- 64 bit integer: *not in Python 3*

4. `complex(real, imaginary)`

- a *complex number out of two floating point numbers*

5. `complex(string)`

- e.g. `complex('0+1j')`

6. `str(number)`

Formatted Output

- Lots of ways: (*old way in Python*)

```
>>> print "x is %.2f" % 5  
x is 5.00
```

```
>>> print "x is %.2f and %2d" % (5,5)  
x is 5.00 and 5
```

Notation comes originally from C's printf function

- printf also is a Bash shell command

http://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm



Conversion	Meaning	Notes
'd'	Signed integer decimal.	
'i'	Signed integer decimal.	
'o'	Signed octal value.	(1)
'u'	Obsolete type – it is identical to 'd'.	(7)
'x'	Signed hexadecimal (lowercase).	(2)
'X'	Signed hexadecimal (uppercase).	(2)
'e'	Floating point exponential format (lowercase).	(3)
'E'	Floating point exponential format (uppercase).	(3)
'f'	Floating point decimal format.	(3)
'F'	Floating point decimal format.	(3)
'g'	Floating point format. Uses lowercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.	(4)
'G'	Floating point format. Uses uppercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.	(4)
'c'	Single character (accepts integer or single character string).	
'r'	String (converts any Python object using <code>repr()</code>).	(5)
's'	String (converts any Python object using <code>str()</code>).	(6)
'%'	No argument is converted, results in a '%' character in the result.	

Formatted Output

```
<template-string> % (<values>)
```

- **%<width>.<precision><type>**
- **<type>** = d, f, s
- **<width>** = minimum number of characters; right-justified by default, -width => left-justified 0 = as wide as needed
- **<precision>** = number of places after decimal point

- e.g. **02d** two digits wide, pad with 0 if needed

Formatted Output

- **Newer way:**

- <https://docs.python.org/2/tutorial/inputoutput.html#fancier-output-formatting>
- Use {} for each argument
 - (can be numbered, e.g. {0}, {1},... or referenced by keyword {x})

```
>>> import math
>>> print 'The value of PI is approximately {0:.3f}.'.format(math.pi)
The value of PI is approximately 3.142.
```

```
>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 7678}
>>> for name, phone in table.items():
...     print '{0:10} ==> {1:10d}'.format(name, phone)
...
Jack          ==>      4098
Dcab          ==>      7678
Sjoerd        ==>      4127
```

{:width}

Note: use parentheses () for print in Python3

Formatted Output

- Let's modify futval.py:

```
8print("Value in 10 years is: {:.2f}".format(principal))
```

```
$ python futval2.py
```

```
This program calculates the future value of a 10 year investment.
```

```
Enter initial principal: 1000
```

```
Enter annual interest rate, e.g. 0.03 (3%): 0.05
```

```
Value in 10 years is: 1628.89
```

```
$
```

function my_range

- def defines a function
- my_range uses yield to compute a general range on the fly.

```
>>> def my_range(start, end, step):
```

```
...     while start <= end:
```

```
...         yield start
```

```
...         start += step
```

```
... 
```

```
>>> my_range(0,9.9,0.3)
```

```
<generator object my_range at 0x103533580>
```

```
>>> list(my_range(0,9.9,0.3))
```

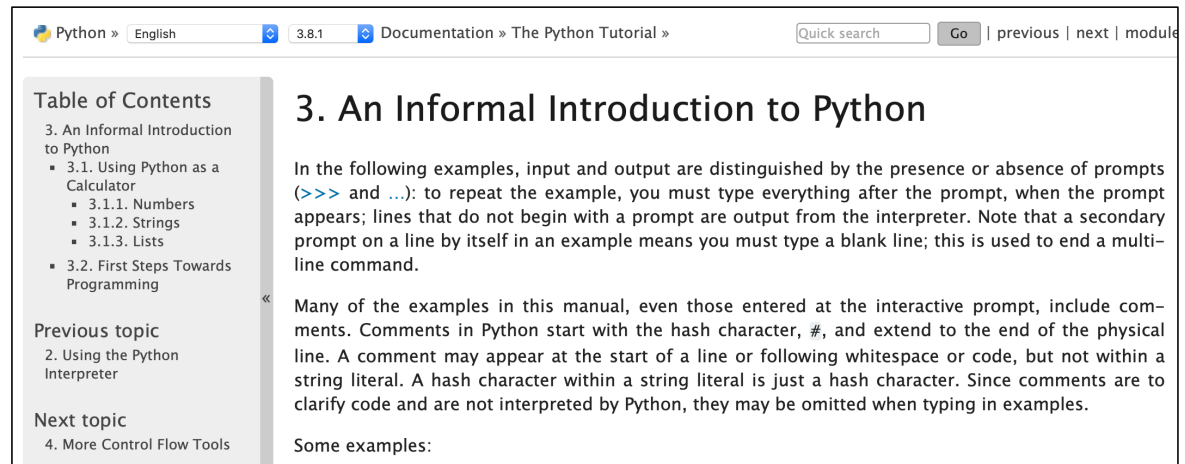
```
[0, 0.3, 0.6, 0.8999999999999999, 1.2, 1.5, 1.8, 2.1, 2.4,  
2.6999999999999997, 2.9999999999999996, 3.2999999999999994,  
3.5999999999999999, 3.8999999999999999, 4.1999999999999999,  
4.4999999999999999, 4.7999999999999999, 5.0999999999999999,  
5.3999999999999999, 5.6999999999999998, 5.9999999999999998,  
6.2999999999999998, 6.5999999999999998, 6.8999999999999998,  
7.1999999999999997, 7.4999999999999997, 7.7999999999999997,  
8.0999999999999998, 8.3999999999999999, 8.7, 9.0, 9.3,  
9.6000000000000001]
```

use TAB key to indent

hit RETURN here

Python

- <https://docs.python.org/3/tutorial/introduction.html>
- Numbers
- **Strings**
- Lists
- Dictionaries



The screenshot shows a web browser window displaying the Python 3.8.1 documentation page for '3. An Informal Introduction to Python'. The browser's address bar shows the URL 'https://docs.python.org/3/tutorial/introduction.html'. The page title is 'Python » English » 3.8.1 » Documentation » The Python Tutorial ». The page content is divided into a left sidebar and a main content area. The sidebar contains a 'Table of Contents' with the following items: '3. An Informal Introduction to Python', '3.1. Using Python as a Calculator' (with sub-items '3.1.1. Numbers', '3.1.2. Strings', and '3.1.3. Lists'), and '3.2. First Steps Towards Programming'. Below the table of contents are links for 'Previous topic' (2. Using the Python Interpreter) and 'Next topic' (4. More Control Flow Tools). The main content area features the heading '3. An Informal Introduction to Python' followed by a paragraph explaining that input and output are distinguished by the presence or absence of prompts (>>> and ...). It notes that lines not beginning with a prompt are output from the interpreter, and a secondary prompt on a line by itself indicates a multi-line command. A second paragraph explains that many examples include comments, which start with the hash character (#) and extend to the end of the physical line. It states that comments 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. The text concludes with 'Some examples:'.

Python: Strings

3.1.2. Strings ¶

Besides numbers, Python can also manipulate strings, which can be expressed in several ways. They can be enclosed in single quotes ('...') or double quotes ("...") with the same result [2]. \ can be used to escape quotes:

```
>>> 'spam eggs' # single quotes
'spam eggs'
>>> 'doesn\'t' # use \' to escape the single quote...
"doesn't"
>>> "doesn't" # ...or use double quotes instead
"doesn't"
>>> '"Yes," they said.'
'"Yes," they said.'
>>> "\"Yes,\" they said."
'"Yes," they said.'
>>> 'Isn\'t," they said.'
'Isn\'t," they said.'
```

In the interactive interpreter, the output string is enclosed in quotes and special characters are escaped with backslashes. While this might sometimes look different from the input (the enclosing

String slicing

- String is like an array of characters (strings):
 - **str[i]** index i (from 0 to len(str)-1)
 - **str[-i]** index i from the end (1 = last)
 - **str[i:j]** slice from index i until index j-1
 - **str[:j]** slice from index 0 until index j-1
 - **str[i:]** slice from index i until end of the string

Operator	Meaning
+	Concatenation
*	Repetition
<string>[]	Indexing
<string>[:]	Slicing
len(<string>)	Length
for <var> in <string>	Iteration through characters

Table 4.1: Python string operations.

List vs. Strings

- Although Strings are like Lists, Lists are **mutable**, Strings are not.

```
>>> myList = [34, 26, 15, 10]
>>> myList[2]
15
>>> myList[2] = 0
>>> myList
[34, 26, 0, 10]
>>> myString = "Hello World"
>>> myString[2]
'1'
>>> myString[2] = 'z'
Traceback (innermost last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
```

← changed!

← not mutable!

Python `sys.argv`

- List of arguments from the command line: what's `argv[0]` then?
- can use `len()` to calculate number of arguments

```
1 from sys import argv
2 print(argv[0])
```

test.py

```
$ python3 test.py
test.py
$ python3 test.py 121 2
test.py
```

argv[1] and argv[2]
ignored

Formatted Output

Let's modify futval.py to accept arguments on the command line or via prompts:

```
$ python futval3.py 1000 0.05
```

```
This program calculates the future value of a 10 year investment.
```

```
Value in 10 years is: 1628.89
```

```
$ python futval3.py 1000
```

```
This program calculates the future value of a 10 year investment.
```

```
Enter annual interest rate: 0.06
```

```
Value in 10 years is: 1790.85
```

```
$ python futval3.py
```

```
This program calculates the future value of a 10 year investment.
```

```
Enter initial principal: 1000
```

```
Enter annual interest rate: 0.07
```

```
Value in 10 years is: 1967.15
```

Homework 10

- No need to submit
- But if you run into trouble, email me!

<https://www.nltk.org/install.html>

nltk.org

Installing NLTK

NLTK requires Python versions 3.7, 3.8, 3.9, 3.10 or 3.11.

For Windows users, it is strongly recommended that you go through this guide to install Python 3 successfully <https://docs.python-guide.org/starting/install3/win/#install3-windows>

Setting up a Python Environment (Mac/Unix/Windows)

Please go through this [guide](https://docs.python-guide.org/dev/virtualenvs/) to learn how to manage your virtual environment managers before you install NLTK, <https://docs.python-guide.org/dev/virtualenvs/>

Alternatively, you can use the Anaconda distribution installer that comes “batteries included” <https://www.anaconda.com/distribution/>

Mac/Unix

1. Install NLTK: run `pip install --user -U nltk`
2. Install Numpy (optional): run `pip install --user -U numpy`
3. Test installation: run `python` then type `import nltk`

Get-Command

```
Windows PowerShell x Windows PowerShell x + v
PS C:\Users\sandiway> Get-Command python

CommandType      Name                Version      Source
-----
Application      python.exe          0.0.0.0     C:\Users\sandiway\AppData\Local\Microsoft\WindowsApps\python.exe

PS C:\Users\sandiway> Get-Command pip

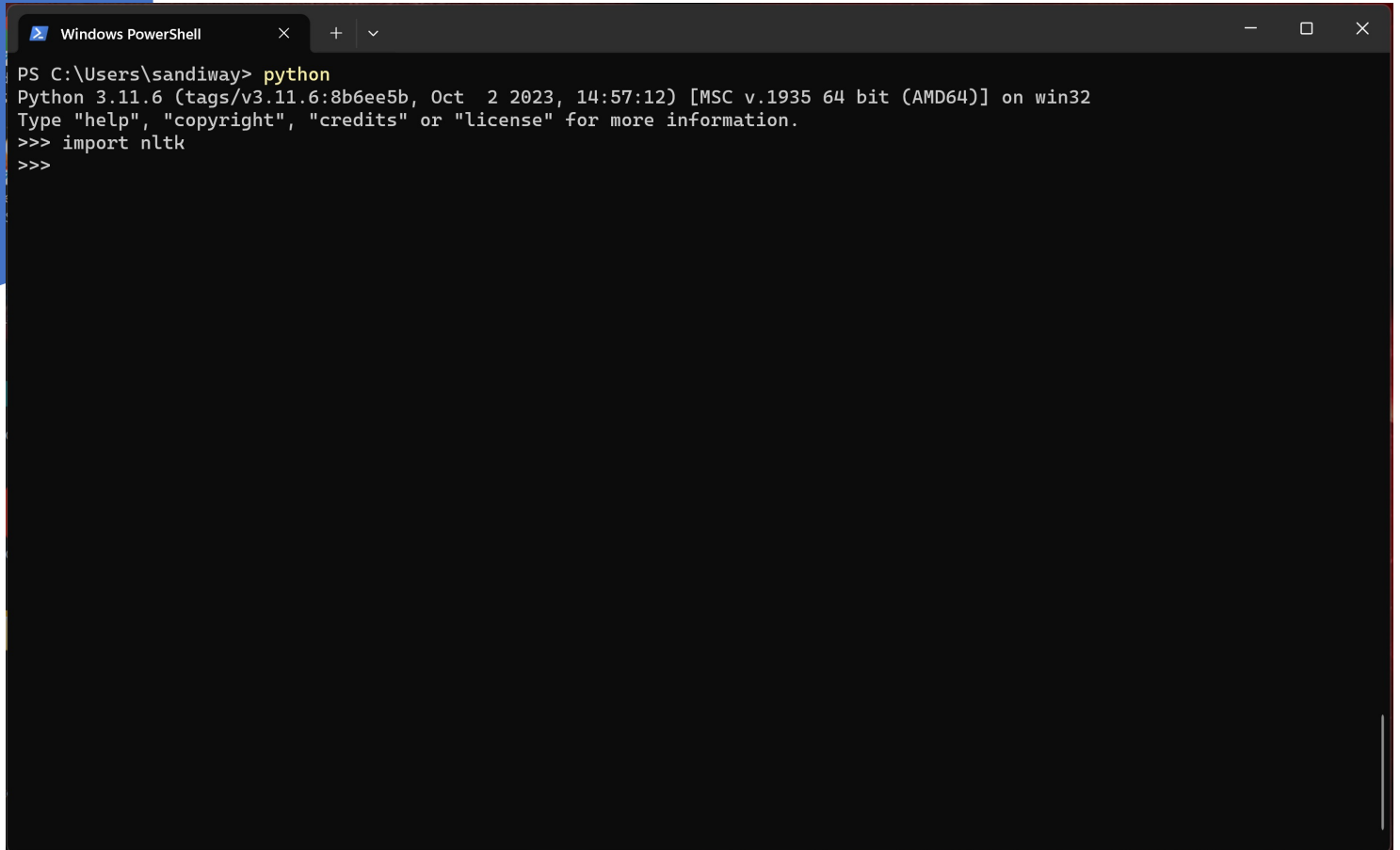
CommandType      Name                Version      Source
-----
Application      pip.exe             0.0.0.0     C:\Users\sandiway\AppData\Local\Microsoft\WindowsApps\pip.exe

PS C:\Users\sandiway>
```

pip on Windows

```
Windows PowerShell
PS C:\Users\sandiway> pip install nltk
Collecting nltk
  Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 4.4 MB/s eta 0:00:00
Collecting click (from nltk)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting joblib (from nltk)
  Downloading joblib-1.3.2-py3-none-any.whl.metadata (5.4 kB)
Collecting regex>=2021.8.3 (from nltk)
  Downloading regex-2023.10.3-cp311-cp311-win_amd64.whl.metadata (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.0/42.0 kB 2.0 MB/s eta 0:00:00
Collecting tqdm (from nltk)
  Downloading tqdm-4.66.1-py3-none-any.whl.metadata (57 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.6/57.6 kB 3.2 MB/s eta 0:00:00
Collecting colorama (from click->nltk)
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
  Downloading regex-2023.10.3-cp311-cp311-win_amd64.whl (269 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 269.6/269.6 kB 5.5 MB/s eta 0:00:00
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.9/97.9 kB 5.8 MB/s eta 0:00:00
  Downloading joblib-1.3.2-py3-none-any.whl (302 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 302.2/302.2 kB 6.2 MB/s eta 0:00:00
  Downloading tqdm-4.66.1-py3-none-any.whl (78 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.3/78.3 kB ? eta 0:00:00
Installing collected packages: regex, joblib, colorama, tqdm, click, nltk
  WARNING: The script tqdm.exe is installed in 'C:\Users\sandiway\AppData\Local\Packages\PythonSoftwareFoundation.Python
.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The script nltk.exe is installed in 'C:\Users\sandiway\AppData\Local\Packages\PythonSoftwareFoundation.Python
.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed click-8.1.7 colorama-0.4.6 joblib-1.3.2 nltk-3.8.1 regex-2023.10.3 tqdm-4.66.1
PS C:\Users\sandiway>
```

nltk on Windows



```
Windows PowerShell
PS C:\Users\sandiway> python
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>>
```

install
nltk
data
(select all)

The image shows a Windows PowerShell terminal window on the left and the NLTK Downloader application window on the right. The PowerShell window shows the execution of the `Get-Command` command to find the Python and pip executables, and the execution of `python` to start the NLTK downloader. The NLTK Downloader window shows a list of collections to download, with 'all' selected. The status of each collection is shown as 'partial' or 'not installed'. The application is currently downloading the 'cess_cat' package, as indicated by the progress bar at the bottom.

```
PS C:\Users\sandaway> Get-Command python.exe
CommandType Name
-----
Application python.exe

PS C:\Users\sandaway> Get-Command pip.exe
CommandType Name
-----
Application pip.exe

PS C:\Users\sandaway> python
Python 3.11.6 (tags/v3.11.6)
Type "help", "copyright", "credits() or "license()" for more
>>> import nltk
>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.html
```

Identifier	Name	Size	Status
all	All packages	n/a	partial
all-corpora	All the corpora	n/a	partial
all-nltk	All packages available on nltk_data gh-pages branch	n/a	partial
book	Everything used in the NLTK Book	n/a	partial
popular	Popular packages	n/a	partial
tests	Packages for running tests	n/a	partial
third-party	Third-party data packages	n/a	not installed

Server Index: https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.html
Download Directory: C:\Users\sandaway\AppData\Roaming\nltk_data
Downloading package 'cess_cat'

nlTK data
installed

The image shows a Windows PowerShell terminal window and the NLTK Downloader application. The PowerShell window displays the following commands and output:

```
PS C:\Users\sandiway> Get-Command python

CommandType      Name
-----
Application      python.exe

PS C:\Users\sandiway> Get-Command pip

CommandType      Name
-----
Application      pip.exe

PS C:\Users\sandiway> python
Python 3.11.6 (tags/v3.11.6:8b6ee5b, C
Type "help", "copyright", "credits" or
>>> import nltk
>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data
```

The NLTK Downloader application window shows a table of available collections:

Identifier	Name	Size	Status
all	All packages	n/a	installed
all-corpora	All the corpora	n/a	installed
all-nltk	All packages available on nltk_data gh-pages bran	n/a	installed
book	Everything used in the NLTK Book	n/a	installed
popular	Popular packages	n/a	installed
tests	Packages for running tests	n/a	installed
third-party	Third-party data packages	n/a	installed

At the bottom of the NLTK Downloader window, the following information is displayed:

Server Index: https://raw.githubusercontent.com/nltk/nltk_data
Download Directory: C:\Users\sandiway\AppData\Roaming\nltk_data
Finished downloading collection 'all'.

The screenshot shows a Windows desktop environment with a WSL terminal window and an NLTK Downloader application window. The terminal window displays the following commands and output:

```
sandiway@DESKTOP-VEPP64:~$ pip install nltk
Requirement already satisfied: nltk in /usr/local/lib/python3.8/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.8/dist-packages (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.8/dist-packages (1.3.2)
Requirement already satisfied: regex in /usr/local/lib/python3.8/dist-packages (2023.10.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (4.66.1)

[notice] A new release of pip is available:
[notice] To update, run: python3.8 -m pip install --upgrade pip

sandiway@DESKTOP-VEPP64:~$ python3.8 -m nltk.downloader
Python 3.8.10 (default, May 26 2023) on linux
[GCC 9.4.0] on linux
Type "help", "copyright", "credits() or "license()" for more
>>> import nltk
>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

The NLTK Downloader application window shows a table of available collections:

Identifier	Name	Size	Status
all	All packages	n/a	installed
all-corpora	All the corpora	n/a	installed
all-nltk	All packages available on nltk_data gh-pages branch	n/a	installed
book	Everything used in the NLTK Book	n/a	installed
popular	Popular packages	n/a	installed
tests	Packages for running tests	n/a	installed
third-party	Third-party data packages	n/a	installed

The application also shows the following settings:

- Server Index: https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
- Download Directory: /home/sandiway/nltk_data

nltk data in WSL

The Windows taskbar at the bottom of the screen shows the system tray with a temperature of 58°F and a 'Clear' button. The search bar is active, and the taskbar contains several application icons including File Explorer, Edge, and the NLTK Downloader application. The system clock shows the time as 10:51 PM on 10/30/2023.