PROGRAMMING FOR AI

Lecture 2

Basics of Python

Instructor: Zafar Iqbal

Agenda

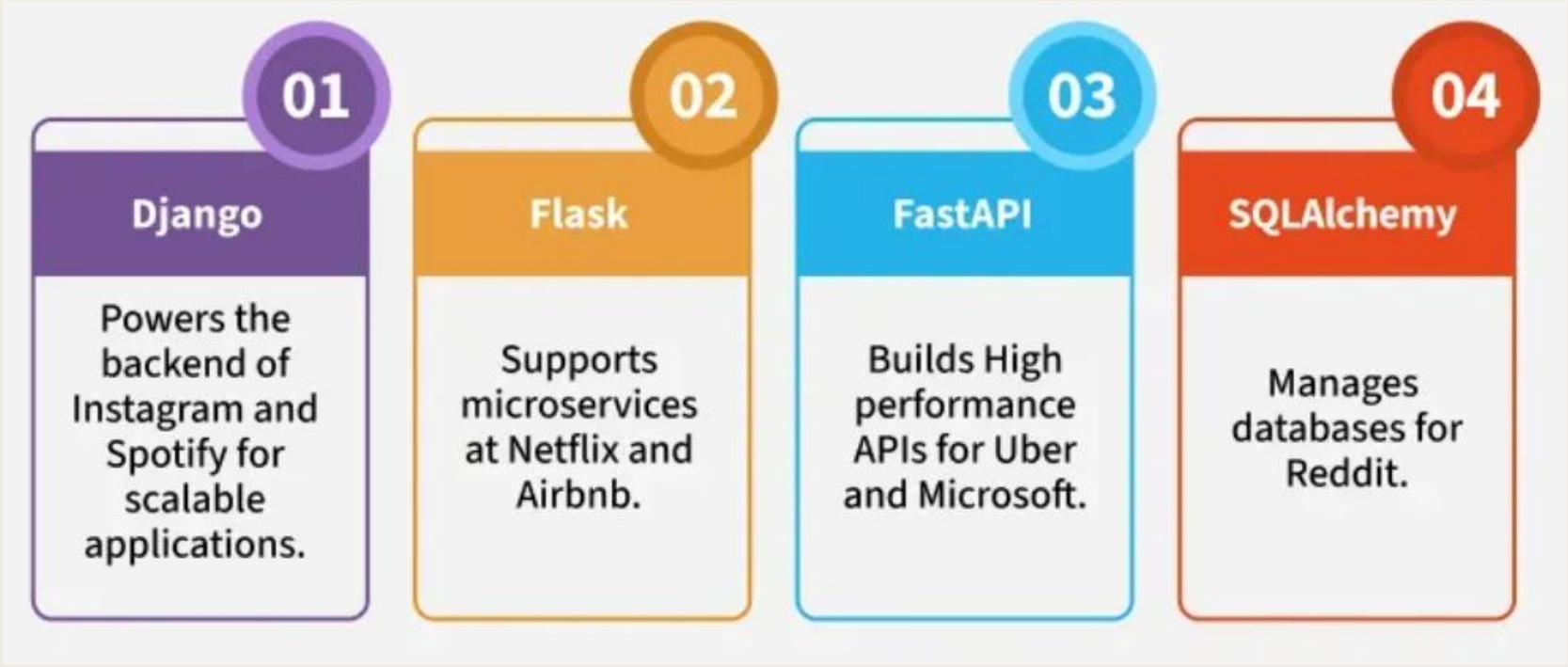
- Why Python?
- Applications of Python
- Basics of Python for AI
  - *Syntax,*
  - *Data Types,*
  - *Control Structures*
  - *Functions*

Why Python for AI?

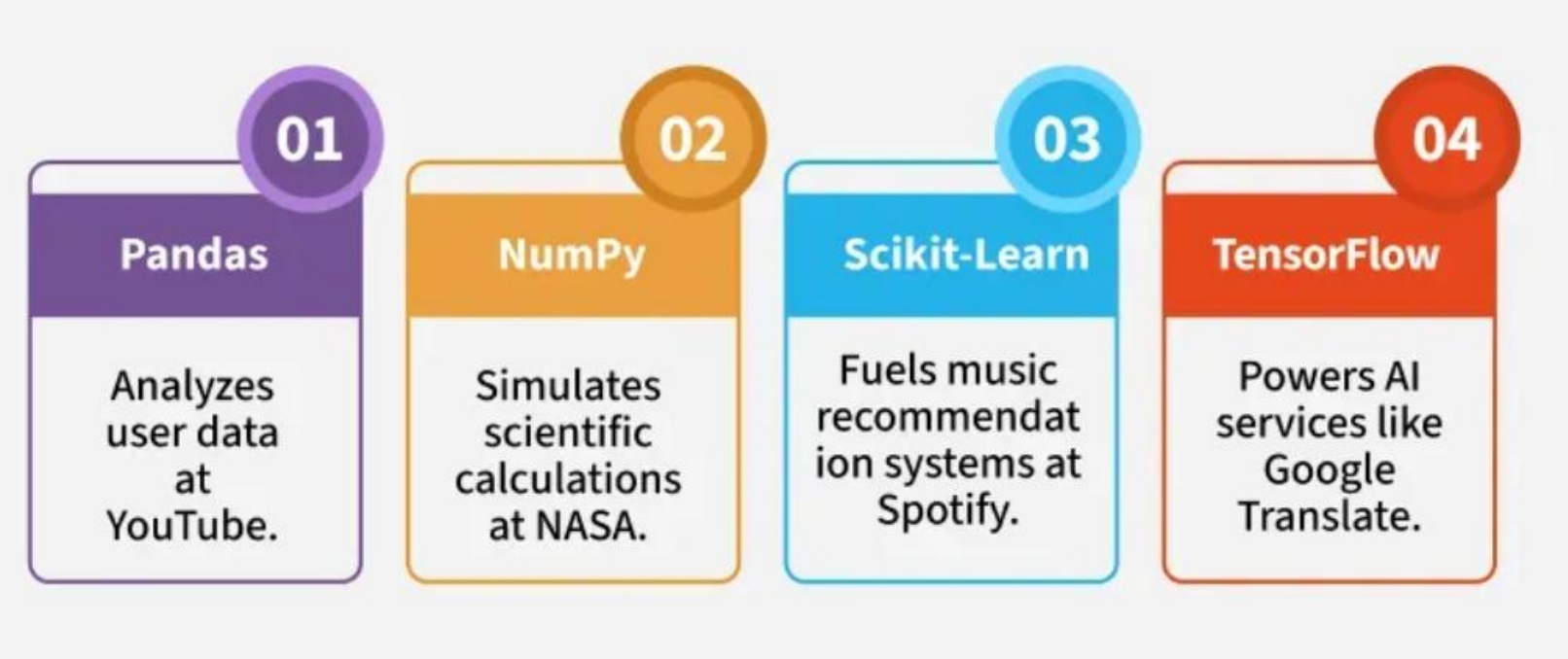
- Simple and readable syntax
- Extensive libraries (like NumPy, Pandas, TensorFlow)
- Strong community support
- Versatile for data science, ML, and AI applications

Applications of Python

Web Development



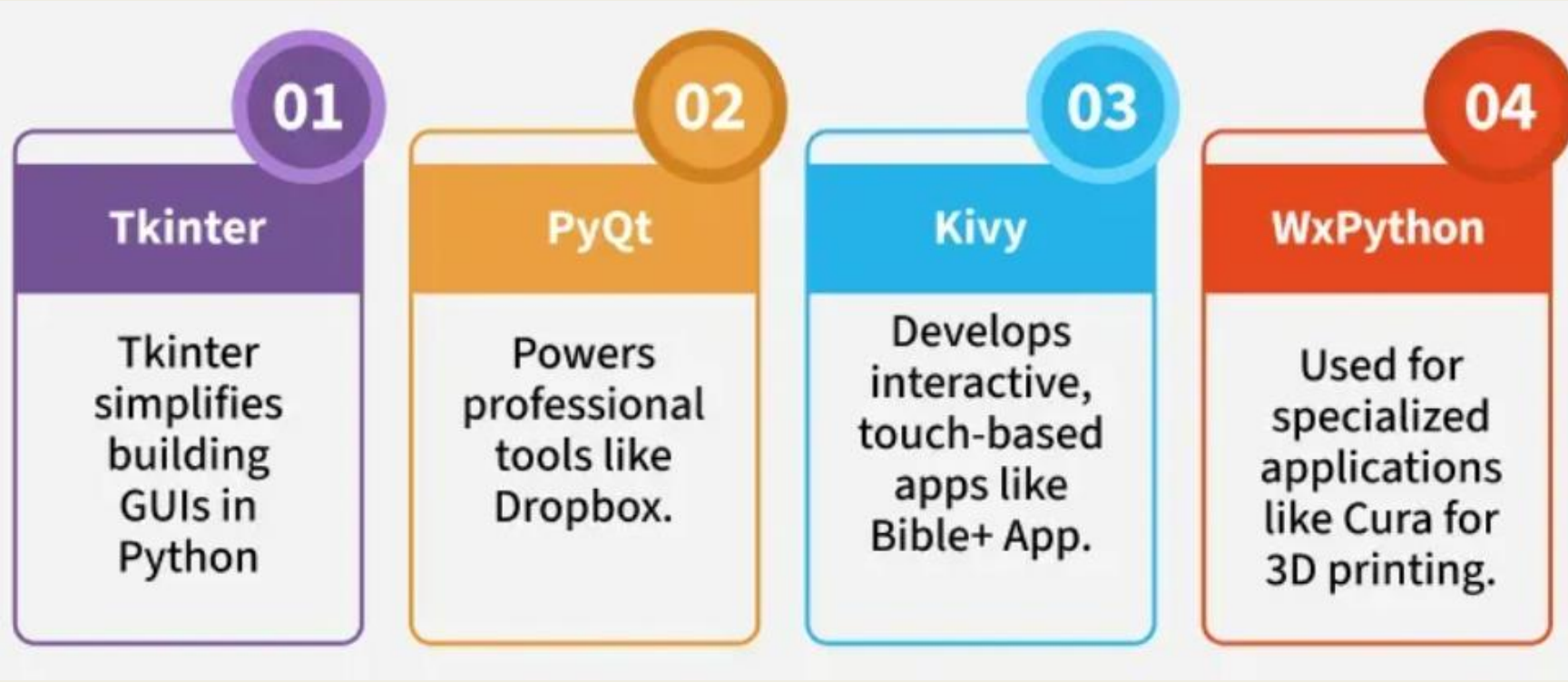
Data Science & Machine Learning



Web Scraping

Automation and Scripting

Desktop Applications



Game Development

Python Basics - Introduction

- *print("Hello, World!“)*

- Python Indentation
  - *Indentation refers to the spaces at the beginning of a code line.*
  - if 5 > 2:
    print("Five is greater than two!")
  - Python will give you an error if you skip the indentation
  - Syntax Error:
  - if 5 > 2:
    print("Five is greater than two!")

Python Comments

- #This is a comment
  print("Hello, World!")
- print("Hello, World!") #This is a comment
- #This is a comment
  #written in
  #more than just one line
  print("Hello, World!")
- """
  This is a comment
  written in
  more than just one line
  """
  print("Hello, World!")

Python Variables

- ```
  x = 5
  y = "John"
  print(x)
  print(y)
  ```

- ```
  x = 4       # x is of type int
  x = "Sally" # x is now of type str
  print(x)
  ```

- Casting
- If you want to specify the data type of a variable, this can be done with casting.
- ```
  x = str(3)    # x will be '3'
  y = int(3)    # y will be 3
  z = float(3)  # z will be 3.0
  ```
- print(x)
- print(y)
- print(z)
  ```

Python Data Types

- Variables can store data of different types, and different types can do different things.
- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set, frozenset
- Boolean Type: bool
- Binary Types: bytes, bytearray, memoryview
- None Type: NoneType

Print the data type of the variable x

```
x = 5
print(type(x))
```

Python Data Types

Example	Data Type
`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool
`x = b"Hello"`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview
`x = None`	NoneType

Python If ... Else

- Python supports the usual logical conditions from mathematics:
  - *Equals: a == b*
  - *Not Equals: a != b*
  - *Less than: a < b*
  - *Less than or equal to: a <= b*
  - *Greater than: a > b*
  - *Greater than or equal to: a >= b*

Python If ... Else

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
a = 33
b = 200
if b > a:
print("b is greater than
a") # you will get an error
```

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a = 34
b = 33
if a > b: print("a is greater than b")
```

```
a = 2
b = 330
print("A") if a > b else print("B")
```

Python If ... Else

```
a = 330
b = 330
print("A") if a > b
else
print("=") if a == b
else print("B")
```

AND keyword is a logical operator

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

OR keyword

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions
is True")
```

NOT keyword

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

Nested If Statement

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

Python Loops

- Python has two primitive loop commands:
  - *while loops*
  - *for loops*

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

For Loops

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
for x in "banana":
  print(x)
```

Python Functions

- In Python a function is defined using the def keyword

```
def my_function():
  print("Hello from a function")
```

Calling a function

```
def my_function():
  print("Hello from a function")

my_function()
```

Arguments

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

Arbitrary Arguments, *args

- If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.
- This way the function will receive a tuple of arguments, and can access the items accordingly

```
def my_function(*kids):
  print("The youngest child is " +
kids[2])

my_function("Emil", "Tobias", "Linus")
```

Keyword Arguments

- *You can also send arguments with the key = value syntax.*
- *This way the order of the arguments does not matter.*

```
def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

Arbitrary Keyword Arguments, **kwargs

- If you do not know how many keyword arguments that will be passed into your function, add two asterisk: ** before the parameter name in the function definition.
- This way the function will receive a dictionary of arguments, and can access the items accordingly:
- Example
- If the number of keyword arguments is unknown, add a double ** before the parameter name:

```
def my_function(**kid):
  print("His last name is " + kid["lname"])

my_function(fname = "Tobias", lname = "Refsnes")
```

Task 1

- Write a Python script that takes user input for their name and age,
- stores them in variables, and prints a message like:
- "Hello, Ahmad! You are 22 years old."
  - *(Replace Ahmad and 22 with user input values).*

Task 1

```
# Taking user input for name and age

name = input("Enter your name: ")
age = input("Enter your age: ")

# Printing the message using an f-string

print(f"Hello, {name}! You are {age} years old.")
```

Task 2

- Convert a string "56789" into an integer and a float.

Task 2

```
# Given string
num_str = "56789"

# Converting to integer
num_int = int(num_str)

# Converting to float
num_float = float(num_str)

# Printing results
print("Original String:", num_str)
print("After converting to Integer:", num_int)
print("After converting to Float:", num_float)
```

Task 3

- Write a for loop that prints numbers from 1 to 10.

- Write a while loop that prints numbers from 10 down to 1.

Task 3

```
for num in range(1, 11):
    print(num)
```

```
num = 10
while num >= 1:
    print(num)
    num -= 1  # Decrease num by 1 in each iteration
```

Task 4

Write a Python program that prints all even numbers between 1 and 50 using a loop

Task 4

```
for num in range(2, 101, 2):  # Start at 2, go up to 100, step by 2
    print(num, end=" ")
```

```
num = 2
while num <= 100:
    print(num, end=" ")
    num += 2
```