intro

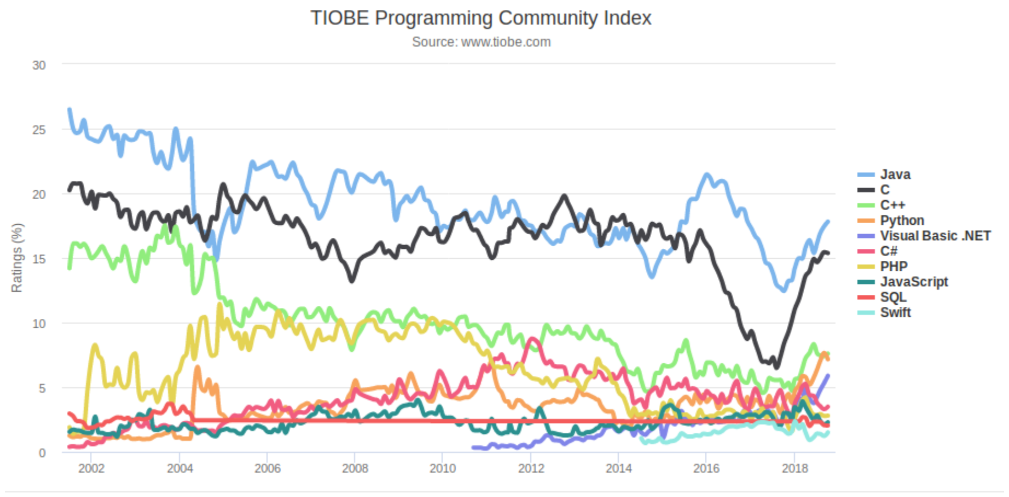
November 7, 2018

1 Intro to programming using python

1.1 Objectives

- To get a brief overview of what Python is
- To understand computer basics and programs
- To write a small python program using: https://repl.it/languages/python3

1.1.1 Python has been increasingly popular in the last few years.

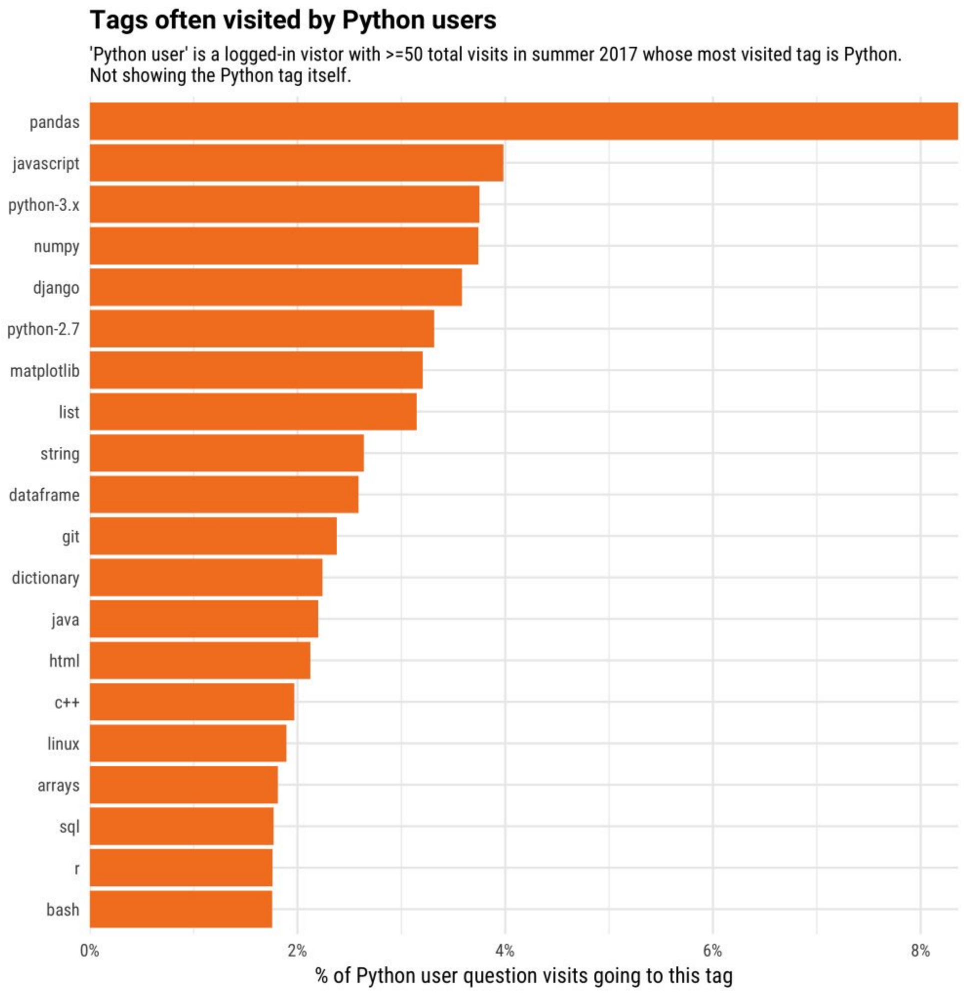


Graph of popularity from tiobe.com

1.1.2 In particular due to the adoption of python by the data science community which is illustrated by this study of stackoverflow:

1.2 What is python used for?

- Web development
- Data analysis
- Web scraping
- Gaming
- Robotics



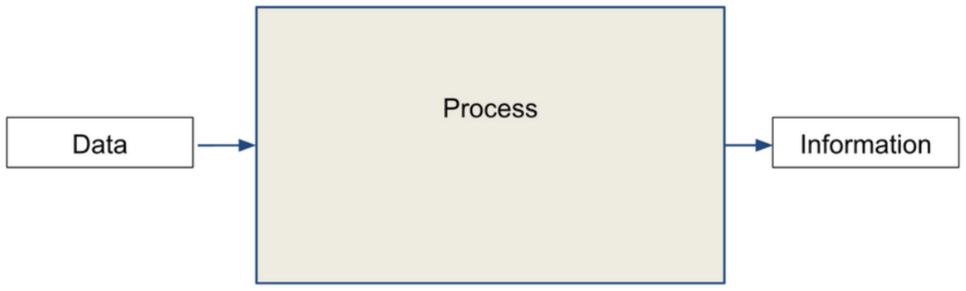
Graph which shows that pandas is the 1st python related search, followed by web related terms

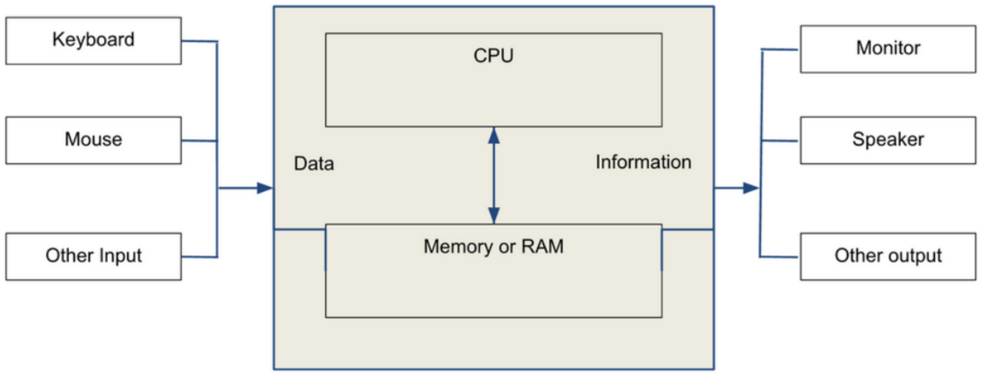
Who uses python

- Machine learning
- Testing
- ...

1.3 Before going more into details... What is a computer? (1/2)

An electronic device that is receiving data input, storing (in RAM) and processing (in the CPU) them and producing information in output.





1.4 What is a computer? (2/2)

1.5 What is a program?

- Computer programs, known as software, are instructions to the computer.
- You tell a computer what to do through programs. Without programs, a computer is an empty machine. Computers do not understand human languages, so you need to use computer languages to communicate with them.
- Programs are written using programming languages.

1.6 Different types of Programming Languages

- Machine language is a set of primitive instructions built into every computer. The instructions are in the form of binary code. The programs in machine language are highly difficult to read and modify. For example, to add two numbers, you might write an instruction in binary like this:

```
1101101010011010
```

- The high-level languages are English-like and easy to learn and program. For example, the following is a high-level language statement that multiply two number:

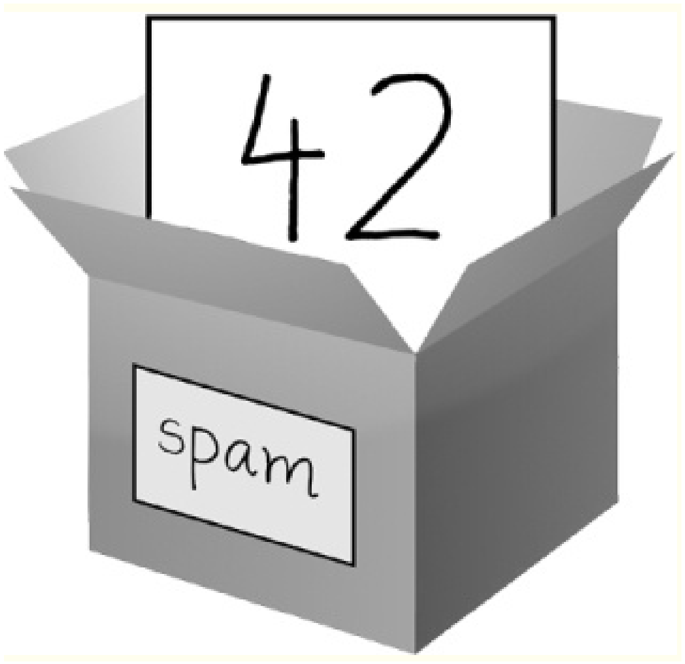
```
area = 5 * 5
```

1.7 Diving in...

- Simple arithmetic operations
- Variables
- Comments
- Expressions
- Indentations
- Functions
- Conditions

1.8 Variable (1)

It is a space created in memory (in RAM) where we can temporarily store values or data. We use the sign '=' for assigning a value to a variable.
You can think of it like a box.



1.9 Variable (2)

When a new value is assigned to a variable, the old one is forgotten.

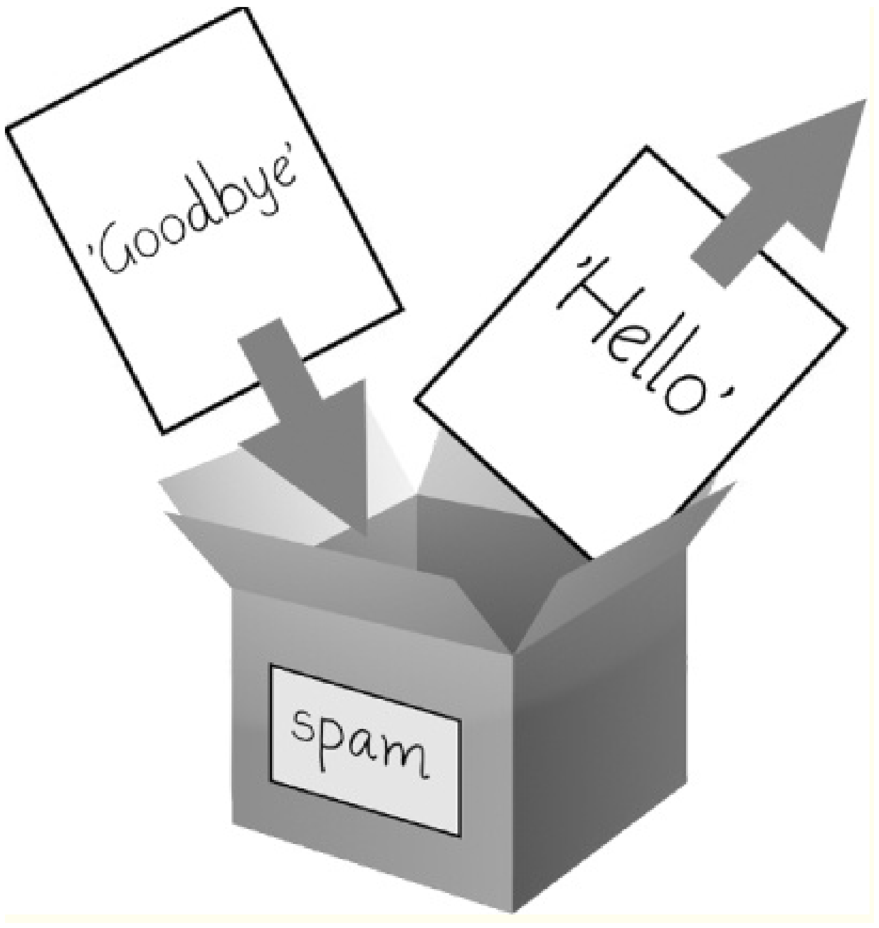
1.10 Variable (3)

- A variable name is a non-empty sequence of characters of any length with:
- The start character can be the underscore "_" or a capital or lower case letter.
- Python keywords are not allowed as identifier names!
- The variable has a name so that we can reuse it. When we use a variable, it is for retrieving the value that it is holding.
- We do not specify the type (integer, string) of the variable, python sees it automatically.

1.11 Comments

- Is not executed, it is ignored by the program
- # is commenting everything on the right of it

```
a = 1
# this is ignored, commented
```


'Goodbye'
'Hello'
spam

1.12 Expression

It represents a value, like a number or a string

```
>>> 1 # is an expression
>>> 2 + 3 # is also an expression
>>> "hello" # as well
```

We can put an expression in a variable

```
my_variable = 1 + 1
```

1.13 Function execution (1)

- A function has a name
- A function has parameters (that can be optional)
- A function is executed!

Example:

```
print('Hello world')
```

1.14 Function execution (2)

Example: input()

- Prompt the user to enter an input, the programs waits for an input and as soon as the user press Enter, the program carries on
- The input function **returns** a value that we can store in a variable and then reuse.

```
In [ ]: name_of_the_user = input("Enter your name: ")
        print(f"Hello {name_of_the_user}")
```

1.15 Exercice: computing the age of the user

Ask a user to enter the year he was born (use `input()`), compute his age and tell him how old he will turn this current year (use `print()`).

1.16 Indentation

The indentation is the increase or decrease of space between the left margin and the first character of the line.

The code need to be properly indented, else python will raise an error.
For example, what is wrong here?

```
In [ ]: if True:
        print("indented properly ")
```

1.17 Function definition (1)

Aside the built-in functions such as `print` and `input` that are already implemented, we can also define our own functions. We use the keyword **def**, the name of the function, the brackets, and the colon

Then the body of the function needs to be indented

```
def name_of_the_function():
    # body of the function
```

When we define a function, we just make python see that the function exist but it is not executed

```
def my_function():
  print("THIS IS MY FUNCTION")
```

1.18 Function definition vs execution (2)

To call or execute or run a function, we use the name of the function AND the brackets, without the brackets, the function is not called.

`name_of_the_function()`

Notice the difference between defining and calling a function

```
def my_function():
  print("THIS IS MY FUNCTION")

my_function()
```

1.19 Conditions: Controlling the flow of our programs

We can represent the flow of execution with a flow chart

1.20 Structure of a simple if statement

Pseudo code:

```
if condition:
    # statement (mind the indentation)
```

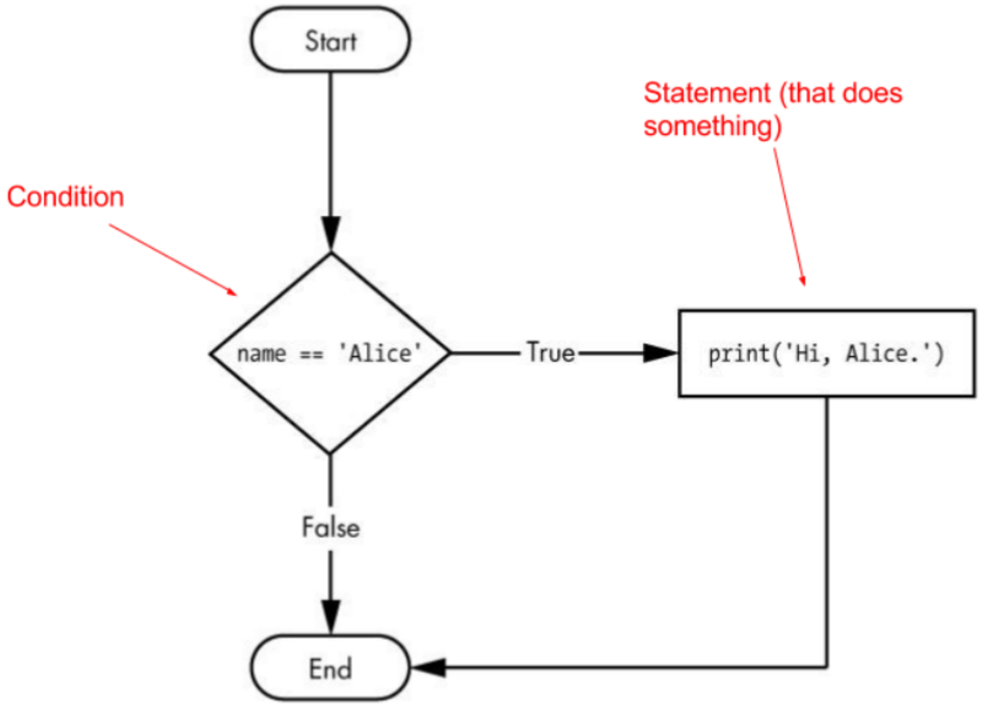
Example, representation of the flow chart example in python code:

```
if name=='Alice':
    print('Hi Alice')
```

1.21 The two-way if statement

Pseudo code:

```
if condition:
    # statement (mind the indentation)
else:
    # statement executed when the condition is False
```



Example, representation of the flow chart example in python code with an else statement:

```python
if name=='Alice':
    print('Hi Alice')
else:
    print('Hi')
```

1.22 Difference between ’==’ and ’=’

- The sign = is the sign of **assignment**, it is used for assigning a value to a variable
- The sign == is the sign of **comparison**, it compares 2 values and return a boolean (True or False)

1.23 Exercise: password

Create a program that ask the user for a password.

- Have the password defined in "clear" (i.e. not encrypted as you would do in standard application) in your program, in a variable called "PASSWORD"
- Use input() to receive the password entered by the user
- If the word entered by the user matches the password, display "Access Granted", else, "Forbidden"

```
In [ ]: # This is a comment
        # Replace the comment with the code
        # You begin by setting the password in a variable PASSWORD (remember that the sign '=' i
        # Use input and retrieve the value of input in an other variable
        # use conditions to check if the password is correct and display if it is correct or not
```

1.24 Extra resources

- Automate the boring stuff with python