3DC

Developer Student Club

Python Workshop Series 2020 Spring

Session 01: Introduction to Programming in Python

27th Oct 2020

01 Computational Thinking

02 Terminologies

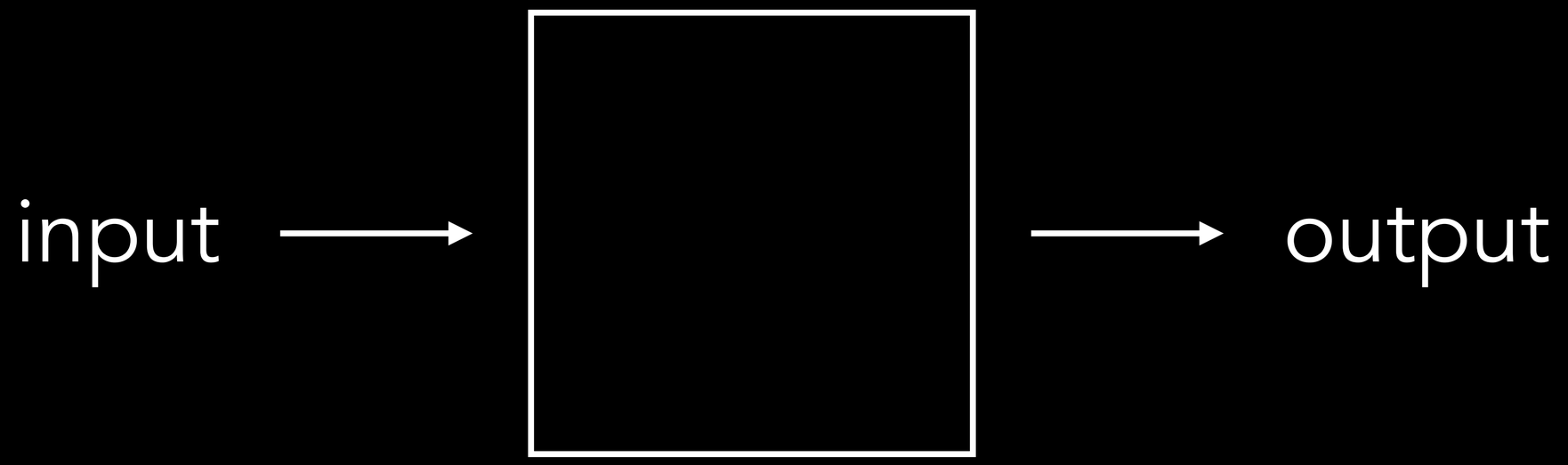
03 Python 101

04 Put Into Practice

01

Computational Thinking

What is programming?



Program / Code / Software / Algo

pseudocode

Pseudocode

```
While store operating:
    Wait at the counter until customer comes
        If someone approaches the counter:
            Say “Hey, what can I get for you?”, with smile
            If customer replies “Chocolate Waffle”
                Start making chocolate waffle
                Say “chocolate waffle is 1 dollar and 20 cents.”
            If customer replies “Plain Waffle”
                Start making Plain waffle
                Say “plain waffle is 1 dollar.”
        If payment received
            Check if payment amount is correct
            Say “thank you”
        If the waffle is ready:
            Pass waffle to the current customer
```

Pseudocode

```
While store operating:
    Wait at the counter until customer comes
        If someone approaches the counter:
            Say “Hey, what can I get for you?”, with smile
            If customer replies “Chocolate Waffle”
                Start making chocolate waffle
                Say “chocolate waffle is 1 dollar and 20 cents.”
            If customer replies “Plain Waffle”
                Start making Plain waffle
                Say “Plain waffle is 1 dollar.”
        If payment received
            Check if payment amount is correct
            Say “thank you”
        If the waffle is ready:
            Pass waffle to the current customer
```

Pseudocode

```
While store operating:
    Wait at the counter until customer comes
        If someone approaches the counter:
            Say “Hey, what can I get for you?”, with smile
            If customer replies “Chocolate Waffle”
                Start making chocolate waffle
                Say “chocolate waffle is 1 dollar and 20 cents.”
            If customer replies “Plain Waffle”
                Start making Plain waffle
                Say “Plain waffle is 1 dollar.”
        If payment received
            Check if payment amount is correct
            Say “thank you”
        If the waffle is ready:
            Pass waffle to the current customer
```

Pseudocode

```
while storeIsOperating():
    if customerApproches():
        if someoneApproaches():
            sayWithSmile("Hey, what can I get for you?")
            if customerReply == "Chocolate Waffle"
                startMaking("Chocolate Waffle")
                say("chocolate waffle is 1 dollar and 20 cents.")
            if customerReply == "Plain Waffle"
                startMaking("Plain Waffle")
                say("Plain waffle is 1 dollar.")
        if paymentReceived():
            checkPayment()
            say("thank you")
        if waffleReady():
            passWaffle(currentCustomer)
```

02

Terminologies

Interpreted Language? Compiled Language?

Compiled	Interpreted
Converted directly into machine code that the processor can execute	A different program, the interpreter, reads and executes the code
Entire program needs to be manually compiled first - ‘Build’ step	No ‘build’ step - Runs through the program line by line, executing each command
Using the analogy given below, if you want to use a recipe for Hummus originally written in Greek, this refers to you **obtaining the translated recipe**	Meanwhile, this refers to you **asking your friend to translate each line of the recipe** to you when you want to use the recipe.

Source: https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/

Source Code —Parsing / Semantic Analysis→ —Compile (Code Generation) / By Compiler→ **Machine Code** —Execute / By CPU→ **Result**

Source Code —Parsing / Semantic Analysis→ —Interpret & Execute / By Interpreter→ **Result**


C++
Java™

Compiled Programming Languages

Interpreted Programming Languages

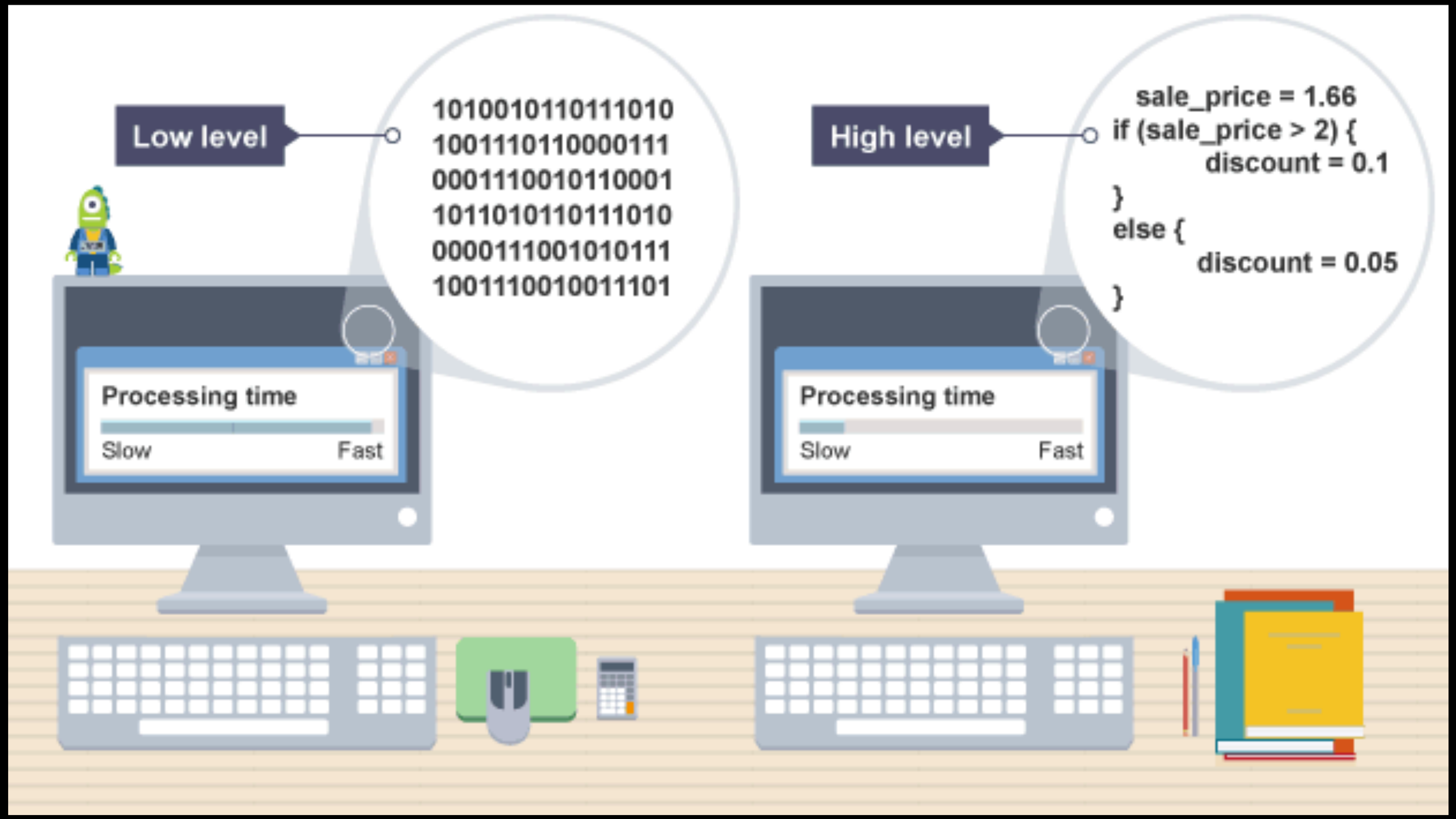
High Level / Low Level Programming Languages?

```
100                                 ;-----------------------------------------------------------
101                                 ; zstr_count:
102                                 ; Counts a zero-terminated ASCII string to determine its size
103                                 ; in:   eax = start address of the zero terminated string
104                                 ; out:  ecx = count = the length of the string
105
106                                 zstr_count:                     ; Entry point
107 00000030 B9FFFFFFFF                 mov  ecx, -1                ; Init the loop counter, pre-decrement
108                                                                 ;  to compensate for the increment
109                                 .loop:
110 00000035 41                         inc  ecx                    ; Add 1 to the loop counter
111 00000036 803C0800                   cmp  byte [eax + ecx], 0    ; Compare the value at the string's
112                                                                 ;  [starting memory address Plus the
113                                                                 ;  loop offset], to zero
114 0000003A 75F9                       jne  .loop                  ; If the memory value is not zero,
115                                                                 ;  then jump to the label called '.loop',
116                                                                 ;  otherwise continue to the next line
117                                 .done:
118                                                                 ; We don't do a final increment,
119                                                                 ;  because even though the count is base 1,
120                                                                 ;  we do not include the zero terminator in the
121                                                                 ;  string's length
122 0000003C C3                         ret                         ; Return to the calling program
```

Equivalent machine code

Assembly language showing mnemonics and comments to explain the code.

Source: https://www.mrdfinch.com/high-and-low-level-languages.html



Source: https://medium.com/@brettschules/high-level-and-low-level-languages-62776d0b89f0

Domain-specific Visual Programming	Dynamo DesignScript
	Scratch
Very high-level Languages	Python
	C#
High-level Languages	C
	Assembly Language
Low-level Languages	Machine Code

IDE? Text Editor?

Atom, Vim, Emacs, Sublime Text, Visual Studio Code, PyCharm, Spyder

Java – reposearch/src/it/unimi/dsi/mg4j/index/AbstractBitStreamIndexWriter.java – Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Ex | Type Hierar

- Jade
  - src
  - JRE System Library [jre6]
  - Referenced Libraries
  - deps
  - jars
  - libraries
  - Jade.ico
- reposearch
  - src
  - Referenced Libraries
  - build
  - css
  - daisydiff
  - deps
  - Empty Database
  - httpclient
  - jsse
  - ruby
  - sqlite4java-213
  - wcopies
  - build.properties
  - build.xml
  - crawl.log
  - index.log
  - lgcrawl.xml
  - lgindex.xml
  - lgservice.xml
  - service.log
  - x.xml

AbstractBitStreamIndexWriter.java

```java
public abstract class AbstractBitStreamIndexWriter implements IndexWriter {

    /** The number of documents of the collection to be indexed. */
    protected final int numberOfDocuments;
    /** The flag map. */
    public Map<Component,Coding> flags;
    /** The coding for frequencies. */
    protected Coding frequencyCoding;
    /** The coding for pointers. */
    protected Coding pointerCoding;
    /** The coding for counts. */
    protected Coding countCoding;
    /** The coding for positions. */
    protected Coding positionCoding;
    /** Whether this index contains payloads. */
    protected final boolean hasPayloads;
    /** Whether this index contains counts. */
    protected final boolean hasCounts;
    /** Whether this index contains positions. */
    protected final boolean hasPositions;

    /** The number of indexed postings (pairs term/document). */
    protected long numberOfPostings;
    /** The number of indexed occurrences. */
    protected long numberOfOccurrences;
    /** The current term. */
    protected int currentTerm;
    /** The number of bits written for frequencies. */
    public long bitsForFrequencies;
```

Problems | Javadoc | Declaration | Console | Call Hierarchy | Search | Error Log | Properties | TCP/IP Monitor | Progress

Workspace Log

type filter text

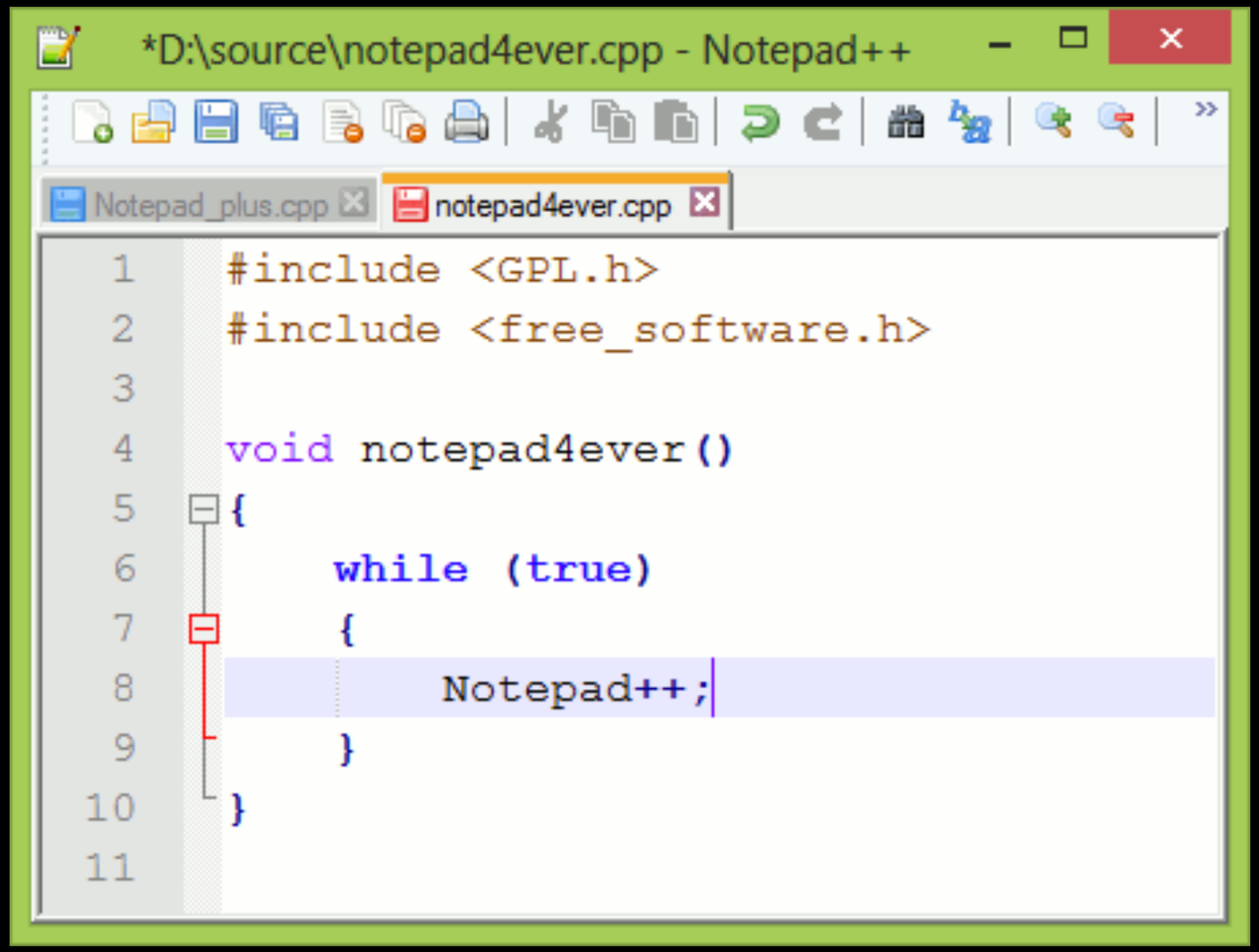
Message	Plug-in	Date
NLS unused message: osgi.nls.warnings in: org.eclipse.wst.jsdt.debug.internal.ui.message	org.eclipse.osgi	2/22/12 1:16 PM
NLS unused message: the_argument_0_is_not_valid in: org.eclipse.wst.jsdt.debug.internal	org.eclipse.osgi	2/22/12 1:16 PM
NLS unused message: suspend_thread in: org.eclipse.wst.jsdt.debug.internal.ui.messages	org.eclipse.osgi	2/22/12 1:16 PM
NLS unused message: suspend_target in: org.eclipse.wst.jsdt.debug.internal.ui.messages	org.eclipse.osgi	2/22/12 1:16 PM
NLS unused message: set_bp_hit_count in: org.eclipse.wst.jsdt.debug.internal.ui.message	org.eclipse.osgi	2/22/12 1:16 PM
NLS unused message: select_javascript_file in: org.eclipse.wst.jsdt.debug.internal.ui.mess	org.eclipse.osgi	2/22/12 1:16 PM
NLS unused message: scripts in: org.eclipse.wst.jsdt.debug.internal.ui.messages	org.eclipse.osgi	2/22/12 1:16 PM
NLS unused message: no_description_provided in: org.eclipse.wst.jsdt.debug.internal.ui.n	org.eclipse.osgi	2/22/12 1:16 PM

Writable | Smart Insert | 34 : 3

IDE (Many Features)

- Debugger
- Code Completion
- Compiler

Text Editor


*D:\source\notepad4ever.cpp - Notepad++
Notepad_plus.cpp
notepad4ever.cpp
#include <GPL.h>
#include <free_software.h>
void notepad4ever()
{
while (true)
{
Notepad++;
}
}

App.js — calculator

EXPLORER

CALCULATOR

- src
  - component
    - App.css
    - App.js
    - App.test.js
    - Button.css
    - Button.js
    - ButtonPanel.css
    - ButtonPanel.js
    - Display.css
    - Display.js
  - logic
    - calculate.js
    - calculate.test.js
    - isNumber.js
    - operate.js
  - favicon.ico
  - index.css
  - index.js
  - .gitignore
  - index.html
  - package.json
  - README.md

CODE OUTLINE

GITLENS

App.js

```
import ButtonPanel from './ButtonPanel';
import Display from './Display';
import React from 'react';
import calculate from '../logic/calculate';

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      total: null,
      next: null,
      operation: null,
    };
  }

  handleClick = (buttonName) => {
    this.setState(calculate(this.state, buttonName));
  }

  render() {
    return (
      <div className="component-app">
        <Display value={this.state.next || this.state.total || '0'} />
        <ButtonPanel clickHandler={this.handleClick} />
      </div>
    );
```

Lightweight IDE

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: zsh

```
~/Development/calculator master*
❯ ls
src/  README.md  index.html  package.json

~/Development/calculator master*
❯ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/component/App.js
```

master* 0 0 Ln 5, Col 27 Spaces: 2 UTF-8 LF JavaScript ESLint

Does it matter? Why / Why not?

- Size of project
- Faster Development time
- Less errors

03

Python 101

Var, let, const?

Name	Type	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey" : "value" , "name" : "Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating **True** or **False**

Source: https://medium.com/@shawnren527/learn-about-python-3-data-types-numbers-and-strings-76c75a917c9b

Practice Time

Script	Module	Package
A single file of python code that is **meant to be executed directly**. I.e. python my_script.py	A module is a single file of python code that is **meant to be imported**. I.e. import my_module	A package is a **collection of python modules** (Under a common namespace / directory) I.e. from my_package import my_module

Source: https://stackoverflow.com/questions/19198166/whats-the-difference-between-a-module-and-a-library-in-python

Assignment Operators

Operators

Arithmetic Operators	`+ - * ** / // %`
Assignment Operators	`= += -= *= **= /= //= %=`
Relational Operators	`== != > >= < <=`
Boolean Operators	`and or not`
Conditional Operators	`if elif else`

Loops

For Loop

Traversing a sequence

```
for i in someList:
    # do something about i
    print(i)
```

```
range(start, stop, step)
```

```
for _ in range(100):
    # do something 100 times
    pass
```

`break, continue, pass` Statements

`break`:
break out of the loop

`continue`:
skip the remaining code in the current iteration, enter the next iteration

`pass`:
syntax placeholder

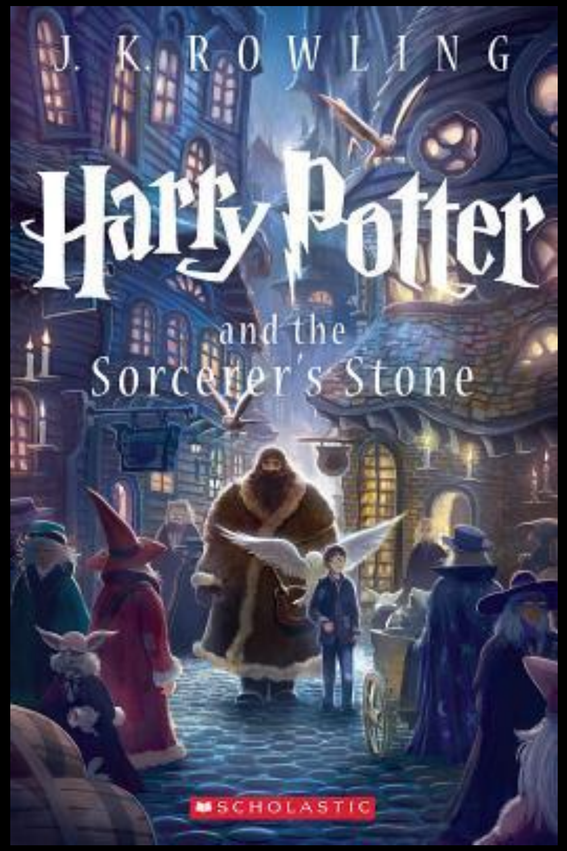
04

Put Into Practice

Make a guess, what is the most frequently used English word in this book?

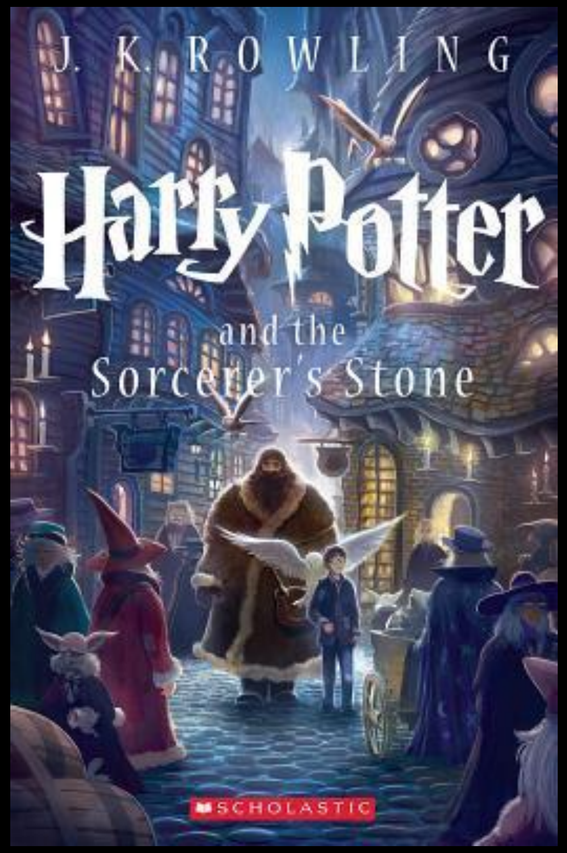
which word below is the least frequently used among these five?

Options: It Harry The Said A



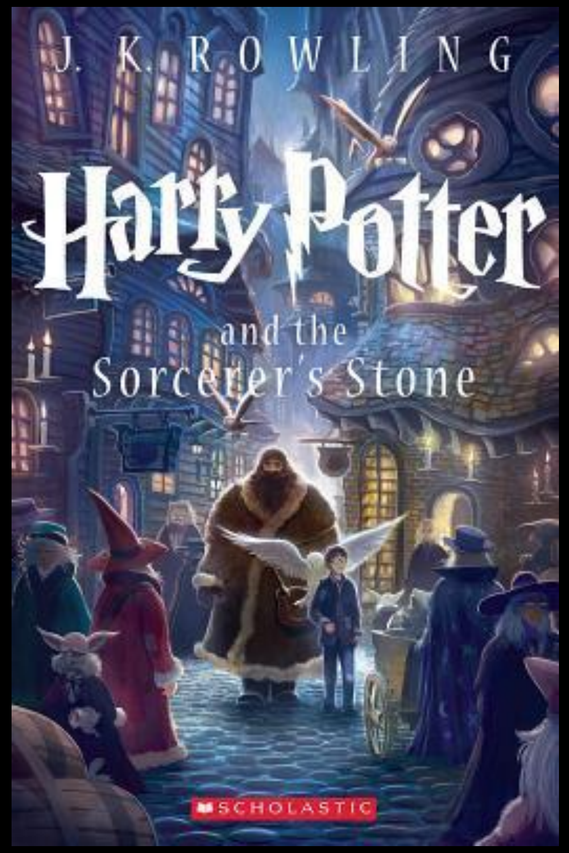
J. K. Rowling - Harry Potter and the Sorcerer's Stone

Let's find out!



J. K. Rowling - Harry Potter and the Sorcerer’s Stone

Let's find out!



You are given a sample text file called hp1.txt

You are given the book content in a text file called HP1.txt

What is the top 10 most frequently used words in this book?

J. K. Rowling - Harry Potter and the Sorcerer's Stone

For deeper analysis, check out:

https://medium.com/zareen-farooqui/harry-potter-text-analysis-4d89ffe59d5b

05

Going Further

Explore More Python Modules

Requests	is an elegant and simple HTTP library for Python.
Numpy	is the fundamental package for scientific computing with Python.
Openpyxl	is a Python library to read/write Excel
Matplotlib	is a Python 2D plotting library

Thank you

Thank you for coming

Feedback Form

https://forms.gle/LsERkgShakHK43BW8