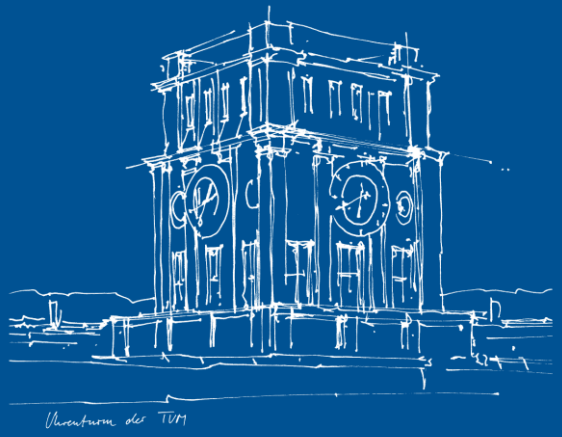
INTRODUCTION TO PYTHON PROGRAMMING FOR DATA ANALYTICS

Part I - Lecture 01

Introduction to Python



Lecturer: Maotong Sun

Technische Universität München

TUM Campus Heilbronn

Reference book:

- *Python Distilled (David Beazley)*
- *Python Crash Course (Eric Matthes)*

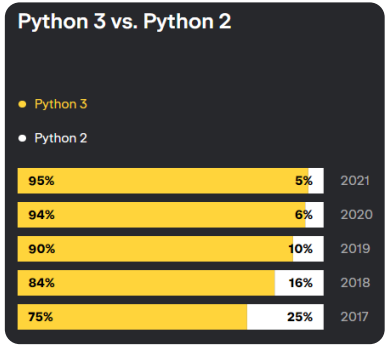
History of Python

- Started in the late 1980s by Guido van Rossum.
- Named after the BBC comedy series “Monty Python's Flying Circus.”
- In 2018, Rossum stepped down as leader (although he remains the “Benevolent Dictator for Life”).
- The five-member Steering Council now leads the project.

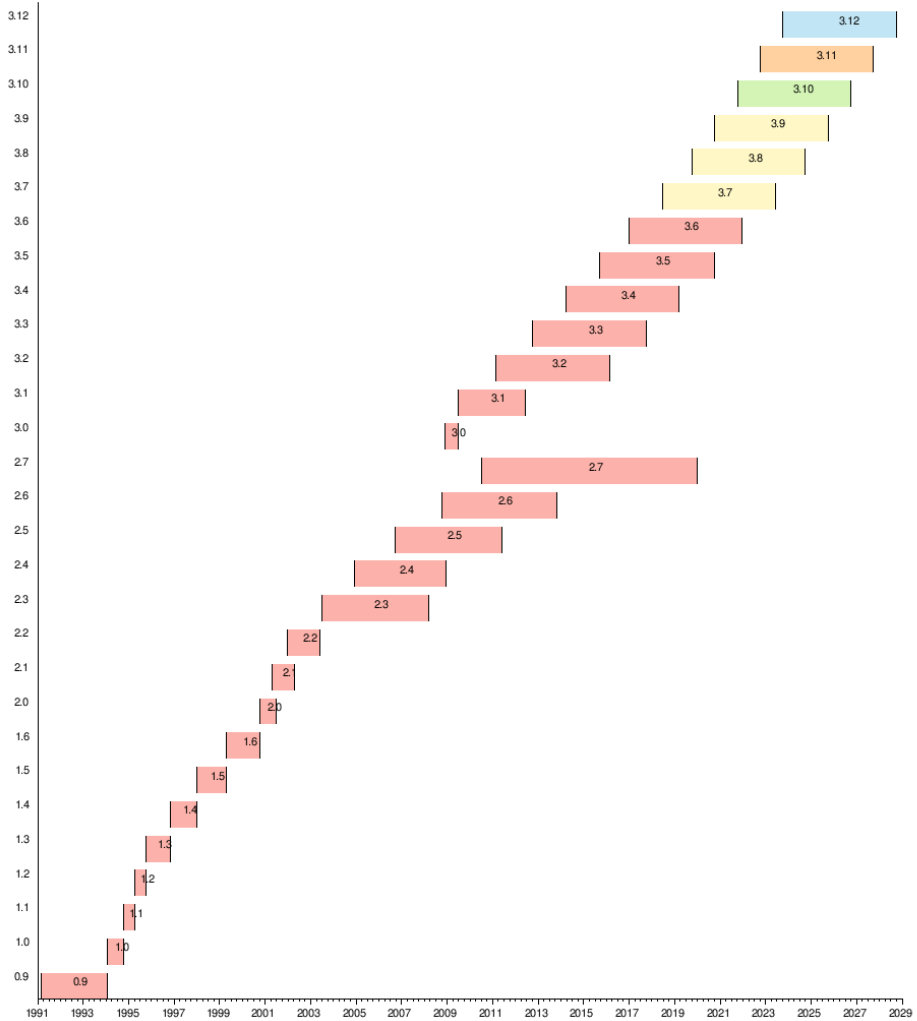
Guido van Rossum

https://gvanrossum.github.io/

History of Python

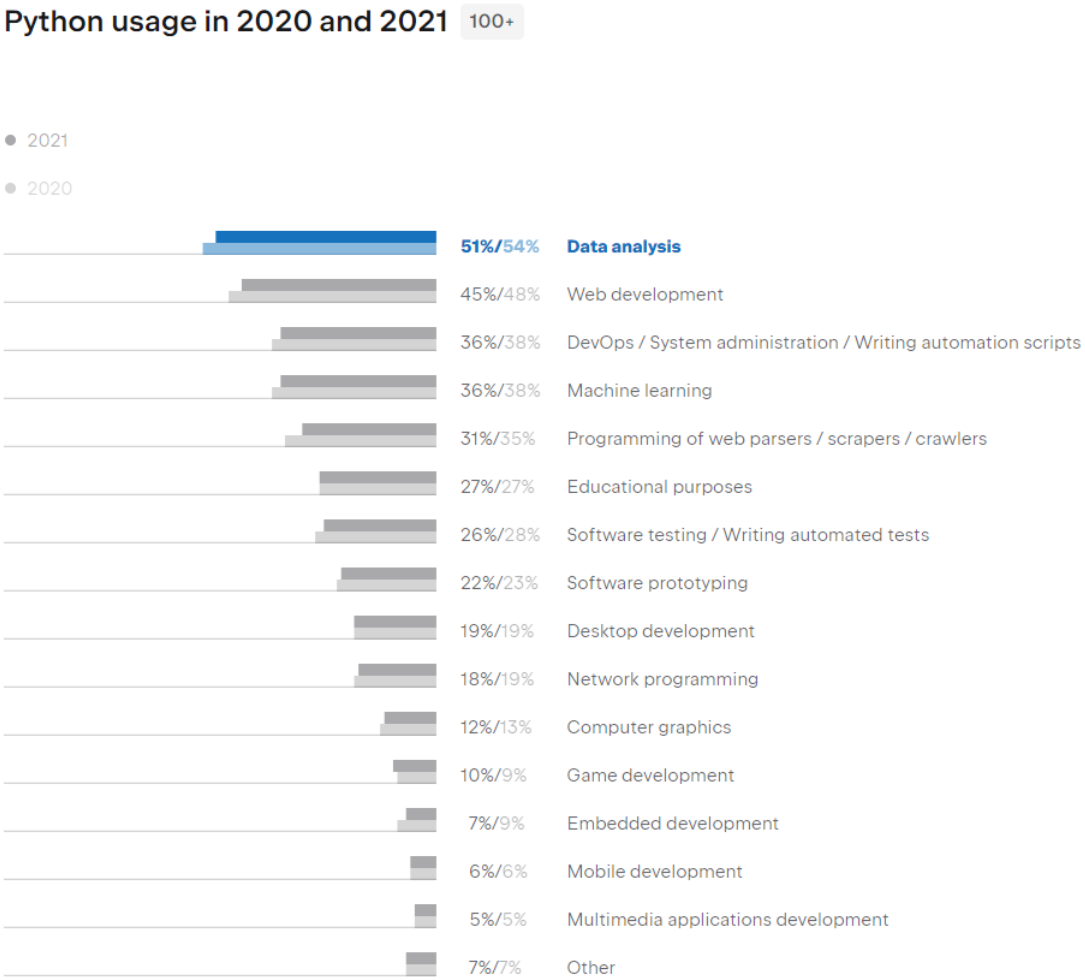


https://lp.jetbrains.com/python-developers-survey-2021



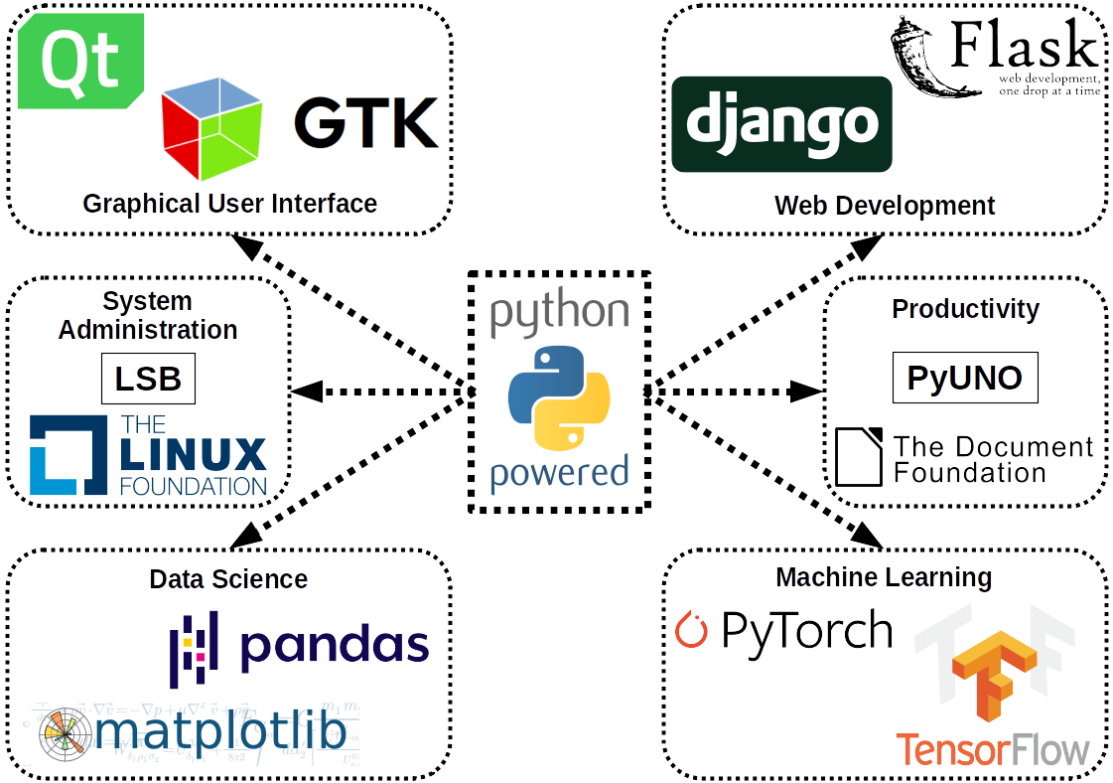
https://en.wikipedia.org/wiki/History_of_Python

Application areas



https://lp.jetbrains.com/python-developers-survey-2021

Frameworks



https://en.wikipedia.org/wiki/Python_(programming_language)#/

Apps built with Python

Instagram

Pinterest

Spotify

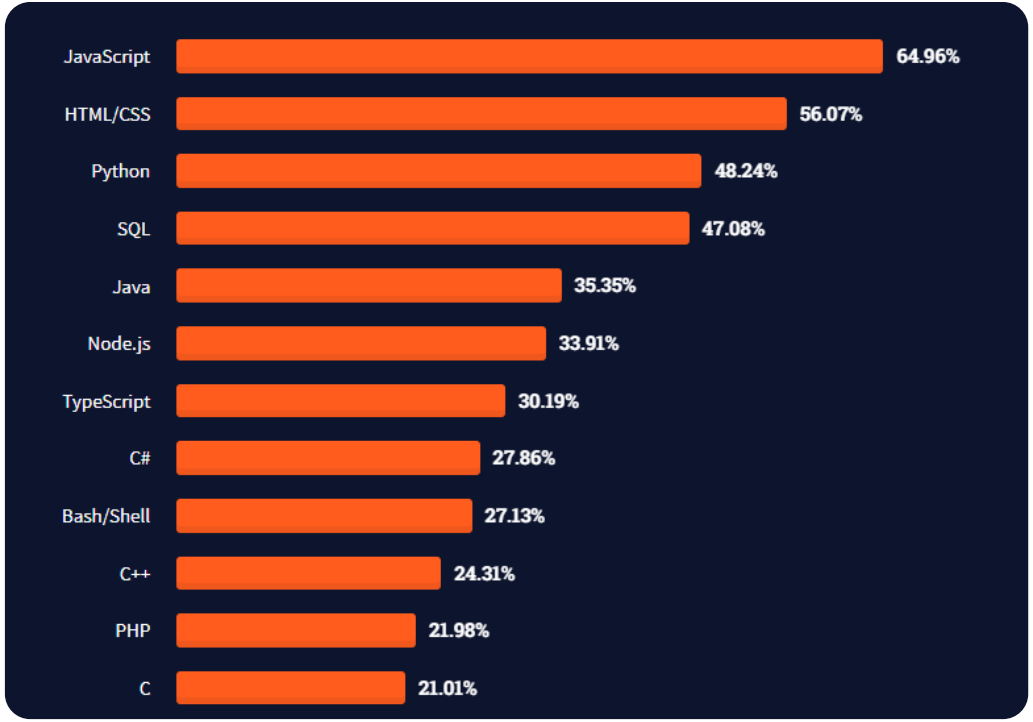
Dropbox

Uber

Reddit

Source: https://appinventiv.com/blog/types-of-apps-developed-using-python/

Popularity



Source: 2021 Stackoverflow Developer Survey

Variables

Definition

- A variable is a name that refers to a value.

```python
message = "Hello Python world!"
print(message)
```

```
Hello Python world!
```

- The value of a variable can be changed at any time in the program.

```python
message = "Hello Python world!"
print(message)
message = "Hello Python Crash Course world!"
print(message)
```

```
Hello Python world!
Hello Python Crash Course world!
```

Variables

Rules and conventions

- **Rules:**
  - Variable names can contain letters, numbers, and underscores. A variable name is not allowed to start with a number.

```python
study_program = "Master in Management"
print(study_program)
```

```
Master in Management
```

```python
1st_study_program = "Master in Management"
```

```
Input In [7]
  1st_study_program = "Master in Management"
   ^
SyntaxError: invalid syntax
```

  - Avoid using Python reserved keywords. For instance: print, for, while, etc.

```python
print = "This is a message"
```

```python
print("Something")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [12], in <cell line: 1>()
----> 1 print("Something")

TypeError: 'str' object is not callable
```

Variables

Rules and conventions

- **Conventions:**
  - Variable names should be short but descriptive. Use small letters and separate words using underscores _.

```
quarterly_revenue = 20

this_variable_is_for_quaterly_revenue = 20

r = 20
```

  - Constants should be named using CAPITALIZED variable names.

```
VOTING_AGE = 18
DRINKING_AGE = 21
```

  - It takes time and practice to be able to name variables properly. Expect your code to be read by other people!

Simple data types

Strings

- A string is a series of characters.
- A string can be defined inside double quotes or single quotes.

```
study_program = "Master in Management"
study_program = 'Master in Management'
```

- Literal (double) single quotations can be used inside a (single)double-quoted string.

```
text_content = 'The name of my study program is "Master in Management"'
print(text_content)
```

```
The name of my study program is "Master in Management"
```

Simple data types

Strings

- Multiple-lines string can be defined using the triple quote.
- This is useful when the textual content span multiple lines (e.g., writing docstring, long comments, etc.).

```
"""This is a very long description.
Such a long description can be defined
using the triple quote.
"""
```

Simple data types

Modern string formatting

- f-string was introduced in Python 3.6 and is currently the preferable way to format a string.

```python
name = "Dan"
age = 30
campus = "Heilbronn"
print(f"Hi, my name is {name}. I am {age}, and I am a student at TUM {campus}")
```

```
Hi, my name is Dan. I am 30, and I am a student at TUM Heilbronn
```

Simple data types

Manipulating strings

- The case of a string can be changed using the following methods:

```
text_content = "TUM campus Heilbronn"
```

```
text_content.upper()
```

```
'TUM CAMPUS HEILBRONN'
```

```
text_content.lower()
```

```
'tum campus heilbronn'
```

```
text_content.title()
```

```
'Tum Campus Heilbronn'
```

Simple data types

Manipulating strings

- Python uses the + symbol to concatenate strings.

```python
first_name = "Daniel"
last_name = "Ross"
full_name = first_name + " " + last_name
print(full_name)
```

```
Daniel Ross
```

- Note: Python will never implicitly interpret strings as numerical data.

```python
a = "34"
b = "13"
y = a + b
print(y)
```

```
3413
```

```python
a = "34"
b = "13"
y = int(a) + int(b)
print(y)
```

```
47
```

Simple data types

Manipulating strings – other useful methods

- To strip leading and trailing white spaces, use the strip() method.

```
last_name = " Ross "
last_name.strip()
```

```
'Ross'
```

- A string can be split using the split() method and specifying the delimiter.

```
address = "Bildungscampus-9"
address.split("-")
```

```
['Bildungscampus', '9']
```

- We can check if a string starts or ends with a certain string using the startswith() and endswith() methods.

```
address = "Bildungscampus-9"
address.startswith("Bi")
```

```
True
```

```
address = "Bildungscampus-9"
address.endswith("8")
```

```
False
```

- We can replace a substring within a string using the replace() method.

```
address = "Bildungscampus-9"
address.replace("-9",",9")
```

```
'Bildungscampus,9'
```

Simple data types

Manipulating strings – indexing and slicing

```
my_string = "Heilbronn City"
```

Forward indexing

0	1	2	3	4	5	6	7	8	9	10	11	12	13
H	e	i	l	b	r	o	n	n		C	i	t	y
-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Backward indexing

```
my_string[4]
```

```
'b'
```

```
my_string[-10]
```

```
'b'
```

```
my_string[4:9]
```

```
'bronn'
```

```
my_string[3:]
```

```
'lbronn City'
```

```
my_string[:5]
```

```
'Heilb'
```

```
my_string[-4:]
```

```
'City'
```

Simple data types

Integers & Floats - arithmetic operators

```python
int() # 1, 2, 58, 129
float() # 1.5, 5.7, 129.0
```

Operation	Description
`x + y`	Addition
`x - y`	Subtraction
`x * y`	Multiplication
`x / y`	Division
`x // y`	Truncating division
`x ** y`	Power (x to the y power)
`x % y`	Modulo (x mod y). Remainder.
`-x`	Unary minus
`+x`	Unary plus

```python
x = 7.2
y = 2
print(f"x + y = {x+y}")
print(f"x - y = {x-y}")
print(f"x * y = {x*y}")
print(f"x / y = {x/y}")
print(f"x // y = {x//y}")
print(f"x ** y = {x**y}")
print(f"x % y = {x%y}")
print(f"-x = {-x}")
print(f"+x = {+x}")
```

```
x + y = 9.2
x - y = 5.2
x * y = 14.4
x / y = 3.6
x // y = 3.0
x ** y = 51.84
x % y = 1.2000000000000002
-x = -7.2
+x = 7.2
```

Simple data types

Integers & Floats – common mathematic functions

Function	Description
`abs(x)`	Absolute value
`divmod(x,y)`	Returns `(x // y, x % y)`
`pow(x,y [,modulo])`	Returns `(x ** y) % modulo`
`round(x,[n])`	Rounds to the nearest multiple of 10 to the nth power.

```python
x = 7.25
y = 2
print(f"abs(x) = {abs(x)}")
print(f"divmod(x,y) = {divmod(x,y)}")
print(f"pow(x,y) = {pow(x,y)}")
print(f"round(x,1) = {round(x,1)}")
```

```
abs(x) = 7.25
divmod(x,y) = (3.0, 1.25)
pow(x,y) = 52.5625
round(x,1) = 7.2
```

Simple data types

Booleans

```
True
False
```

Operation	Description
`x == y`	Equal to
`x != y`	Not equal to
`x < y`	Less than
`x > y`	Greater than
`x >= y`	Greater than or equal to
`x <= y`	Less than or equal to

```python
x = 7.25
y = 2
print(f"x == y: {x==y}")
print(f"x != y: {x!=y}")
print(f"x < y: {x<y}")
print(f"x > y: {x>y}")
print(f"x >= y: {x>=y}")
print(f"x <= y: {x<=y}")
```

```
x == y: False
x != y: True
x < y: False
x > y: True
x >= y: True
x <= y: False
```

Simple data types

Logical operators

A	B	A and B	A or B	not A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

```python
x = True
y = False
print(f"x or y: {x or y}")
print(f"x and y: {x and y}")
print(f"not x: {not x}")
```

```
x or y: True
x and y: False
not x: False
```

```python
(2 > 3) and (6/2==3)
```

```
False
```

```python
(2 > 3) or (6/2==3)
```

```
True
```

Bit manipulation

Cautions

Operation	Description
`x << y`	Left shift
`x >> y`	Right shift
`x & y`	Bitwise and
`x | y`	Bitwise or
`x ^ y`	Bitwise xor (exclusive or)
`~x`	Bitwise negation

```
a = 0b11001001 # Binary representation for int(201)
mask = 0b11110000 # Binary representation for int(240)
print(bin(a & mask))
print(a & mask)
```

```
0b11000000
192
```

- Most common data science libraries use the & and | operators differently! (not as bit manipulation operators)

Question?

Quiz Time!

Which of the following operations results in 16?

a) 4*4

b) 4x4

c) 2^4

d) 17//16

Quiz Time!

Given the string mystring = “Technical University of Munich”, what is the results of mystring[-6:]

a) Techni

b) TUM

c) Munich

d) Univer

Quiz Time!

What is the printout of the following script?

```
a = 6

b = 10

print( a == 10 or b == 10 )
```

a) **True**

b) False

c) None

d) Cannot print

Quiz Time!

In Python, what is the proper way to name a variable that stores the value of the net profit calculation?

a) netProfit

b) NET_PROFIT

c) net_profit

d) net profit

Quiz Time!

Given the variable my_str = “data_manipulation.csv”

What is the printout of the following code:

```
print(my_str.split("_")[1].startswith("m"))
```

a) **True**

b) False

c) d

d) m