# strings

March 22, 2022

[3]:
```
# This is a cell
# It can contain code or formatted text (Markdown)

2+2
```

[3]:
```
4
```

[3]:
```
total = 2 + 2
```

[5]:
```
# But what happens if we restart the kernel?
print(total)
```

```
4
```

## 1 Heading 1

### 1.1 heading 2

#### 1.1.1 heading 3

Here is a list of bullets * one * two * three

This works too: - item1 - item2 - item3

Here is **bold**, *italics*

Here is a link to google.

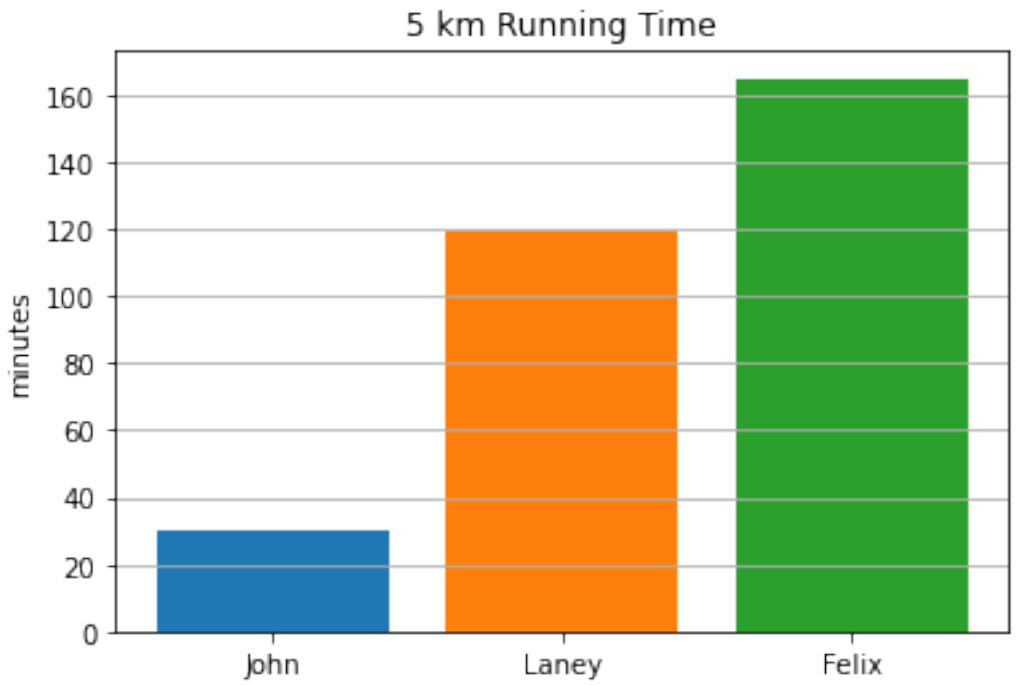
A simple table:

| Runner | 5km |
|---|---|
| John | 29m 59s |
| Laney | 1h 59m 35s |
| Felix | 2h 45m 18s |

You can even insert an image:

## 2 Embedding a simple visualization

[10]:
```
import matplotlib.pyplot as plt
plt.bar('John', 30)
plt.bar('Laney', 119)
plt.bar('Felix', 165)
plt.grid(axis='y')
plt.ylabel('minutes')
plt.title('5 km Running Time')
plt.show()
```




### 2.0.1 Let's learn about strings

[12]:
```
# Strings are immutable - they can't be changed

"hello"[1]
```

[12]:
```
'e'
```

[13]:
```
"hello"[1] = 'u'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [13], in <module>
----> 1 "hello"[1] = 'u'

TypeError: 'str' object does not support item assignment
```

[14]:
```
"hello".replace("e", "u")
```

[14]:
```
'hullo'
```

[21]:
```
msg = "hello"
```

```
msg.replace("e", "u") # This produced a NEW string that wasn't assigned to␣
 ↪anything
msg # The original string is unchanged

# but this makes it permanent
msg = msg.replace("e", "u")
msg
```

[21]:
```
'hullo'
```

[22]:
```
# Transformations
msg.upper()
msg.lower()
msg.capitalize()
```

[22]:
```
'Hullo'
```

[23]:
```
# slicing

msg[1:3]
```

[23]:
```
'ul'
```

[24]:
```
# Replacing multiple occurrences
msg.replace("l", "L")
```

[24]:
```
'huLLo'
```

[25]:
```
msg.replace("ll", "llllll")
```

[25]:
```
'hullllllo'
```

[26]:
```
# remove L's
msg.replace("l", "")
```

[26]:
```
'huo'
```

[32]:
```
# Removing punctuation - method 1:
# simpler but you might miss some
phrase = "Isn't this easy? Fun, easy, and practical!!"
punctuation = "'?,!.;"
for p in punctuation:
    phrase = phrase.replace(p,"")
phrase
```

[32]:
```
'Isnt this easy Fun easy and practical'
```

[34]:
```
# method 2
# better coverage but a bit more verbose
phrase = "Isn't this easy? Fun, easy, and practical!!"
new_phrase = ""
for letter in phrase:
    if letter.isalpha() or letter==' ':
        new_phrase += letter
new_phrase
```

[34]: 'Isnt this easy Fun easy and practical'

[39]:
```
# method3 - complicated / not obvious but fast and concise
import string
phrase.translate(str.maketrans('', '', string.punctuation))
```

[39]: 'Isnt this easy Fun easy and practical'

[53]:
```
## Formatted printing - a few examples

pi = 3.14159
name = 'Joe'

print(f"{name}'s favorite number is {pi}")
print(f"{name}'s favorite number is {pi:.2f}")
print(f"{name}'s favorite number is {pi:.3f}")
print(f"{name}'s favorite number is {pi:.4f}")

print(f"{'A'})
```

```
Joe's favorite number is 3.14159
Joe's favorite number is 3.14
Joe's favorite number is 3.142
Joe's favorite number is 3.1416
Two plus two is 4
```

[ ]: