ICT LITERACY AND COMPETENCY DEVELOPMENT BUREAU

python

PROGRAMMING ESSENTIALS WEBINAR

DAY 5

Contents

substrings

Substrings in Python

- Substrings are parts of a greater string, derived for further processing.
- A.k.a. *string slicing*
- In Python, this can be done using the following format:
  - `>>> string[start:stop:step]`
- But commonly the following format is widely in practice amongst industry practitioners:
  - `>>> string[start:stop]`

Substrings in Python

```
>>> a = "hello world"
>>> a[0:2]
'he'
>>> a[2:5]
'llo'
```

Substrings in Python

```
>>> a = "hello world"
>>> a[0:]
'hello world'
>>> a[:6]
'hello '
```

Substrings in Python

```
>>> a = "hello world"
>>> a[:-1]
'hello worl'
>>> a[:]
'hello world'
```

Substrings in Python

```
>>> site = "https://www.google.com"
>>> tld = site[-3:]
>>> sch = site.split('://')[0]
>>>
```

lists and strings

Lists and Strings

- The pythonic list and string data structures are often used in tandem especially when dealing with string handling / string manipulation.
- Strings can be converted into lists.
- Lists can be converted to strings.

Strings to lists

```
friends = "Jo JoJo Joe"
friends_list = friends.split()
print(friends_list)
# ['Jo', 'JoJo', 'Joe']
friends_list = friends.split('J')
# ['', 'o ', 'o', 'o ', 'oe']
```

List to string

```
friends_list = ['Jo', 'JoJo', 'Joe']
friends = ''.join(friends_list)
print(friends)
# JoJoJoJoe
friends = ' '.join(friends_list)
# Jo JoJo Joe
```

Final notes on lists and strings

- Use `.split()` to convert space-separated strings into lists.
- `.split()` can also be used to convert a custom-delimiter-separated string into respective lists.

- Use `.join()` to combine elements within a list into a string.
- Use `' '.join()` to separate each element of the list – on its way to being a combined string.

list comprehension

List comprehension

- List comprehension is a *Pythonic* way of expressing the generation of lists using for loops – in <u>**one line of code.**</u>

List Comprehensions

```
numbers = [0,1,2,3,4]
times_two = []
for number in numbers:
    times_two.append(number * 2)
print(times_two)
```

```
numbers = [0,1,2,3,4]
times_two = [number * 2 for number in numbers]
print(times_two)
```

functions

Nature of Functions

- Blocks of code that are reusable in any part of your script.
- May or may not accept data (parameters), and may or may not return any data.
- Follows the DRY principle! (Don't repeat yourself!)
- Ideally: written once – reused multiple times.

Defining and calling functions

```
def function_name():
    # code block here


function_name()
```

Parameters and Return

```
def square(number):
    answer = number * number
    return answer

sq = square(2)
print(sq)
```

Default Value Parameters

```
def print_num(number=1):
    print(number)

sq = input()
if sq != '':
    sq = print_num(int(sq))
else:
    sq = print_num()
```

Returning Multiple Data

```
def return_msg(number):
    status = "isn't"
    if number % 2 == 0:
        status = "is"
    return number, status
num = int(input("input number")
num2, stat = return_msg(num)
print(f"{num2} {stat} divisible by two")
```

Returning Multiple Data

```
def return_msg(number):
    status = "isn't"
    if number % 2 == 0:
        status = "is"
    return number, status
num = int(input("input number")
num2, stat = return_msg(num)
print(f"{num2} {stat} divisible by two")
```