Introduction to

PYTHON PROGRAMING

INSTALLATION: WINDOWS

1. DOWNLOAD

Download: Go to the official Anaconda website and download the graphical installer for Windows (64-bit is most common).

2. RUN

Double-click the downloaded .exe file.

3. Follow the Prompts:

Click "Next," agree to the license terms, and select "Just Me" for the installation type unless you need it for all users on the computer.

4. Choose a Location:

Accept the default installation location or specify a custom one.

5. Advanced Options:

Leave the "Add Anaconda to my PATH environment variable" option unchecked unless you know what you are doing, as it can cause conflicts with other software. The installer will prompt you to register Anaconda as your default Python.

6. Complete Installation:

Click "Install" and wait for the process to finish. Click "Finish" to complete the setup.

INSTALLATION: LINUX

1. DOWNLOAD

CHOOSE A DISTRIBUTION:
A.https://repo.anaconda.com/archive/
B. wget https://repo.anaconda.com/archive/Anaconda3-202X.XX-Linux-x86_64.sh

2. RUN

bash Anaconda3-202X.XX-Linux-x86_64.sh

3. Follow the Prompts:

Read and accept the license agreement by typing yes.

4. Choose a Location:

Accept the default installation location or specify a custom one.

5. Initialize Anaconda:

When prompted to initialize Anaconda, type yes. This will modify your .bashrc or .zshrc file to ensure the conda command works.

6. Reload Terminal:

Close and reopen the terminal, or run source ~/.bashrc (or source ~/.zshrc) to apply the changes.

INSTALLATION: UNIX (MAC)

1. DOWNLOAD

CHOOSE A DISTRIBUTION:
A.https://repo.anaconda.com/archive/
B. wget https://repo.anaconda.com/archive/Anaconda3-202X.XX-Linux-x86_64.sh

2. RUN

bash Anaconda3-202X.XX-Linux-x86_64.sh

3. Follow the Prompts:

Read and accept the license agreement by typing yes.

4. Choose a Location:

Accept the default installation location or specify a custom one.

5. Initialize Anaconda:

When prompted to initialize Anaconda, type yes. This will modify your .bashrc or .zshrc file to ensure the conda command works.

6. Reload Terminal:

Close and reopen the terminal, or run source ~/.bashrc (or source ~/.zshrc) to apply the changes.

CREATING CONDA ENVIRONMENT: WINDOWS

- Press START
- Search: "Anaconda Prompt"
- Click on it to open
- Check Conda Installation: conda --version
- Create a New Conda Environment: conda create --name myenv python=3.10
- Activate the Environment: conda activate myenv
- Verify Python Version: python --version
- Deactivate the Environment: conda deactivate

CREATING CONDA ENVIRONMENT: LINUX/MAC

- Open Terminal: Press Ctrl + Alt + T
- Check Conda Installation: conda --version
- Create a New Conda Environment:
  conda create --name myenv python=3.10
- Activate the Environment: conda activate myenv
- Verify Python Version: python --version
- Deactivate the Environment: conda deactivate

WHAT IS A STRING?

1. Collections of text in Python are called strings.
2. Special functions called string methods are used to manipulate strings.
3. There are string methods for changing a string from lowercase to uppercase, removing whitespace from the beginning or end of a string, or replacing parts of a string with different text, etc.
4. String methods: Python commands to manipulate strings

WHAT IS A STRING?

1. Strings are one of the fundamental Python data types.
2. The term data type refers to what kind of data a value represents.
3. Strings are used to represent text.

```
type("HELLO WORLD!")
```

```
str
```

PROPERTIES OF A STRING?

1. Strings contain characters, which are individual letters or symbols.
2. Strings have a length, which is the number of characters contained in the string.
3. Characters in a string appear in a sequence, meaning each character has a numbered position in the string.

STRING LITERAL

1. Whenever you create a string by surrounding text with quotation marks, the string is called a string literal. The name indicates that the string is literally (explicitly) written out in your code.
2. The quotes surrounding a string are called delimiters because they tell Python where a string begins and where it ends.
3. When one type of quotes is used as the delimiter, the other type of quote can be used inside of the string.

```
var1 = "We're #1!"
print(var1)
```

```
We're #1!
```

```
var2 = 'I said, "Put it over by the llama."'
print(var2)
```

```
I said, "Put it over by the llama."
```

```
var3 = "I said, 'Put it over by the llama.'"
print(var3)
```

```
I said, 'Put it over by the llama.'
```

```
var4 = "I said, "Put it over by the llama.""
print(var4)
```

```
  Cell In[9], line 1
    var4 = "I said, "Put it over by the llama.""
                    ^
SyntaxError: invalid syntax
```

BEST PRACTICES

- A common pet peeve among programmers is the use of mixed quotes as delimiters. When you work on a project, it's a good idea to use only single quotes or only double quotes to delimit every string.

- Keep in mind that there isn't really a right or wrong choice! The goal is to be consistent, because consistency helps make your code easier to read and understand.

DETERMINE THE LENGHT OF A STRING

The number of characters contained in a string, including spaces, is called the **length of the string**.

To determine a string's length, you use Python's built-in **len()** function.

```
var1 = "We're #1!"
len(var1)
```

```
9
```

```
var2 = 'I said, "Put it over by the llama."'
len(var2)
```

```
35
```

MULTILINE STRINGS

- The PEP 8 style guide recommends that each line of Python code contain no more than 79 characters—including spaces.
- PEP 8's 79-character line-length is recommended because, among other things, it makes it easier to read two files side-by-side. However, many Python programmers believe forcing each line to be at most 79 characters sometimes makes code harder to read.
- To deal with long strings, you can break the string up across multiple lines into a multiline string.
- Break the string up across multiple lines and put a backslash (\) at the end of all but the last line.

MULTILINE STRINGS

- Multiline strings can also be created using triple quotes as delimiters (""" or ''').

- Triple-quoted strings preserve whitespace. This means that running print(paragraph) displays the string on multiple lines just like it is in the string literal, including newlines.

- Triple-quoted strings have a special purpose in Python. They are used to document code. You'll often find them at the top of a .py with a description of the code's purpose. They are also used to document custom functions.

- When used to document code, triple-quoted strings are called docstrings.

```
paragraph= """This planet has - or rather had - a problem, which was
this: most of the people living on it were unhappy for pretty much
of the time. Many solutions were suggested for this problem, but
most of these were largely concerned with the movements of small
green pieces of paper, which is odd because on the whole it wasn't
the small green pieces of paper that were unhappy."""

print(paragraph)
```

```
This planet has - or rather had - a problem, which was
this: most of the people living on it were unhappy for pretty much
of the time. Many solutions were suggested for this problem, but
most of these were largely concerned with the movements of small
green pieces of paper, which is odd because on the whole it wasn't
the small green pieces of paper that were unhappy.
```

CONCATENATION

1. Two strings can be combined, or concatenated, using the+ operator:

```
string1= "abra"
string2= "cadabra"
magic_string= string1 + string2
magic_string
```

```
'abracadabra'
```

```
string1= "abra"
string2= "cadabra"
magic_string= string1+ " " + string2
magic_string
```

```
'abra cadabra'
```

STRING INDEXING

1. Each character in a string has a numbered position called an “index”.
2. You can access the character at the Nth position by putting the number N in between two brackets [N].
3. In Python—and most other programming languages—counting always starts at zero.
4. Forgetting that counting starts with zero and trying to access the first character in a string with the index 1 results in an off-by-one error.

```
string1= "abra"
string2= "cadabra"
magic_string= string1+ " " + string2
magic_string
```

```
'abra cadabra'
```

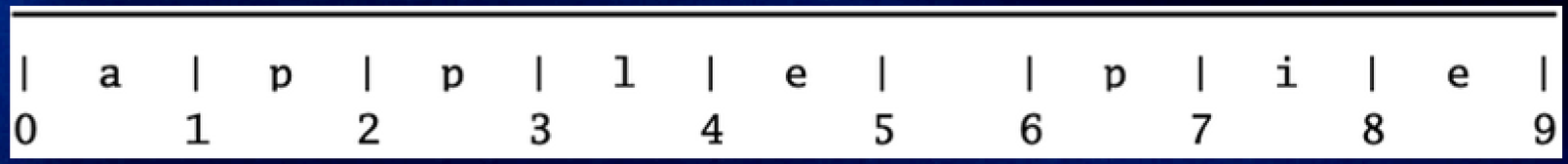
```
magic_string[0]
```

```
'a'
```

```
magic_string[4]
```

```
' '
```

STRING SLICING



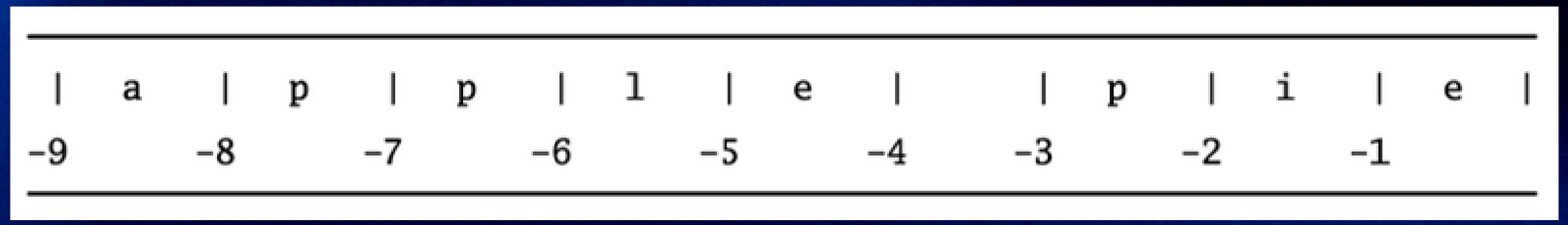

```
string1 = "apple"
string2 = "pie"
flavor = string1+ " " + string2
flavor
```

```
'apple pie'
```

```
flavor[0:4]
```

```
'appl'
```

STRING SLICING




```
string1 = "apple"
string2 = "pie"
flavor = string1+ " " + string2
flavor
```

```
'apple pie'
```

```
flavor[-9:-5]
```

```
'appl'
```

NOTE:

Strings are immutable, which means that you can't change them once you've created them.

```
word= "gold"
word[0]= "f"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[38], line 2
      1 word= "gold"
----> 2 word[0]= "f"

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

```
word= "goal"
word= "f" + word[1:]
word
```

```
'foal'
```

REVIEW QUESTIONS:

- Create a string and print its length using the **len()** function.
- Create two strings, concatenate them, and print the resulting string.
- Create two strings and use concatenation to add a space inbetween them. Then print the result.
- Print the string "fox" by using slice notation on the string "The quick brown fox jumped over the lazy dog. #1234567890!" to specify the correct range of characters.

1..lower() converts all string to lower case.
2.upper() converts all string to upper case.

CONVERTING STRING CASE

```
word= "GoLd"
new_word= word.lower()
new_word
```

```
'gold'
```

```
word= "GoLd"
new_word= word.upper()
new_word
```

```
'GOLD'
```

REMOVING WHITESPACE FROM A STRING

- .rstrip() removes whitespace from the right side of a string.
- .lstrip() removes whitespace from the left side of the string
- .strip() removes whitespace from the both sides of the string

```
word = ' royals '
word.rstrip()
```

```
' royals'
```

```
word = ' royals '
word.lstrip()
```

```
'royals '
```

```
word = ' royals '
word.strip()
```

```
'royals'
```

DETERMINE IF A STRING STARTS OR ENDS WITH A PARTICULAR STRING

- .startswith() and .endswith()

```
word = 'Dictionary'
word.startswith('Dic')
```

```
True
```

```
word = 'Dictionary'
word.startswith('dic')
```

```
False
```

```
word = 'Dictionary'
word.endswith('ry')
```

```
True
```

```
word = 'Dictionary'
word.endswith('RY')
```

```
False
```

REVIEW QUESTIONS:

1. Write a script that converts the following strings to lowercase: "Animals", "Badger", "Honey Bee", "Honeybadger". Print each lowercase string on a separate line.
2. Repeat Exercise 1, but convert each string to uppercase instead of lowercase.
3. Write a script that removes whitespace from the following strings:
   string1= "   Filet Mignon"
   string2= "Brisket   "
4. Write a script that prints out the result of .startswith("be") on each of the following strings:
   string1= "Becomes"
   string2= "becomes"
5. Using the same strings from Exercise 4, write a script that uses string methods to alter each string so that .startswith("be") returns True for all of them.

INTERACT WITH USER INPUT

Get some input from a user with the input() function.

```
prompt= "Hi, what's your facebook account? "
user_input= input(prompt)
print("You said:", user_input)
```

```
Hi, what's your facebook account?  I dont have facebook
You said: I dont have facebook
```

REVIEW QUESTIONS:

1. Write a script that takes input from the user and displays that input back.
2. Write a script that takes input from the user and displays the input in lowercase.
3. Write a script that takes input from the user and displays the number of characters inputted.

REVIEW QUESTIONS:

Write a script named first_letter.py that first prompts the user for input by using the string "Tell me your name:" The script should then determine the first letter of the user's input, convert that letter to upper-case, and display it back.

STRING ARITHMETIC

- The “+” operator concatenates two string together.
- Strings can be “multiplied” by a number as long as that number is an integer.

```
num1 = '12'
num2 = '3'
num1+num2
```

```
'123'
```

```
num1 = '12'
num2 = 3
num1*num2
```

```
'121212'
```

```
num1 = '12'
num2 = '3'
num1*num2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 3
      1 num1 = '12'
      2 num2 = '3'
----> 3 num1*num2

TypeError: can't multiply sequence by non-int of type 'str'
```

```
num1 = '12'
num2 = 3
num1+num2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[4], line 3
      1 num1 = '12'
      2 num2 = 3
----> 3 num1+num2

TypeError: can only concatenate str (not "int") to str
```

CONVERTING STRINGS TO INTEGER/FLOAT

- int() stands for integer and converts objects into whole numbers,while float() stands for oating-point number and converts objects into numbers with decimal points.

```
num1 = '12'
num2 = int(num1)
num2
```

```
12
```

```
num1 = '12.0'
num2 = float(num1)
num2
```

```
12.0
```

```
num1 = '12.0'
num2 = int(num1)
num2
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[7], line 2
      1 num1 = '12.0'
----> 2 num2 = int(num1)
      3 num2

ValueError: invalid literal for int() with base 10: '12.0'
```

```
num1 = '12'
num2 = float(num1)
num2
```

```
12.0
```

- str() converts both integer and float to string.

CONVERTING INTEGER/FLOAT TO STRINGS

```
num1 = '12'
num2 = str(num1)
num2
```

```
'12'
```

```
num1 = '12.0'
num2 = str(num1)
num2
```

```
'12.0'
```

- str() converts both integer and float to string.

CONVERTING INTEGER/FLOAT TO STRINGS

```
num1 = '12'
num2 = str(num1)
num2
```

```
'12'
```

```
num1 = '12.0'
num2 = str(num1)
num2
```

```
'12.0'
```

REVIEW QUESTIONS:

- Create a string containing an integer, then convert that string into an actual integer object using int(). Test that your new object is a number by multiplying it by another number and displaying the result.
- Repeat the previous exercise, but use a floating-point number and float().
- Create a string object and an integer object, then display them side by side with a single print statement by using the str() function.
- Write a script that gets two numbers from the user using the input() function twice, multiplies the numbers together, and displays the result.

FINDING STRINGS IN STRINGS

- find() finds the location of one string in another string.

```
phrase= "the surprise is in here somewhere"
phrase.find("surprise")
```

```
4
```

```
phrase= "the surprise is in here somewhere"
phrase.find("apple")
```

```
-1
```

```
"My number is 555-555-5555".find(5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 "My number is 555-555-5555".find(5)

TypeError: must be str, not int
```

```
"My number is 555-555-5555".find("5")
```

```
13
```

REPLACING STRINGS

- .replace(<old string>, <new_string>)

```
phrase= "the surprise is in here somewhere"
phrase.replace("surprise", "cat")
```

```
'the cat is in here somewhere'
```

REVIEW QUESTIONS:

1. In one line of code, display the result of trying to .find() the sub-string "a" in the string "AAA".
2. Replace every occurrence of the character "s" with "y" in the string "Somebody said something to Samantha.".
3. Write and test a script that accepts user input using the input() function and displays the result of trying to .find() a particular letter in that input.

REVIEW QUESTIONS:

Write a script called translate.py that asks the user for some input with the following prompt: Enter some text:. Then use the .replace() method to convert the text entered by the user into “leetspeak” by making the following changes to lower-case letters:

- The letter a becomes 4
- The letter b becomes 8
- The letter e becomes 3
- The letter l becomes 1
- The letter o becomes 0
- The letter s becomes 5
- The letter t becomes 7

Your program should then display the resulting string as output.

Below is a sample run of the program:

```
Enter some text: I like to eat eggs and spam.
I 1ik3 70 347 3gg5 4nd 5p4m.
```

THANK YOU