Introduction to Programming Problem Solving

Problem Solving using Python - Week 2

Homework and Last Week Q&A

Programming Problem Solving

Programming Problem Solving

Why?

Programming Problem Solving

Why?

Programming requires multiple steps and it is challenging for novice programmers

Programming Problem Solving

Why?

Programming requires multiple steps and it is challenging for novice programmers

What?

Programming Problem Solving

Why?

Programming requires multiple steps and it is challenging for novice programmers

What?

Learning to solve programming problems using Python

Programming Problem Solving

Why?

Programming requires multiple steps and it is challenging for novice programmers

What?

Learning to solve programming problems using Python

How?

Programming Problem Solving

Why?

Programming requires multiple steps and it is challenging for novice programmers

What?

Learning to solve programming problems using Python

How?

With *Programming Problem Solving model* and tackling real problems

Learning Objectives

Learning Objectives

At the end of this lecture, you will...

Learning Objectives

At the end of this lecture, you will...

1. describe the motivation for the *Programming Problem Solving model*, its structure and its phases.

Learning Objectives

At the end of this lecture, you will...

1. describe the motivation for the *Programming Problem Solving model*, its structure and its phases.
2. apply the model to solve (simple) programming problems.

The Plan for Today

1. Introducing the model
2. Demonstrating the model on a tiny-very-easy programming problem
3. Solving a real programming problem in groups using the model

Programming Problem Solving Model

Programming Problem Solving Model

1. Reinterpret the Problem
2. Design a Solution
3. Code
4. Test
5. Debug
6. Evaluate & Reflect

Basic Example

```
Write a program that will compute
the area of a circle.
Prompt the user to enter the radius
and print a nice message
back to the user with the answer.
```

1. Reinterpret the Problem

Understanding, Interpreting, and Clarifying the Problem

What should be the **Output** for every **Input**?

Strategy:

Reading three times, explaining to myself (aloud)

The criterion of success

```
Write a program that will compute the area of a circle.
Prompt the user to enter the radius
and print a nice message back to the user with the answer.
```

Input / Output

- Input: 1
- Output: 3.14

- Input: 2.5
- Output: 19.625

- Input: 0
- Output: 0.

2. Design a Solution

2. Design a Solution

Breaking the problem into sub-problems

2. Design a Solution

Breaking the problem into sub-problems

Strategies:

1. Working at least 3 examples by hand
2. Writing the sub-problems as normal english as comments
3. Drawing the sub-problems on a paper

Writing the sub-problems as normal english as comments

```
# 1. get the radius from the user
# 2. calc the area using the formula: pi * (r**2)
# 3. print the area to the user
```

3. Code

```
# 1. get the radius from the user
# 2. calc the area using the formula: pi * (r**2)
# 3. print the area to the user
```

3. Code

```
radius = float(input("Radius: "))
# 2. calc the area using the formula: pi * (r**2)
# 3. print the area to the user
```

3. Code

```
radius = float(input("Radius: "))
area = 3.14 * (radius ** 2)
# 3. print the area to the user
```

3. Code

```
radius = float(input("Radius: "))
area = 3.14 * (radius ** 2)
print("The area of the circle is", area)
```

3. Code

```
radius = float(input("Radius: "))
area = 3.14 * (radius ** 2)
print("The area of the circle is", area)
```

```
Radius: 2.5
The area of the circle is 19.625
```

3. Code

Style

1. Meaningful Names
2. PEP8

1. Meaningful Names

- Programmers generally choose names for their variables that are **meaningful to the human readers of the program**
- ... they help the programmer document, or remember, what the variable is used for.

1. Meaningful Names

- Programmers generally choose names for their variables that are **meaningful to the human readers of the program**
- ... they help the programmer document, or remember, what the variable is used for.

Strategy

Convert English comments (from the design phase) to function names

2. PEP 8 - the Style Guide for Python Code

- One of Guido’s key insights: Code is read much more often than it is written he guidelines provided here are intended
  - to improve the readability of code
  - to make it consistent across the wide spectrum of Python code
- "Readability counts"
- **Consistency**

2. PEP 8 - the Style Guide for Python Code

- One of Guido’s key insights: Code is read much more often than it is written he guidelines provided here are intended
  - to improve the readability of code
  - to make it consistent across the wide spectrum of Python code
- "Readability counts"
- **Consistency**

PEP8 - Variables Names

```
lower_case_with_underscores
```

PEP8 - Constants Names

```
UPPER_CASE_WITH_UNDERSCORES
```

Style

```python
PI = 3.14
radius = float(input("Radius: "))
area = PI * (radius ** 2)
print("The area of the circle is", area)
```

3. Test

3. Test

Heavily depends on the **Problem Reinterpreting** and **Design** Phases

3. Test

- First
  - Input: 1
  - Output: 3.14
- Second
  - Input: 2.5
  - Output: 19.625
- Third
  - Input: 0
  - Output: 0.0

```
radius = float(input("Radius: "))
area = 3.14 * (radius ** 2)
print("The area of the circle is", area)
```

4. Debug

4. Debug

This time we didn't find a bug in our program, because:

4. Debug

This time we didn't find a bug in our program, because:

1. The program was quite simple
2. We have followed the model phases

4. Debug

This time we didn't find a bug in our program, because:

1. The program was quite simple
2. We have followed the model phases

But don't expect it to be like that, as you have already encountered in the lab.

4. Debug

This time we didn't find a bug in our program, because:

1. The program was quite simple
2. We have followed the model phases

But don't expect it to be like that, as you have already encountered in the lab.

EVERYONE has bugs in her/his code, no matter whether it is an expert or novice.

4. Debug

This time we didn't find a bug in our program, because:

1. The program was quite simple
2. We have followed the model phases

But don't expect it to be like that, as you have already encountered in the lab.

EVERYONE has bugs in her/his code, no matter whether it is an expert or novice.

What makes the difference?

4. Debug

This time we didn't find a bug in our program, because:

1. The program was quite simple
2. We have followed the model phases

But don't expect it to be like that, as you have already encountered in the lab.

EVERYONE has bugs in her/his code, no matter whether it is an expert or novice.

What makes the difference?

1. Following the model (Reinterpret the problem, Design a solution, Code, Test, ...)

4. Debug

This time we didn't find a bug in our program, because:

1. The program was quite simple
2. We have followed the model phases

But don't expect it to be like that, as you have already encountered in the lab.

EVERYONE has bugs in her/his code, no matter whether it is an expert or novice.

What makes the difference?

1. Following the model (Reinterpret the problem, Design a solution, Code, Test, ...)

2. The way of debugging

6. Evaluate & Reflect

Programming Problem Solving Model

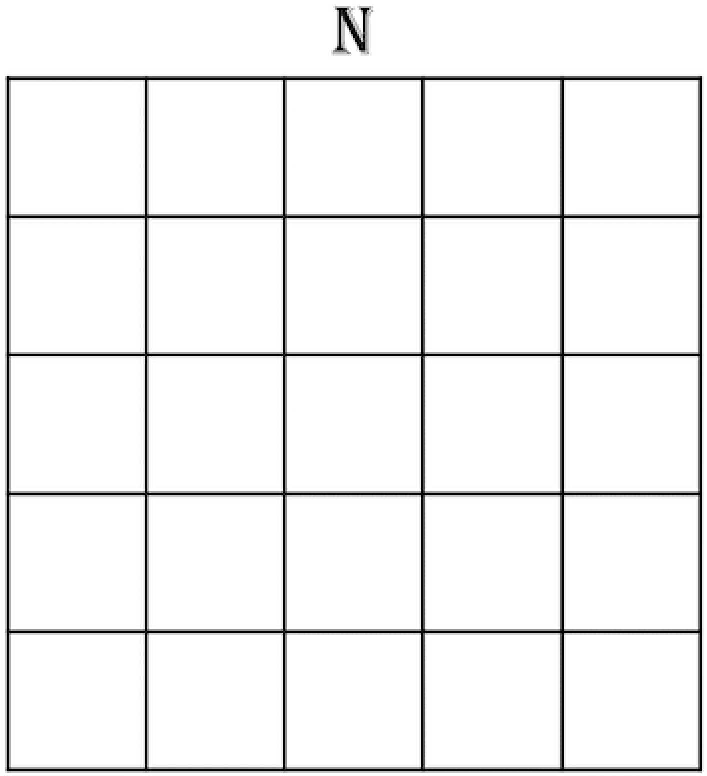
1. Reinterpret the Problem
2. Design a Solution
3. Code
4. Test
5. Debug
6. Evaluate & Reflect

Questions?

Activity - Counting Squares

The Story

1. Robot
2. Grid of squares rooms
3. Four directions: `N`, `E`, `W`, `S`



Robot Programming Language

- `Check N` (or `Check E`, or `S` or `W`)
- `Go N` (or `Check E`, or `S` or `W`)
- Basic math and remember numbers

Your Task

To program the robot to solve a specific task

i.e. with its "language"

Using the Programming Problem Solving model

First Task - Simple Square

- Suppose the robot is dropped into a square grid of rooms and starts in the north-west corner of the grid. Come up with an program that the robot can use to figure out how many rooms are in the grid.
- Is there any size grid for which your program does not work? How about on a 1-by-1, or a 1000-by-1000?
- In a 3-by-3 grid, how many times will the robot have to move through a door to run your program? How about a 4-by-4? An n-by-n?
- Assume that we want to save on robot fuel. Can you make an program that uses fewer moves for the same size grid?
- Once you have an program you think is general (works for all size squares) and efficient (uses few moves), write it on a paper. Include a simple statement about how many moves it takes, on an n-by-n grid the robot moves through 2n+5 doors or something like that.

First Task - Simple Square - Reflection

1. What is not clear in the model?
2. What is the *input* for testing? What is the *output*?
3. Is it easy to see in the code your design (i.e. composition to sub-problems)?

Second Task - Simple Rectangular

- Suppose the robot is dropped into a rectangular (not necessarily square) grid of rooms and starts in the north-west corner of the grid. Come up with an program that the robot can use to figure out how many rooms are in the grid.

- Write on a paper a program for this case, including a description of the number of moves needed for an n-by-m grid.

Second Task - Simple Rectangular - Reflection

1. What is not clear in the model?

2. Did your solution for the first task solve also the second task? Why yes/not?

3. Did you encounter bugs? How did you handle them?

Third Task - Rectangular

- Suppose the robot is dropped into a rectangular (not necessarily square) grid of rooms and might start in any arbitrary room in the grid. Come up with an program that the robot can use to figure out how many rooms are in the grid.

- Write on a paper a program for this case, including a description of the number of number of moves your robot will make for an n-by-m grid. Since this will probably depend on the starting location of the robot, just tell us the biggest number you could see (assuming the robot started in the worst possible room).

Third Task - Rectangular - Reflection

1. How did you come up with the design?

2. Did your solution for the second task solved also the third task? Why yes/not?

3. Did you encounter bugs? How did you handle them?

Wrap-up + Q&A

Problem Solving using Python - Week 1

Introduction to Programming Problem Solving

Credit for the Counting Squares activity:

Department of Computer Science at The University of Virginia

Luther Tychonievich, Mark Sherriff, and Ryan Layer

Text Readability Worked Example

Text Readability

Intuition

Winnie the Pooh is easier to understand than a linguistics paper.

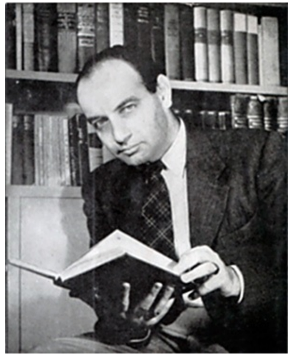
Applications

- Selecting reading materials for school children
- New York Times is designed to be read by ninth graders

Demo

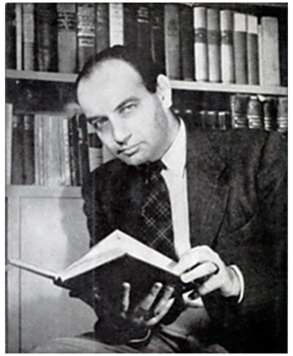
http://www.readabilityformulas.com/free-readability-formula-tests.php

Flesch Reading Ease Score

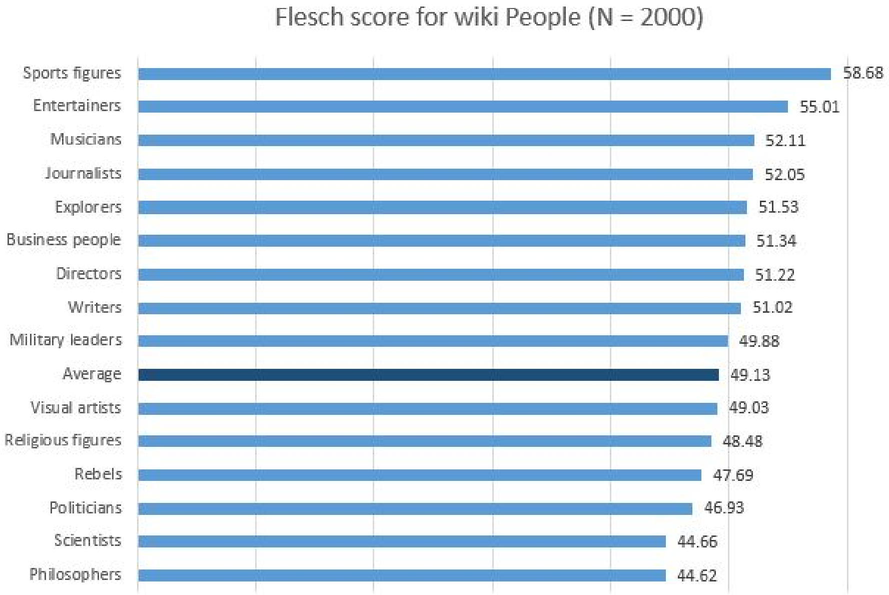


Published in **Journal of Applied Psychology** in 1948 by Rudolf Flesch

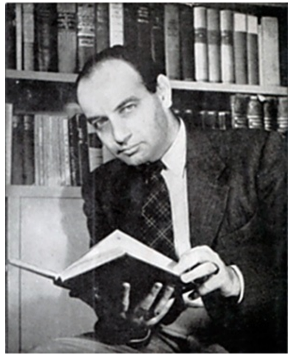
Flesch Reading Ease Score

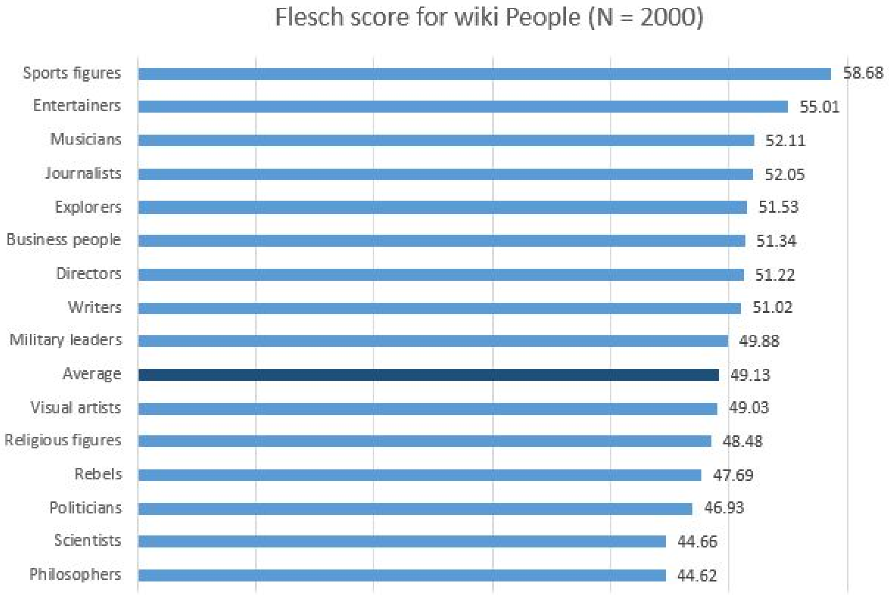


Published in **Journal of Applied Psychology** in 1948 by Rudolf Flesch



Flesch Reading Ease Score





Published in **Journal of Applied Psychology** in 1948 by Rudolf Flesch

$$ 206.835 - 84.6 \frac{total\, syllables}{total\, words } - 1.015 \frac{total\, words}{total\, sentences} $$

Implementation: Where to Begin?...

Implementation: Where to Begin?...

In:

Text: A sequence of sentences, each is a sequence of words.

Implementation: Where to Begin?...

In:

Text: A sequence of sentences, each is a sequence of words.

Out:

Score

Now in Python?

Now in Python?

In:

```
[
  ["Then", "he", "climbed", "a", "little", "further"],
  ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
]
```

Now in Python?

In:

```
[
  ["Then", "he", "climbed", "a", "little", "further"],
  ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
]
```

Out:

```
88.72514705882355
```

Let's Pretend!

```
>>> text = [
...  ["Then", "he", "climbed", "a", "little", "further"],
...  ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
... ]

>>> calculate_flesch_score(text)
```

Let's Pretend!

```
>>> text = [
...  ["Then", "he", "climbed", "a", "little", "further"],
...  ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
... ]

>>> calculate_flesch_score(text)
```

What's the output?

Let's Pretend!

```
>>> text = [
...  ["Then", "he", "climbed", "a", "little", "further"],
...  ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
... ]

>>> calculate_flesch_score(text)
```

What's the output?

```
88.72514705882355
```

Fulfilling Promises

We owe ourselves a function! Let's write one!

Fulfilling Promises

We owe ourselves a function! Let's write one!

```python
def calculate_flesch_score(text):
    pass
```

Fulfilling Promises

We owe ourselves a function! Let's write one!

```python
def calculate_flesch_score(text):
    pass
```

We're in luck, there's a formula!

```python
def calculate_flesch_score(text):
    return (
      206.835
      - 84.6 * n_syllables / n_words
      - 1.015 * n_words / n_sents
    )
```

Not so Fast...

Where do we get `n_syllables`, `n_words`, `n_sents` from?

Not so Fast...

Where do we get `n_syllables`, `n_words`, `n_sents` from?

I'm impatient to get my code to run, let's set them all to constants!

```python
def calculate_flesch_score(text):
    n_syllables = 0
    n_words = 15
    n_sents = 2

    return (
      206.835
      - 84.6 * n_syllables / n_words
      - 1.015 * n_words / n_sents
    )
```

Avoiding the Hard Stuff

How **do** you count syllables in text?

Avoiding the Hard Stuff

How **do** you count syllables in text?

- Do multiple vowels count as one syllable?

Avoiding the Hard Stuff

How **do** you count syllables in text?

- Do multiple vowels count as one syllable?
- What about word-final "e", like in "plane"?

Avoiding the Hard Stuff

How **do** you count syllables in text?

- Do multiple vowels count as one syllable?
- What about word-final "e", like in "plane"?
- What if a word contains no vowels at all, but still has syllables?

Avoiding the Hard Stuff

How **do** you count syllables in text?

- Do multiple vowels count as one syllable?
- What about word-final "e", like in "plane"?
- What if a word contains no vowels at all, but still has syllables?

My brain says "This is starting to sound complicated"...

Avoiding the Hard Stuff

How **do** you count syllables in text?

- Do multiple vowels count as one syllable?
- What about word-final "e", like in "plane"?
- What if a word contains no vowels at all, but still has syllables?

My brain says "This is starting to sound complicated"...

... so let's skip it for now! We'll come back to it later :)

Low-Hanging Fruit

In fact, I'm so lazy that I'll skip counting words too...

Low-Hanging Fruit

In fact, I'm so lazy that I'll skip counting words too...

How many sentences?

```
>>> text = [
...   ["Then", "he", "climbed", "a", "little", "further"],
...   ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
... ]
```

Low-Hanging Fruit

In fact, I'm so lazy that I'll skip counting words too...

How many sentences?

```
>>> text = [
...   ["Then", "he", "climbed", "a", "little", "further"],
...   ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
... ]
```

How do we ask this in Python?

Low-Hanging Fruit

In fact, I'm so lazy that I'll skip counting words too...

How many sentences?

```
>>> text = [
...   ["Then", "he", "climbed", "a", "little", "further"],
...   ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
... ]
```

How do we ask this in Python?

```
>>> len(text)
2
```

Tweak the Function!

```
def calculate_flesch_score(text):
    n_syllables = 0
    n_words = 15
    n_sents = len(text)

    return (
      206.835
      - 84.6 * n_syllables / n_words
      - 1.015 * n_words / n_sents
    )
```

Counting Words

How many words?

```
>>> text = [
...  ["Then", "he", "climbed", "a", "little", "further"],
...  ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
... ]
```

Counting Words

How many words?

```
>>> text = [
...   ["Then", "he", "climbed", "a", "little", "further"],
...   ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
... ]
```

In Python?

Counting Words

How many words?

```
>>> text = [
...   ["Then", "he", "climbed", "a", "little", "further"],
...   ["By" "that" "time" "he" "had" "thought" "of" "another" "song"]
... ]
```

In Python?

```
n_words = 0
for sentence in text:
  for word in sentence:
    n_words = n_words + 1
```

Tweak the Function!

```python
def calculate_flesch_score(text):
    n_syllables = 0

    n_words = 0
    for sentence in text:
        for word in sentence:
            n_words = n_words + 1

    n_sents = len(text)
    return (
      206.835
      - 84.6 * n_syllables / n_words
      - 1.015 * n_words / n_sents
    )
```

Finally Ready to Face The Monster

Kinda... What if we only count the number of vowels in a word?

Finally Ready to Face The Monster

Kinda... What if we only count the number of vowels in a word?

Input - Output

```
"abracadabra" - 5
```

Finally Ready to Face The Monster

Kinda... What if we only count the number of vowels in a word?

Input - Output

`"abracadabra" - 5`

Let's Pretend Again

```
>>> count_syllables("abracadabra")
5
```

No Formula... What to do?

No Formula... What to do?

In Plain English

- For every character in a word, if it is a vowel, add 1 to a counter
- Return the counter

No Formula... What to do?

In Plain English

- For every character in a word, if it is a vowel, add 1 to a counter
- Return the counter

In Python

```python
def count_syllables(word):
    vowels = "aeiou"
    n_vowels = 0
    for char in word:
        if char in vowels:
            n_vowels = n_vowels + 1
    return n_vowels
```

Tweak the Function!

```
def calculate_flesch_score(text):
    n_syllables = 0
    for sentence in text:
        for word in sentence:
            n_syllables = n_syllables + count_syllables(word)

    n_words = 0
    for sentence in text:
        for word in sentence:
            n_words = n_words + 1

    n_sents = len(text)

    return (
      206.835
      - 84.6 * n_syllables / n_words
      - 1.015 * n_words / n_sents
    )
```

Let's Clean Up

```
def calculate_flesch_score(text):
    n_syllables = 0
    n_words = 0
    for sentence in text:
        for word in sentence:
            n_syllables = n_syllables + count_syllables(word)
            n_words = n_words + 1

    n_sents = len(text)

    return (
      206.835
      - 84.6 * n_syllables / n_words
      - 1.015 * n_words / n_sents
    )
```

Play Time!!

Try out our new function with different inputs.

Because I'm dumb, I will keep track of what I have tried.

Acknowledgements

Michelle Craig, University of Toronto

Programming Problem Solving Model

1. Reinterpret the Problem
2. Design a Solution
3. Code
4. Test
5. Debug
6. Evaluate & Reflect

Wrap-up + Q&A

Problem Solving using Python - Week 1

Introduction to Programming Problem Solving