# Introduction to Python

Joshua Garrison Burkhart, Ph.D.
Postdoctoral Researcher
Bioinformatics Core
Department of Quantitative Health Sciences
University of Hawaii John A. Burns School of Medicine

# Overview

Origin

Release History

Popularity

Getting Started (DIY)

Distinguishing Features

Basic Python Syntax

Available Functions

File I/O

Array Operations

Workshop

References

Questions

Suggested Topics for Self-study

# Origin

First released by Guido van Rossum in 1991 as Python 0.9.0.

Guido is currently working at Microsoft but has worked at DropBox, Google, NIST among other technology and security groups.

Listed as #9 on the Linkedin list of top 10 programmers of all time.

https://www.linkedin.com/pulse/top-10-programmer-world-all-time-kiran-deshmukh-k-d-




https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html

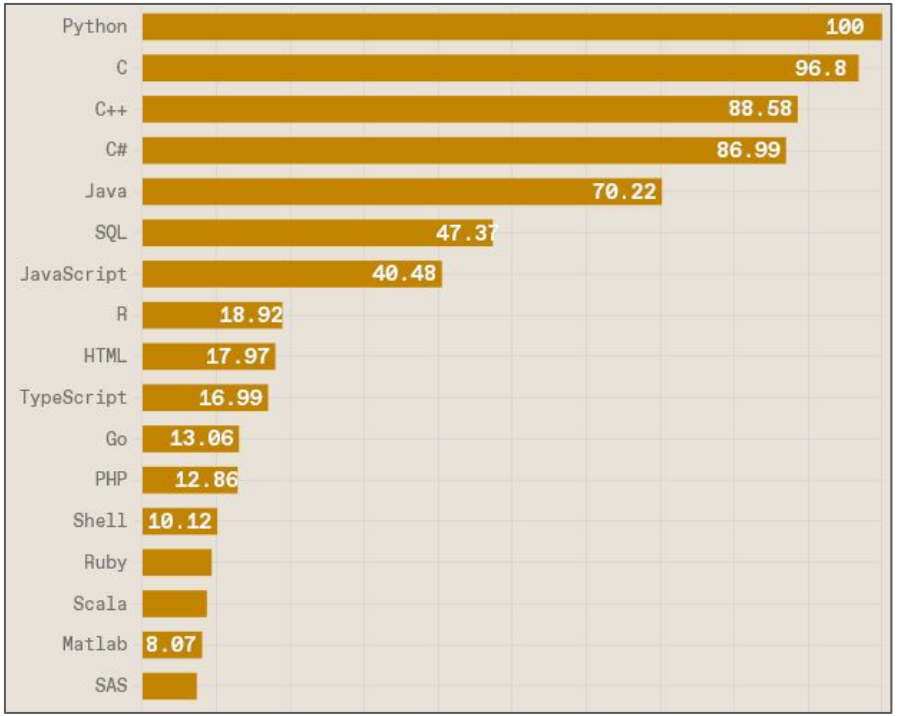
# Release History

| | | | | |
|---|---|---|---|---|
| 2.6 | 2.6.9[27] | 2008-10-01[27] | 2010-08-24[b][27] | 2013-10-29[27] |
| 2.7 | 2.7.18[32] | 2010-07-03[32] | 2020-01-01[c][32] | |
| 3.0 | 3.0.1[44] | 2008-12-03[27] | 2009-06-27[51] | |
| 3.1 | 3.1.5[52] | 2009-06-27[52] | 2011-06-12[53] | 2012-04-06[52] |
| 3.2 | 3.2.6[54] | 2011-02-20[54] | 2013-05-13[b][54] | 2016-02-20[54] |
| 3.3 | 3.3.7[55] | 2012-09-29[55] | 2014-03-08[b][55] | 2017-09-29[55] |
| 3.4 | 3.4.10[56] | 2014-03-16[56] | 2017-08-09[57] | 2019-03-18[a][56] |
| 3.5 | 3.5.10[58] | 2015-09-13[58] | 2017-08-08[59] | 2020-09-30[58] |
| 3.6 | 3.6.15[60] | 2016-12-23[60] | 2018-12-24[b][60] | 2021-12-23[60] |
| 3.7 | *3.7.15*[61] | 2018-06-27[61] | 2020-06-27[b][61] | 2023-06-27[61] |
| 3.8 | *3.8.15*[62] | 2019-10-14[62] | 2021-05-03[b][62] | 2024-10[62] |
| 3.9 | *3.9.15*[63] | 2020-10-05[63] | 2022-05-17[b][63] | 2025-10[63][64] |
| 3.10 | *3.10.8*[65] | 2021-10-04[65] | 2023-05[65] | 2026-10[65] |
| **3.11** | 3.11.0[66] | 2022-10-24[66] | 2024-05[66] | 2027-10[66] |
| 3.12 | [67] | 2023-10[67] | 2025-05[67] | 2028-10[67] |
| **Legend:** | | Old version, Older version, still maintained, **Latest version**, Latest preview version, Future release | | |
| *Italics* indicates the latest micro version of currently supported versions as of 2022-10-13[needs update]. | | | | |

https://en.wikipedia.org/wiki/History_of_Python

# Popularity

A 2022 programming language ranking by the Institute of Electrical and Electronics Engineers (IEEE) found Python was #1 using social media, job postings and online repository results.




https://spectrum.ieee.org/top-programming-languages-2022

# Getting Started (DIY)

Default python versions differ across OS

```
PS C:\Users\jgbur> python --version

Python was not found; run without arguments to install from the
Microsoft Store, or disable this shortcut from Settings > Manage App
Execution Aliases.
```

```
Last login: Thu Jan 1 00:00:00 on ttys000

$ python --version

Python 2.7.10
```

```
(base) jgburk@pop-os:~$ python --version

Python 3.9.10
```

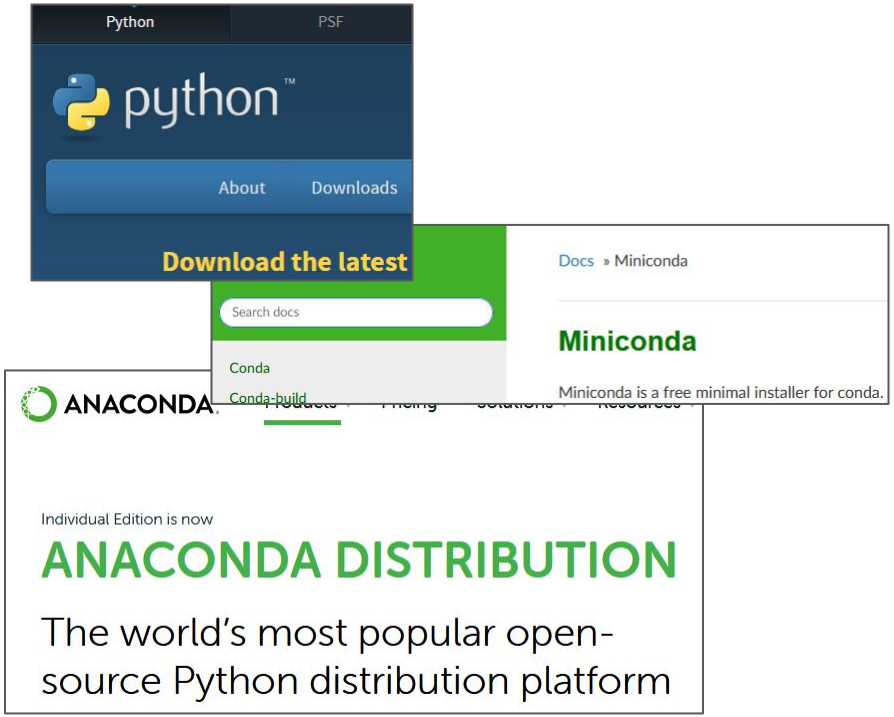
# Getting Started (DIY)

If it's not already installed, install python or use a virtual environment

https://www.python.org/downloads/

https://docs.conda.io/en/latest/miniconda.html

https://www.anaconda.com/products/distribution



# Getting Started (DIY)

## Install an IDE

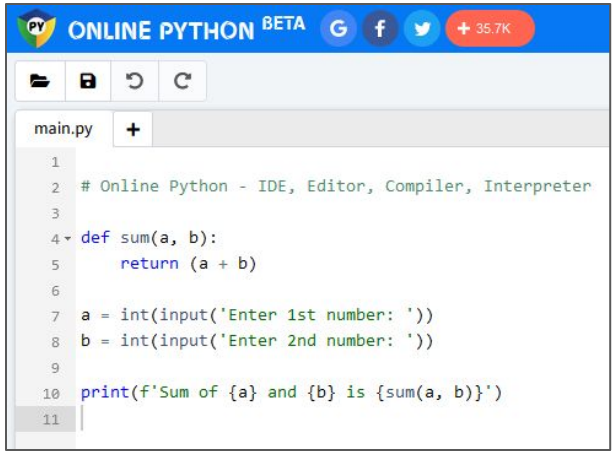
| | | |
|---|---|---|
| IDLE<br>Python Software Foundation License | PyCharm<br>Apache License | Spyder<br>MIT License |
| PyDev<br>Eclipse Public License | Atom<br>MIT License | Thonny<br>MIT License |
| Visual Studio<br>Proprietary software | Wing IDE<br>Proprietary software | eric<br>GNU General Public License |
| Python Tools for Visual Studio<br>Apache License | PyScripter<br>MIT License | Komodo Edit<br>GNU General Public License |
| KDevelop<br>GNU General Public License | PIDA<br>GNU General Public License | Komodo IDE<br>Proprietary software |
| Geany<br>GNU General Public License | Anjuta<br>GNU General Public License | DrPython<br>GNU General Public License |
| Leo<br>MIT License | Cloud9 IDE<br>Freeware | |

# Getting Started (DIY)

Or use an online environment

https://colab.research.google.com/

https://www.online-python.com/



# Distinguishing Features

Python is a general-purpose programming language.

Unlike most programming languages, indentation (white space) matters in Python.

Python includes many built-in functions.

Many domain-specific libraries are available in Python.

# Basic Python Syntax

Keywords in Python are similar to many other programming languages.

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

https://docs.python.org/3/reference/lexical_analysis.html#keywords

# Basic Python Syntax

Truth can be evaluated across types and will be “True” if the object has a nonzero length or specifically returns a “False” value when tested.

Examples of False

```
None

False

0

0.0

“”

[]
```



https://docs.python.org/3/library/stdtypes.html#truth-value-testing

# Basic Python Syntax

Boolean Logic in Python uses keywords “and”, “or” and “not”.

```
A and B
```

True if A is True and B is True

```
A or B
```

True of A is True or B is True

```
not A
```

True if A is False



https://docs.python.org/3/library/stdtypes.html#boolean-operations-and-or-not

# Basic Python Syntax

Values are compared in Python with these operators.

| | |
|---|---|
| `<` | less than |
| `<=` | less than or equal |
| `>` | greater than |
| `>=` | greater than or equal |
| `==` | equal |
| `!=` | not equal |
| `is` | object identity |
| `is not` | negated object identity |



https://docs.python.org/3/library/stdtypes.html#comparisons

# Basic Python Syntax

Numeric Types include `int()` for integers, `float()` for floating point numbers and `complex()` for complex numbers.

These are the numeric operators.

| | |
|---|---|
| `x + y` | sum of *x* and *y* |
| `x - y` | difference of *x* and *y* |
| `x * y` | product of *x* and *y* |
| `x / y` | quotient of *x* and *y* |
| `x // y` | floored quotient of *x* and *y* |
| `x % y` | remainder of x / y |
| `-x` | *x* negated |
| `+x` | *x* unchanged |
| `abs(x)` | absolute value or magnitude of *x* |
| `int(x)` | *x* converted to integer |
| `float(x)` | *x* converted to floating point |
| `complex(re, im)` | a complex number with real part *re*, imaginary part *im*. *im* defaults to zero. |
| `c.conjugate()` | conjugate of the complex number *c* |
| `divmod(x, y)` | the pair (x // y, x % y) |
| `pow(x, y)` | *x* to the power *y* |
| `x ** y` | *x* to the power *y* |

https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex

# Basic Python Syntax

Sequence Types include `list`, `tuple` and `range`.

These are the sequence operators.

| | |
|---|---|
| `x in s` | True if an item of *s* is equal to *x*, else False |
| `x not in s` | False if an item of *s* is equal to *x*, else True |
| `s + t` | the concatenation of *s* and *t* |
| `s * n or n * s` | equivalent to adding *s* to itself *n* times |
| `s[i]` | *i*th item of *s*, origin 0 |
| `s[i:j]` | slice of *s* from *i* to *j* |
| `s[i:j:k]` | slice of *s* from *i* to *j* with step *k* |
| `len(s)` | length of *s* |
| `min(s)` | smallest item of *s* |
| `max(s)` | largest item of *s* |
| `s.index(x[, i[, j]])` | index of the first occurrence of *x* in *s* (at or after index *i* and before index *j*) |
| `s.count(x)` | total number of occurrences of *x* in *s* |

https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range

# Basic Python Syntax

The Text Sequence Type in Python is `str`. Objects of this type are referred to as “strings”.

Examples of strings

```
“A”

“An example string”

‘A single-quoted string’

‘.’

‘A string \

On two lines’
```



https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str

# Basic Python Syntax

Set Types in Python include `set` and `frozenset` for mutable and immutable collections of distinct objects, respectively.

We can try to add the sequence of values “1,2,2,2,2” to set A with:

```
A = set([1,2,2,2,2])
```

However, it will be equal to set B:

```
B = set([1,2])
```



https://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset

# Basic Python Syntax

The Mapping Type in Python is `dict` or “dictionary”. This type “maps” keys to values.

An example of a dictionary.

```
A = dict(key1=1, key2=2)

A[“key3”] = 300

Y = A[“key2”]

Z = A[“key3”]
```



https://docs.python.org/3/library/stdtypes.html#mapping-types-dict

# Basic Python Syntax

Control Flow uses `while` and `for`, along with the keywords `if`, `break`, `continue`, `pass`, `match`, and the `range()` function.

An example of control flow.

```
for x in range(10):
    if x == 8:
        print("8!")
        continue
    print("not 8!")
```



https://docs.python.org/3/tutorial/controlflow.html

# Basic Python Syntax

Functions are specified with the `def` keyword in Python. Encapsulating code in functions organizes files and allows for various forms reuse.

An example of using a function.

```
def square_if_odd(n):
    if n % 2 != 0:
        return n * n
    return n
print(square_if_odd(2))
print(square_if_odd(3))
```



https://docs.python.org/3/tutorial/controlflow.html#defining-functions

# Available Functions

## Built-in Functions

3.11.0 Go

### Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

**Built-in Functions**

| | | | |
|---|---|---|---|
| **A** | **E** | **L** | **R** |
| abs() | enumerate() | len() | range() |
| aiter() | eval() | list() | repr() |
| all() | exec() | locals() | reversed() |
| any() | | | round() |
| anext() | **F** | **M** | |
| ascii() | filter() | map() | **S** |
| | float() | max() | set() |
| **B** | format() | memoryview() | setattr() |
| bin() | frozenset() | min() | slice() |
| bool() | | | sorted() |
| breakpoint() | **G** | **N** | staticmethod() |
| bytearray() | getattr() | next() | str() |
| bytes() | globals() | | sum() |



https://docs.python.org/3/library/functions.html

# File I/O

Reading input from and writing output to files, also referred to as "I/O" uses the `open()` function in Python.

Python variables are stored in random access memory or "RAM" or "memory" and files usually represent data on a storage device such as a "disk" or "drive" and are interacted with differently from other types of variables.

We can open a file and read the first line with:

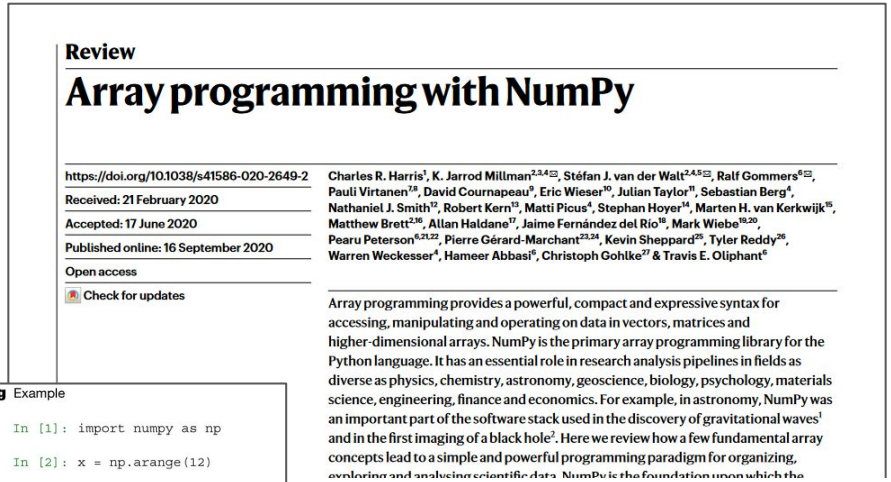
```
file_p = open('example.txt','r')

x = file_p.readline()

file_p.close()
```



https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files

# Array Operations

NumPy is a popular library which supports array programming for Python.



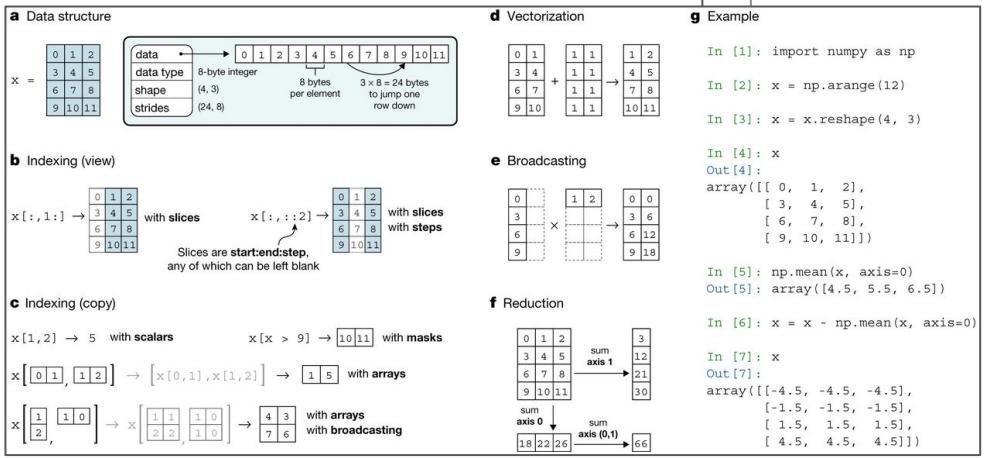
Review

## Array programming with NumPy

https://doi.org/10.1038/s41586-020-2649-2
Received: 21 February 2020
Accepted: 17 June 2020
Published online: 16 September 2020
Open access
Check for updates

Charles R. Harris[1], K. Jarrod Millman[2,3,4], Stéfan J. van der Walt[2,4,5], Ralf Gommers[6], Pauli Virtanen[7,8], David Cournapeau[9], Eric Wieser[10], Julian Taylor[11], Sebastian Berg[4], Nathaniel J. Smith[12], Robert Kern[13], Matti Picus[4], Stephan Hoyer[14], Marten H. van Kerkwijk[15], Matthew Brett[2,16], Allan Haldane[17], Jaime Fernández del Rio[18], Mark Wiebe[19,20], Pearu Peterson[6,21,22], Pierre Gérard-Marchant[23,24], Kevin Sheppard[25], Tyler Reddy[26], Warren Weckesser[4], Hameer Abbasi[6], Christoph Gohlke[27] & Travis E. Oliphant[6]

Array programming provides a powerful, compact and expressive syntax for accessing, manipulating and operating on data in vectors, matrices and higher-dimensional arrays. NumPy is the primary array programming library for the Python language. It has an essential role in research analysis pipelines in fields as diverse as physics, chemistry, astronomy, geoscience, biology, psychology, materials science, engineering, finance and economics. For example, in astronomy, NumPy was an important part of the software stack used in the discovery of gravitational waves[1] and in the first imaging of a black hole[2]. Here we review how a few fundamental array concepts lead to a simple and powerful programming paradigm for organizing, exploring and analysing scientific data. NumPy is the foundation upon which the




Harris CR, Millman KJ, Van Der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R. Array programming with NumPy. Nature. 2020 Sep;585(7825):357-62. https://www.nature.com/articles/s41586-020-2649-2

https://numpy.org/doc/stable/user/basics.html

# Workshop

Notebook 1 (Concepts)
https://colab.research.google.com/drive/1b5o2NFIt8buPxflpT7J6rfMszInkvRmk?usp=sharing

Notebook 2 (Example)
https://colab.research.google.com/drive/1DEid32Tb_TK8WijERCUluUaIGDyhgxdS?usp=sharing

# Workshop

## (U. Hawaii Users Only)

Follow the instructions at

https://www.hawaii.edu/google/extra/

and enable Google@UH Consumer Apps

# Workshop

5 Minute Break

# References


Python Software Foundation

https://www.python.org/about/gettingstarted/

https://docs.python.org/3/tutorial/

Google

https://developers.google.com/edu/python/introduction

MIT OCW

https://ocw.mit.edu/courses/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/

edX

https://www.edx.org/course/introduction-to-computer-science-and-programming-7

PyFormat

https://pyformat.info/

Pandas

https://pandas.pydata.org/docs/user_guide/index.html#user-guide

W3Schools

https://www.w3schools.com/python/python_intro.asp

Coursera

https://www.coursera.org/projects/introduction-to-python

https://www.coursera.org/learn/python-programming-intro

Microsoft

https://learn.microsoft.com/en-us/training/modules/intro-to-python/

Software Carpentry Foundation

https://swcarpentry.github.io/python-novice-inflammation/

Stanford

https://cs.stanford.edu/people/nick/py/

https://colab.research.google.com/github/cs231n/cs231n.github.io/blob/master/python-colab.ipynb

Numpy

https://numpy.org/doc/stable/user/basics.html

Questions?

# Suggested Topics for Self-study

Error Handling

Testing

Modules

Classes

Virtual Environments

Interactive Programs

Web Development

Scientific Programming

Machine Learning