GOVERNMENT OF WESTERN AUSTRALIA

TAFE WA South Metropolitan

String Methods

(Session 4)

Review (Assignment Statements)

- Assigning a **string** to a variable. For example:

```
str_text = "Hello"  # a variable is created by assigning a value to it
print(str_text)
```

- Assigning an **integer** to a variable. For example:

```
num_int = 5  # creates a variable named num_int and assigns an integer value
print(num_int)
```

- Assigning a **float** to a variable. For example:

```
num_float= 20.5   # assigns a float value equal to 20.5
print(num_float)
```

```
Hello
5
20.5
```

- An **expression** is a combination of variables and operators. For example:

```
print(num_float * num_float)
```

Review (String Indexing)

```
str_text = "Strings are Arrays"
# Prints complete content of string variable
print(str_text)
print(str_text[:])    # also  prints all characters – inline comment

# Prints first character of the string
print(str_text[0])
# Prints the last character of the string
print(str_text[-1])

# Prints characters starting from 2nd to 5th
print(str_text[1:5])
# Prints string starting from 3rd character
print(str_text[2:])
#prints the last six characters
print(str_text[-6:])
```

```
Strings are Arrays
Strings are Arrays
S
s
trin
rings are Arrays
Arrays
```

Overview

- String Operators
- Python Standard Library
- ASCII and Unicode
- String Methods

String Operators

- Add (+) and Multiply (*)
- **String Concatenation** - both arguments are strings, the + operator produces a new string. For example:

```
print("When you create a string, " + "it is immutable.")
print("It can't " + "be " + "changed,")
print("but you can " + "concatenate " + "one string " + "with another")
```

```
When you create a string, it is immutable.
It can't be changed,
but you can concatenate one string with another
```

String Operators (cont.)

- Strings are **immutable,** you can't change a string's value. For example:

```
str_text = "you can't overwrite me "
str_text[0] = "Y"  # attempt to modify the context of the string
```

```
    str_text[0] = "Y"  # attempt to modify the context of the string
TypeError: 'str' object does not support item assignment
```

- **Replication** is fine (arguments are a number and string, the * operator produces a new string). For example:

```
print(str_text * 2)
print(3 * "Hello ")
```

```
you can't overwrite me you can't overwrite me
Hello Hello Hello
```

Length of a String

- The **len()** function returns the number of characters in a string

Sytax: *len(string). Examples:*

```
str_text = "len() function is a built-in function."
print(str_text)
print(len(str_text))
# assigning a string to a variable is fine
str_text = "Any string can be empty"
print(str_text)
str_text = ""
print("Empty string?...its lenght is: ", len(str_text))
```

```
len() function is a built-in function.
38
Any string can be empty
Empty string?...its lenght is:  0
```

Python Standard Library

- Built-in functions are loaded automatically and always available
  - For instance, print(), int() or len() functions are integrated with the Python shell.
- In addition to the limited numbers of built-in functions, there are a large numbers of functions available as a part of Python's Standard Library
- They are bundled with Python distribution
- To display a list of available modules, use the following command:

```
help("modules")
```

```
Please wait a moment while I gather a list of all available modules...

pygame 2.1.2 (SDL 2.0.18, Python 3.9.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
__future__          asyncio             html                search
__main__            asyncore            http                searchbase
_abc                atexit              hyperparser         searchengine
_aix_support        audioop             idle                secrets
_ast                autocomplete        idle_test           select
_asyncio            autocomplete_w      idlelib             selectors
_bisect             autoexpand          imaplib             session2
_blake2             base64              imghdr              session3
_bootlocale         bdb                 imp                 session4
_bootsubprocess     binascii            importlib           setuptools
_bz2                binhex              inspect             shelve
```

Code Point Representation

- Computers store characters as numbers
- **Code point** – is a number representing a particular character
- **ASCII** stands for American Standard Code for Information Interchange
  - mainly used to encode the Latin alphabet, characters occupy 1 byte
- To see the list of printable character in ASCII, you need to call *string.printable* from the **string** module
  - To import module (containing other functions, classes and and variables) use the **import** statement.
  - Check out the Python documentation on the string module

Activity 1: printing ASCII letters

```
import string     # imports the string module
# prints ASCII letters and characters considered printable
print("ASCII letters are: ", string.ascii_letters)
print("Printable characters are: ", string.printable)
```

```
ASCII letters are:  abcdefghijklmnopqrstuvwxyzABCDEFGHIJK
LMNOPQRSTUVWXYZ
Printable characters are:  0123456789abcdefghijklmnopqrst
uvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()*+,-./:;<=>?@[\]
^_`{|}~
```

Code Point Representation (cont.)

- **UNICODE** (International Coding Standard, (https://www.unicode.org/)) is a character scheme used for encoding different languages
- For example, encoding standard such as UTF-8
  - UTF - stands for Unicode Transformation Format
  - **UTF-8** (default encoding in Python) uses 1, 2, 3, or 4 bytes
  - The first 128 characters of Unicode, correspondent with ASCII (1 byte)
  - Non-English characters need more bytes. For instance, CJK(China-Japan-Korea) ideographs occupy 24 bits(4 bytes)

Unicode Standard

- Python does not have a character type, but you can use single strings (i.e., strings with length 1). For example:

```
char_one = "A"
char_two = "b"
print(char_one, char_two)
```

```
A b
```

- A **code point** is an integer in range from 0 to 1.1 million values.
- For example, the ASCII value of the letter A is **65.** Note that ASCII only encodes 128 characters.

Code Point example	Character	Description
0	NULL	Null character
..	..	…
32	space	Space character
…	…	…
49	1	Printable character 1
…	…	…
65	A	Latin letter A
…	…	
90	Z	Latin letter Z
…	…	…
97	a	Latin small letter a
…	…	…
98	b	Latin small letter b
…	…	…

ASCII Code

- To get the ASCII code of a character, use the ord() function. For example:

```
print(ord("1"))
print(ord("A"))
print(ord("Z"))
print(ord("a"))
```

```
49
65
90
97
```

- To get the character encoded by an ASCII code number, use the chr() function. For example:

```
print (chr(49))
Print (chr(65))
```

```
1
A
```

Unicode Characters

- From **0 – to 127** - ASCII characters, such as ‘A’, ‘9’, and ‘z’

```
characters = list(map(chr, range(0, 127)))
print(characters)
```

- From **128 to 2047** - Most Latin and Greek alphabets, such as 'Ȼ', 'ȼ', 'Ƚ’, 'Ǿ', 'ǿ’, 'ʊ', 'ʋ', 'ʌ’, 'ɷ', 'ɸ’, ‘©’, 'Ś', 'ś’, 'Ū', 'ū',

```
characters = list(map(chr, range(128, 2047)))
print(characters)
```

- From **2048 to 655535** – Additional characters, such as 'コ', '오', 'ՙ̲', 'ዓ', 'F', 'ㅋ'

- From **65536 to 111411** – other characters, such as ’ 😓 ’, ’ 🂲’

Activity 2: Decoding Unicode Characters

- How to get the code point of a character
- **ord()** function returns an integer representing the Unicode number

```
print(ord('α '))
print(chr(945))
print(ord('ʊ'))
print(ord('ʌ'))
print(ord('©'))
print(ord('😢'))
```

```
945
α
650
652
169
128531
```

Activity 3: the **in** and **not in** Operators

- “in” checks whether as specific value exists
- “not in” does the opposite job.
- Look at the example program below

```
letters = "abcdefgABDZ"
print("f" in letters)
print("F" in letters)
print("5" in letters)
print("gx" not in letters)
print("AB" in letters)
```

```
True
False
False
True
True
```

Activity 4: Functions min() and max()

- The function **min()** finds the minimum element of the sequence
- The function **max()** finds the maximum element of the sequence

```
letters = "abcdefgABDZ"
print(min(letters))      # A is 65, and a is 97
print(min(letters[6:])) # starting with index 6
print(max(letters))
print(min(letters[5:7]))  # from index 5 to 7
```

```
A
A
g
f
```

The index() methods

- Finds the first element. For example:

```
# Demonstrating the index() method:
print("abcdefgABDZ".index("b"))
print("abcdefgABDZ".index("Z"))
letters = "abcdefgABDZ"
print(letters.index("c"))
```

```
1
10
2
```

- The element searched must occur in the sequence, otherwise, its absence will cause an exception. For example:

```
print(letters.index("x"))
```

```
print(letters.index("x"))
ValueError: substring not found
```

Other Methods

- The **count ()** method counts all occurrences
- The **capitalize ()** method converts the first character to upper-case
- The **find()** method looks for a substring
- The **isalpha()** method looks at letters only
- The **isdigit()** method looks at digits only
- The **islower()** method looks at lower-case letters only
- The **isupper()** method looks at upper-case letter only

Activity 5: Other Methods

```
letters = "aaabcdefgABDZ"
print(letters.count("a"))  #counts all occurrences
print(letters.capitalize())  #capitalizes the first char
print(letters.find("AB"))  # finds index of AB
print(letters.isalpha())  # only letters no digits
print(letters.isdigit())  # only digits
print(letters.islower()) # all lower-case letters only
print(letters.isupper())  # all upper-case letters only
```

```
3
Aaabcdefgabdz
9
True
False
False
False
```

Activity 6: Other Methods (cont.)

- The **replace ()** method – returns a copy of string with replacement
- Syntax*: string.replace (oldvalue, newvalue, count)*

```
print("Python two".replace("two", "three"))
str_text = "Word of python"
print("Before replacement: ", str_text)
new_text = str_text.replace("python", "Python")
print("After:", new_text)
str_text = "two two Python"
print("New text:", str_text)
new_text = str_text.replace("two", "three", 2)
print(new_text)
```

```
Python three
Before replacement:  Word of python
After: Word of Python
New text: two two Python
three three Python
```

Strings Methods (cont)

- Some of methods offered by strings are:
  - strip()
  - title()
  - join()
  - center()
  - isspace()
  - startwith()

Go to https://www.w3schools.com/python/python_ref_string.asp and explore them...

Questions?