Python IT Automation: Intro to Python, Regex, and Bash Scripting

Ardy Seto Priambodo

Lecturer at Electronics Engineering
Universitas Negeri Yogyakarta
ardyseto@uny.ac.id | ardy.seto@bangkit.academy

bangkit

Ardy Seto Priambodo

About Me

Latest Work Experiences:

- Lecturer, Electronics Engineering UNY — **2018 - present**
- Spv. Prod. Engineering, Astra Daihatsu Motor — **2013 - 2014**

Education:

- Universitas Gadjah Mada — **2014 - 2018**
  Master of Electrical Engineering
- Institut Teknologi Sepuluh Nopember — **2007- 2012**
  Bachelor of Electrical Engineering

bangkit

Ground Rules

Observe the following rules to ensure a supportive, inclusive, and engaging classes

Give full attention in class

Mute your microphone when you're not talking

Keep your camera on

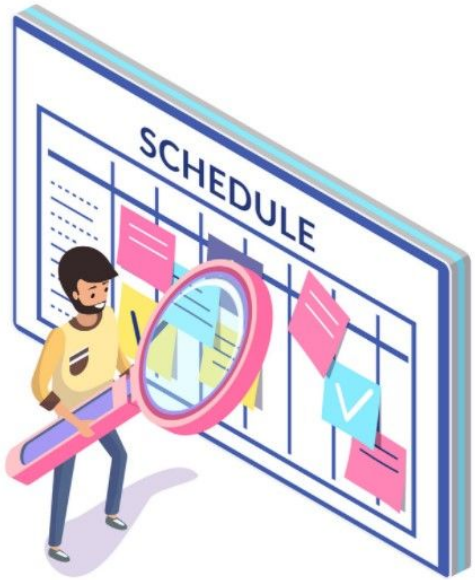
Use raise hand or chat to ask questions

Make this room a safe place to learn and share

bangkit

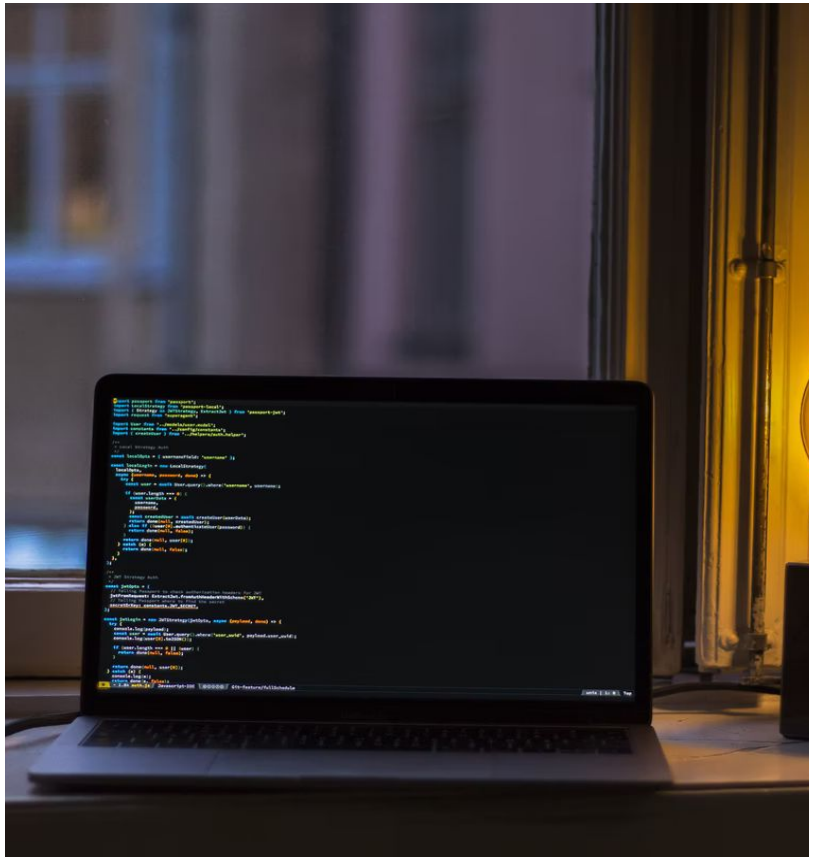
Time Scheduling

- 40 min. Introduction and Deliver ILT Topic.
- 40 min. Sharing Session.
- 30 min. Discussion
- 10 min. Quiz and Closing



Outline Session

- Hello Python
- Python Basic Syntax
- Python Data Structure
- Regular Expressions
- Managing Files, Data & Processes with Python
- Bash Scripting



bangkit

Hello Python

bangkit

Hello Python

What is **Python**?

Python is a dynamic, interpreted (bytecode-compiled) language.

Why programming with **Python**?

- Easy syntax
- Most chosen language for IT
- Omnipresent

Hello in Python Programming Language

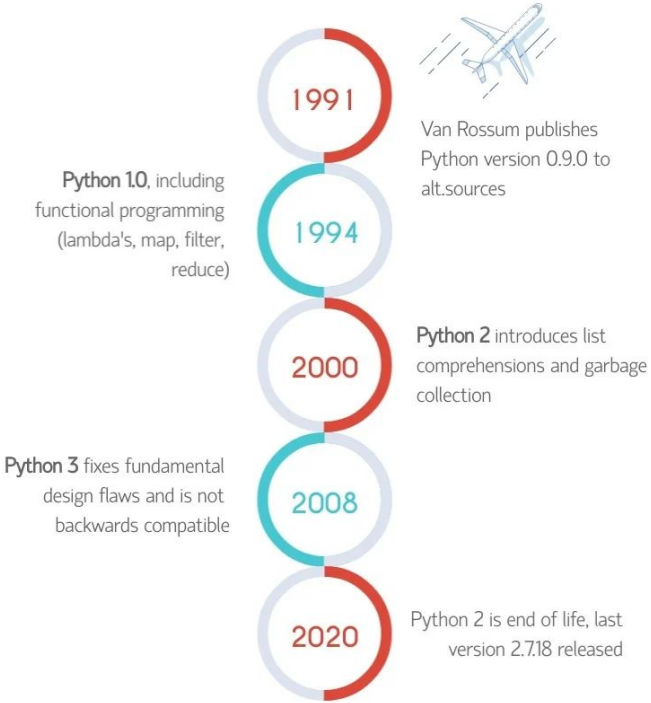
```
print('hello, bangkit!')

# this is comment,
# won't be interpreted
```

History of Python

Who created Python?

It was created by Guido van Rossum, and first released on February 20, 1991 as a hobby project



Why learn Python?

Feb 2022	Feb 2021	Change	Programming Language	Ratings
1	3	^	Python	15.33%
2	1	v	C	14.08%
3	2	v	Java	12.13%
4	4		C++	8.01%
5	5		C#	5.37%
6	6		Visual Basic	5.23%
7	7		JavaScript	1.83%
8	8		PHP	1.79%
9	10	^	Assembly language	1.60%
10	9	v	SQL	1.55%

TIOBE Index for February 2022

Python is the most popular programming language

bangkit

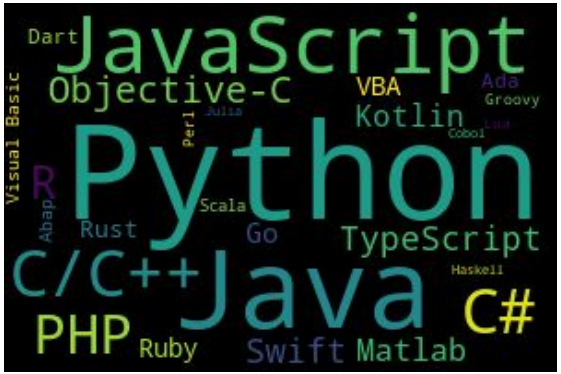
Why learn Python?

Worldwide, Feb 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.52 %	-1.7 %
2		Java	18.12 %	+1.2 %
3		JavaScript	8.9 %	+0.4 %
4	↑	C/C++	7.62 %	+1.1 %
5	↓	C#	7.39 %	+0.6 %
6		PHP	5.81 %	-0.3 %
7		R	4.04 %	+0.2 %
8		Objective-C	2.46 %	-1.1 %
9		Swift	2.03 %	+0.0 %
10		TypeScript	1.94 %	+0.2 %

PYPL

Python get the first position in PYPL Index



bangk!t

Why learn Python?

- it's easy to learn
- it's easy to teach
- it's easy to use for writing new software
- it's easy to understand
- it's easy to obtain, install and deploy
- Machine Learning

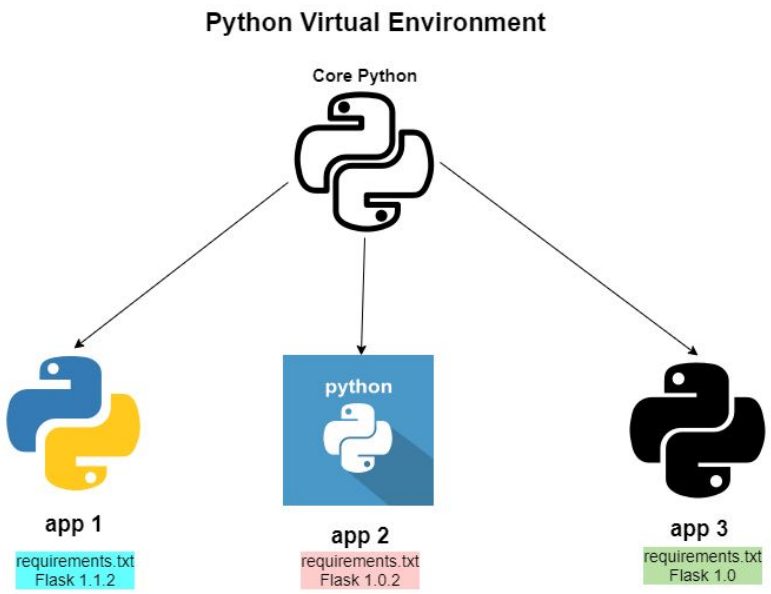
Python Distribution

There are some of python distribution:

- ActivePython
- Anaconda (https://anaconda.org/)
- ChinessePython
- Win9xPython
- IPython
- Portable Python
- PyPy
- PythonForArmLinux
- PythonLabsPython
- MicroPython

Virtual Environment

- At its core, the main purpose of Python virtual environments is to create an isolated environment for Python projects.
- This means that each project can have its own dependencies, regardless of what dependencies every other project has.
- Conda is one of tools can be used for maintaining virtual environments in Python



Getting Ready for Python

- To check Python installed
  - open a terminal or command prompt
  - execute Python command
  - passing --version as a parameter.
- Result similar to "unrecognized command" means no Python installed.

Check on terminal (OS Linux or MacOS)

shell prompt, no need to type $

```
$ python --version
Python 3.7.9

$ python3 --version
Python 3.7.9
```

as alternative, also check python3

Python installed, version 3.7.9

Check on Command Prompt or PowerShell (OS Windows)

command prompt, no need to retype

```
C:\Users\bangkit> python
--version
Python 3.7.9
```

Python installed, version 3.7.9

Getting Ready for Python

shell prompt,
no need to type $

Check on terminal (OS Linux or MacOS)

```
$ python --version
Python 3.9.7

$ python3 --version
Python 3.7.9
```

as alternative,
also check python3

Python installed,
version 3.7.9

```
┌─[ardyseto@192:~]─[10:42:47]
└─>$ python --version
Python 3.9.7
```

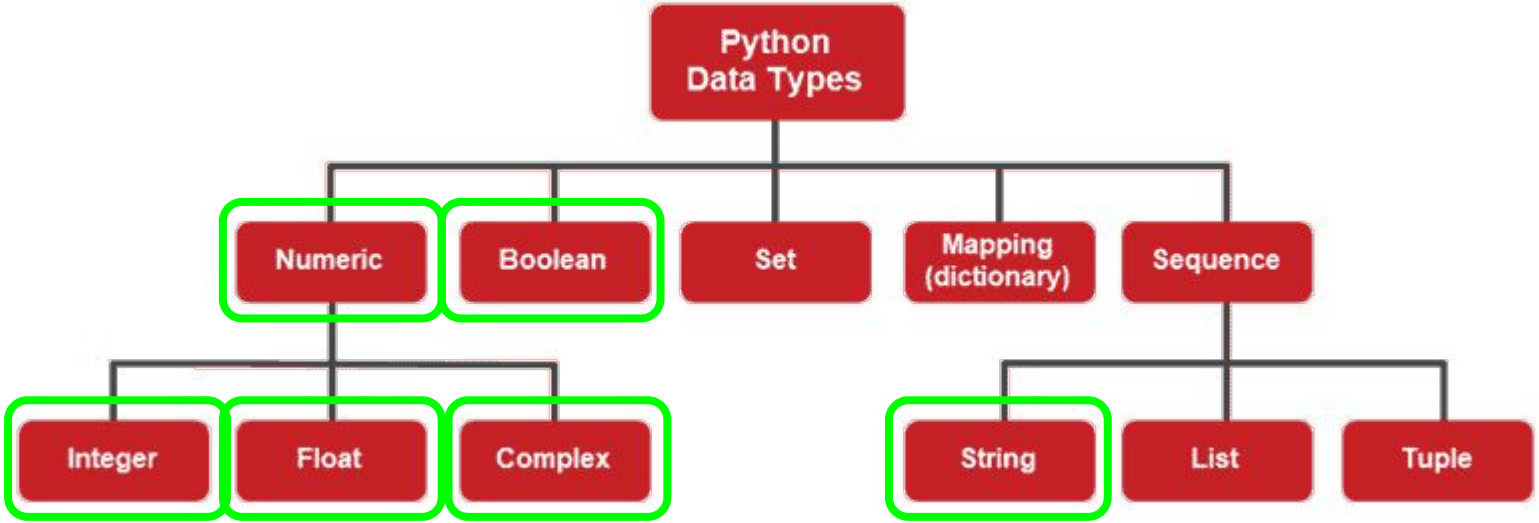
Basic Syntax

Basic Python Syntax: Data Types

- String (str): text.
- Integer (int): numbers, without fraction.
- Float: numbers with fraction.
- Boolean (bool): data type which only has 2 values

We can convert from one data type to others by committing to implicit conversion or defining an explicit conversion.

Basic Python Syntax: Data Types



Basic Python Syntax: Variables

- Name to certain values
- The values can be any data type
- The process of storing a value inside a variable is called an assignment.
- Can only be made up of letters, numbers, and underscore.
- Can't be Python **reserved keywords**.

Do This

```
length = 10
width = 2
area = length * width
name = 'Saturnus'
print(area)
print(type(width))
print(type(str(area)))
print(name)
```

Don't Do This

```
def = 'Function'
class = 'Class'
print(and)
print(or)
```

Cannot be used,
will expected error

Basic Python Syntax: Functions

- Define function with **def** keyword.
- Function has body, written as a block after colon in function definition. The block has indented to the right.
- To get value from a function use the **return** keyword.

Define Function, to Return String

Function name Parameter

```
def greeting(name):
    return 'hello, ' + name
```

Indentation with 4 spaces

Call Function

```
print(greeting('bangkit'))
```

Output

```
hello, bangkit
```

Conditional & If Statements

- The ability of a program to alter its execution sequence is called branching.
- The if block will be executed **only if the condition is True**.
- Use elif & else statement to handle multiple conditions.

Condition Evaluations

The Condition of Comparison

```
if hour < 12:
    print("Good morning!")
```

Indentation with 4 spaces

```
def check(number):
    if number > 0:
        return "Positive"
    elif number == 0:
        return "Zero"
    else:
        return "Negative"
```

Conditional & If Statements

coursera | Search in course | Search | ML22-0022 Ardy S...

Crash Course on Python > Week 2 > Conditionals Cheat Sheet

Previous Next

Expressions and Variables

Functions

Conditionals

- **Video:** Comparing Things 4 min
- **Reading:** Comparison Operators 10 min
- **Video:** Branching with if Statements 2 min
- **Reading:** if Statements Recap 10 min
- **Video:** else Statements 3 min
- **Reading:** else Statements and the Modulo Operator 10 min
- **Video:** elif Statements 3 min
- **Reading:** More Complex Branching with elif Statements 10 min
- **Reading:** Conditionals Cheat Sheet 10 min
- **Practice Quiz:** Practice Quiz: Conditionals 5 questions

Module Review

Conditionals Cheat Sheet

Conditionals Cheat Sheet

In earlier videos, we took a look at some of the built-in Python operators that allow us to compare values, and some logical operators we can use to combine values. We also learned how to use operators in if-else-elif blocks.

It's a lot to learn but, with practice, it gets easier to remember it all. In the meantime, this handy cheat sheet gives you all the information you need at a glance.

Comparison operators

- a == b: a is equal to b
- a != b: a is different than b
- a < b: a is smaller than b
- a <= b: a is smaller or equal to b
- a > b: a is bigger than b
- a >= b: a is bigger or equal to b

Logical operators

- a and b: True if both a and b are True. False otherwise.
- a or b: True if either a or b or both are True. False if both are False.
- not a: True if a is False, False if a is True.

Branching blocks

In Python, we branch our code using if, else and elif. This is the branching syntax:

```
if condition1:
    if-block
```

https://www.coursera.org/learn/python-crash-course/supplement/R9diu/conditionals-cheat-sheet

bangkit

Loops: while & for

- while loop instruct computer to continuously execute code based on the value of a condition.
- for loop iterates over a sequence of values.

while loop

Initialization of variable

```
x = 7  # also try with x = 0
```

The conditions

```
while x > 0:
    print("positive x=" + str(x))
    x = x - 1
print("now x=" + str(x))
```

for loop

The sequence

```
for x in range(3):  # 0, 1, 2
    print("x=" + str(x))
```

Loops: break & continue

- Both while and for loops can be interrupted using the break keyword.
- Use the continue keyword to skip the current iteration and continue with the next one.

break from loop

```
for x in range(3):
    print("x=" + str(x))
    if x == 1:
        break  # quit from loop
```

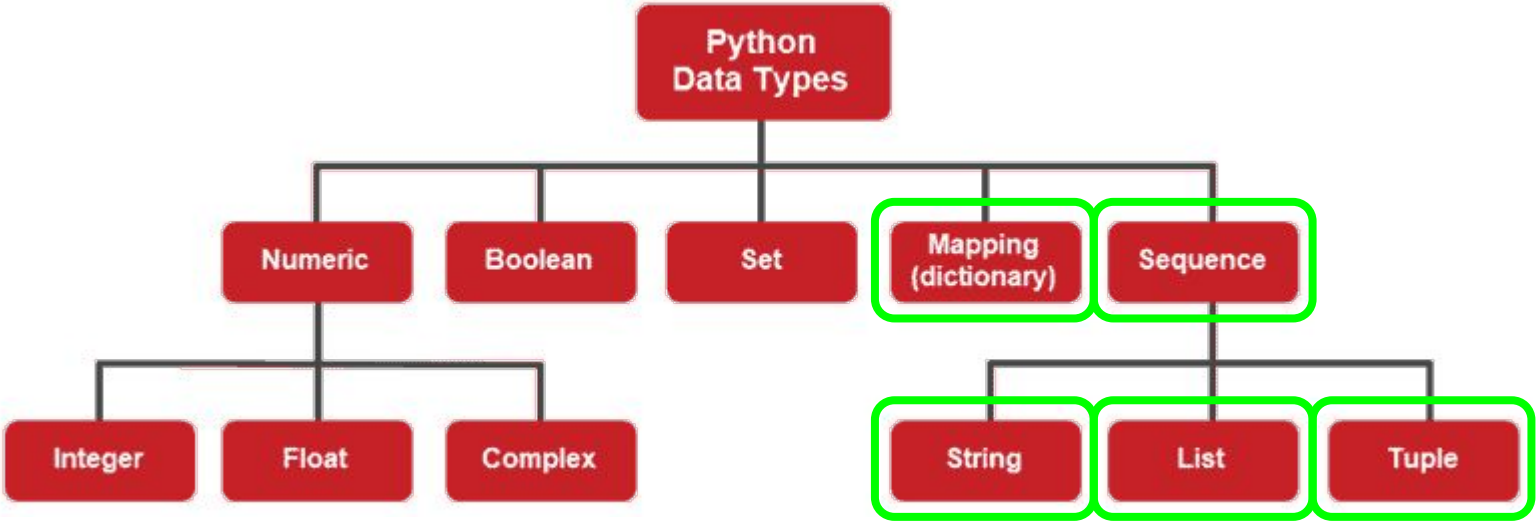
continue inside loop

```
for x in range(3, 0, -1):
    if x % 2 == 0: #  Modulus
        continue  # skip even
    print(x)
```

Python
Data Structure

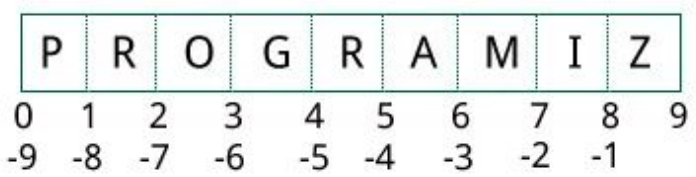
bangkit

Basic Python Syntax: Data Types



Data Strings

- Represent a piece of text.
- To access substring, use index or slicing.
- Strings in Python are immutable
- Provide a bunch of methods for working with text.



Strings

```
name = 'bangkit'
program_year = "it's the 2nd"
multi_line = """hello,
email test. Signature."""

print(name[1])  # a

print(name[4:len(name)-1])  # ki

year = "it's 2021"
year[-1] = "0"  # TypeError

print(name.upper())  # BANGKIT
```

Index

Slicing

bangkit

* Complete string methods in Python docs https://docs.python.org/3/library/stdtypes.html#string-methods

Python String Operations

- Concatenation of Two or More Strings. Joining of two or more strings into a single one is called concatenation.
- Iterating Through a string. We can iterate through a string using a for loop. Here is an example to count the number of 'l's in a string.
- String Membership Test. We can test if a substring exists within a string or not, using the keyword in.
- Built-in functions to Work with Python. Various built-in functions that work with sequence work with strings as well.

List

- In Python list can contain a different value.
- Python use square brackets [] to indicate where the list starts and ends.
- List in Python are mutable.

List

list starts

list ends

```
paths = ['ML', 'Cloud']
for path in paths:
    print(path)  # element per line

paths.append('Android')
paths.remove('Cloud')
paths.insert(1, 'Mobile')
paths.pop(-1) # remove 'Android'

# change 'ML' to 'Machine Learning'
paths[0] = 'Machine Learning'

# list comprehensions
even = [x*2 for x in range(1,5)]
print(even)  # [2, 4, 6, 8]
```

Tuples & Dictionary

- Tuples can contain elements of any data type. But, unlike lists, tuples are immutable.
- Dictionary in Python contain **pairs of keys and values**.
- To get a dictionary value, use its corresponding key.
- Dictionary in Python are mutable.

Tuple

tuples starts

tuples ends

```
paths = ('ML', 'Cloud')
for path in paths:
    print(path)  # element per line
```

Dictionary

dictionary key

dictionary value

```
students = {'ml': 500,'mobile': 700,
'cloud': 900}
print(students['cloud'])  # 900

students['ml'] = 1000
```

Regular Expression

bangkit

Regular Expressions

- Regex is a search query for text that's expressed by string pattern.
- Regular expressions in Python uses raw string (r"")
- Circumflex or Caret (^) pattern matches the beginning of the line.
- Dot or period (.) matches any character.

Simple Matching in Python re

```
import re

result = re.search(r"aza", "plaza")
print(result)
<re.Match object; span=(2, 5),
match='aza'>
print(re.search(r"aza", "maze"))
None

print(re.search(r"^x", "xenon"))
<re.Match object; span=(0, 1),
match='x'>

print(re.search(r"p.ng", "sponge"))
<re.Match object; span=(1, 5),
match='pong'>
```

Regular Expressions

- To matched a range of characters, use another feature of regexes called character classes or square brackets ([ ]).
- Use the pipe symbol or vertical bar (|) to match one expression or another.
- Dollar sign ($) pattern match the end of the line.

```
import re

print(re.search(r"cloud[a-zA-Z0-9]",
"cloud9"))
<re.Match object; span=(0, 6),
match='cloud9'>

print(re.search(r"[^a-zA-Z]", "This is a
sentence."))
<re.Match object; span=(4, 5), match=' '>

print(re.search(r"cat|dog", "I like cats."))
<re.Match object; span=(7, 10), match='cat'>

print(re.search(r"cats$", "I like cats"))
<re.Match object; span=(7, 11),
match='cats'>
```

Regular Expressions

Repeated matches is another regex concept.

- The star (*) **takes as many character as possible.**
- The plus (+) character **matches one or more occurrences of the character before it**.
- The question (?) **mark symbol means either zero or one occurrence of the character** before it.

```
import re

print(re.search(r"Py[a-z]*n",
"Python Programming"))
<re.Match object; span=(0, 6),
match='Python'>

print(re.search(r"o+l+", "woolly"))
<re.Match object; span=(1, 5),
match='ooll'>

print(re.search(r"p?each", "I like
peaches"))
<re.Match object; span=(7, 12),
match='peach'>
```

bangkit

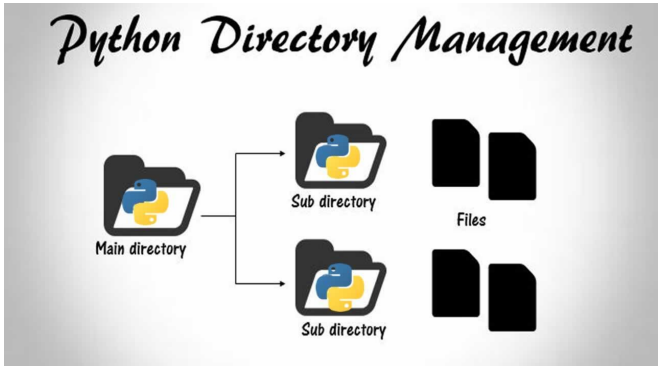
* More about regex introduction visit https://www.regex101.com

Managing Files Using Python

bangkit

Managing Files with Python

- Files are named locations on disk to store related information.
- They are used to permanently store data in a non-volatile memory (e.g. hard disk).
- When we want to read from or write to a file, we need to open it first.
- When we are done, it needs to be closed so that the resources that are tied with the file are freed.
- Hence, in Python, a file operation takes place in the following order:
  - Open a file
  - Read or write (perform operation)
  - Close the file



Managing Files with Python

- Function open will start to open the file.
- To read file, use the readline & read function.
- To ensure that all open files are always closed, use an alternative method to write it as a block of code using the with keyword.

Read Existing File

```
file = open("spider.txt")
print(file.readline())
file.close()
```

```
The itsy bitsy spider climbed
up the waterspout.
```

open mode

```
with open("spider.txt", “r”) as
file:
    print(file.read())
```

```
The itsy bitsy spider climbed
up the waterspout.
Down came the rain
and washed the spider out.
```

bangkit

* Complete modes for function open in Python docs https://docs.python.org/3/library/functions.html#open

Managing Files with Python

- Use **os** module to interact with operating system in Python
- To read and writing tabular data in CSV format, use an **csv** module.

Working with Directory

```
import os
os.mkdir("new_dir")
```

Reading CSV Files

```
import csv
file = open("data.csv")
csv_f = csv.reader(file)
for row in csv_f:
    name, phone, role = row
    print(name+': '+role)
file.close()
```

```
Sabrina Green: System Administrator
Eli Jones: IT specialist
```

bangkit

* Python Documentation for module os & csv
https://docs.python.org/3/library/os.html, https://docs.python.org/3/library/csv.html

Managing Data & Processes

Three different I/O streams by default:

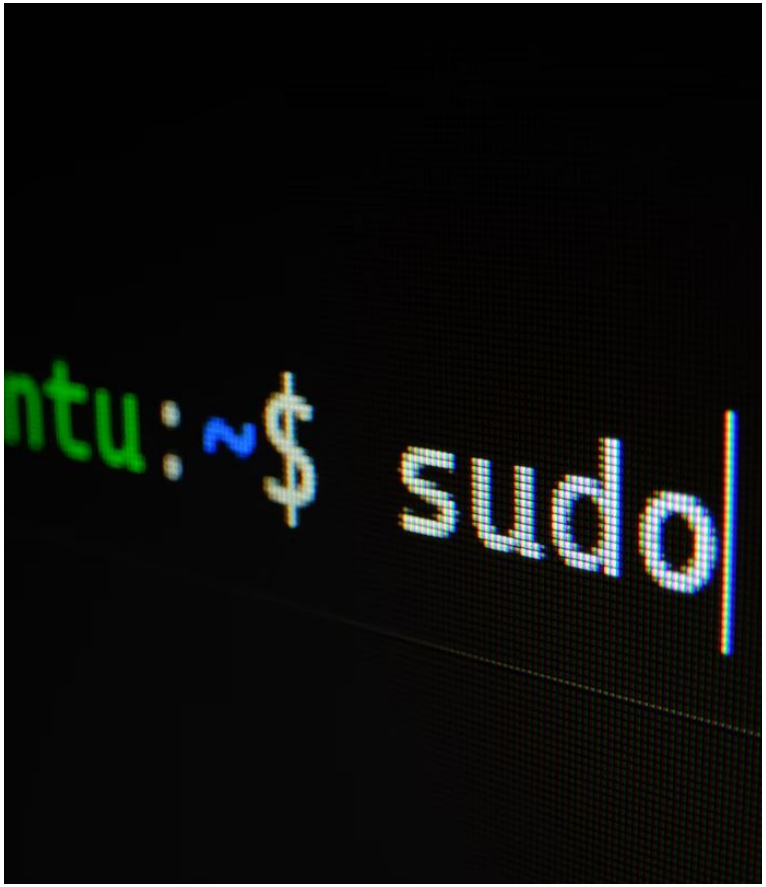
- Standard input stream (**STDIN**): the input function.
- Standard output stream (**STDOUT**): the print function.
- Standard error stream (**STDERR**): specifically as a channel to show error messages and diagnostic from the program.

Bash Scripting

bangkit

Why Bash Scripting?

- Flexible
- Less Resource
- Used in cloud environment
- Automate command



bangkit

Most Commonly Used Bash Command

- Linux commands:
  - **echo**: print information (like environment variable) to standard output
  - **cat** file: shows the content of the file through standard output
  - **ls**: lists the contents of the current directory
  - **cd** directory: change current working directory to the specified one
  - **rm**: remove file or directory (with specific arguments)
  - **chmod** modifiers files: change permissions for the files according to the provided modifiers
  - **man:** show command documentation
  - **clear**: clear a command line screen/window for a fresh start

Example of Linux Terminal

```
ardyseto — fish /Users/ardyseto — -fish — 80×17
┌─[ardyseto@192:/V/D/Downloads]─[16:56:27]
└─>$ cd ~/                                                    (base)
┌─[ardyseto@192:~]─[16:56:33]
└─>$ pwd                                                      (base)
/Users/ardyseto
┌─[ardyseto@192:~]─[16:56:35]
└─>$ ls                                                       (base)
Applications                Google Drive                bin
Applications (Parallels)    Library                     disconnect.txt
Creative Cloud Files        Movies                      opt
Desktop                     Music                       reconnect.txt
Documents                   Pictures
Downloads                   Public
┌─[ardyseto@192:~]─[16:56:50]
└─>$                                                          (base)
```

bangkit

Sharing Session

bangkit

Discussions

bangkit

Quiz

bangkit

Thank You

bangkit