



PRIMEINTUIT

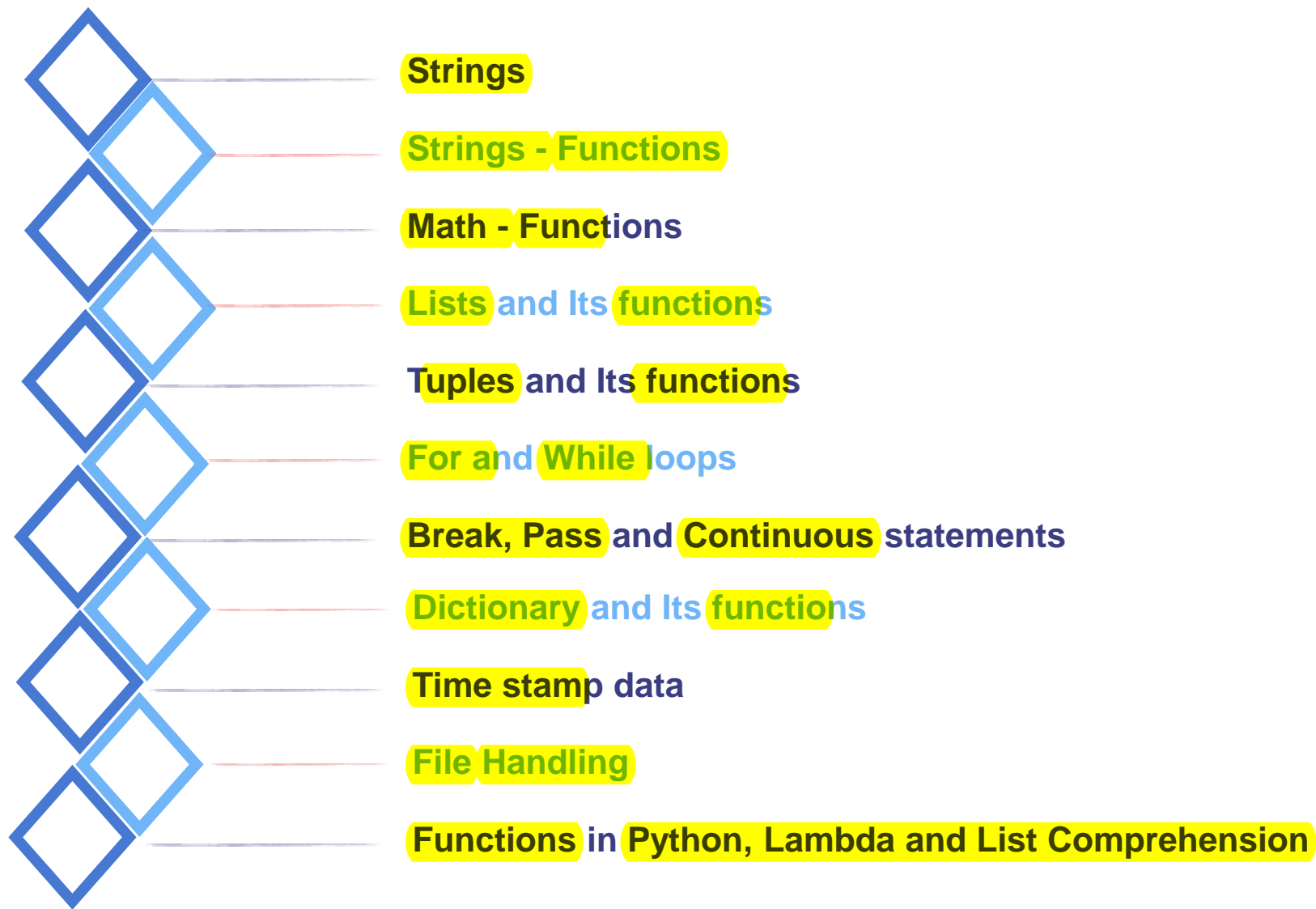
Finishing School

Strings and String Functions

Python – Session – 3



Agenda

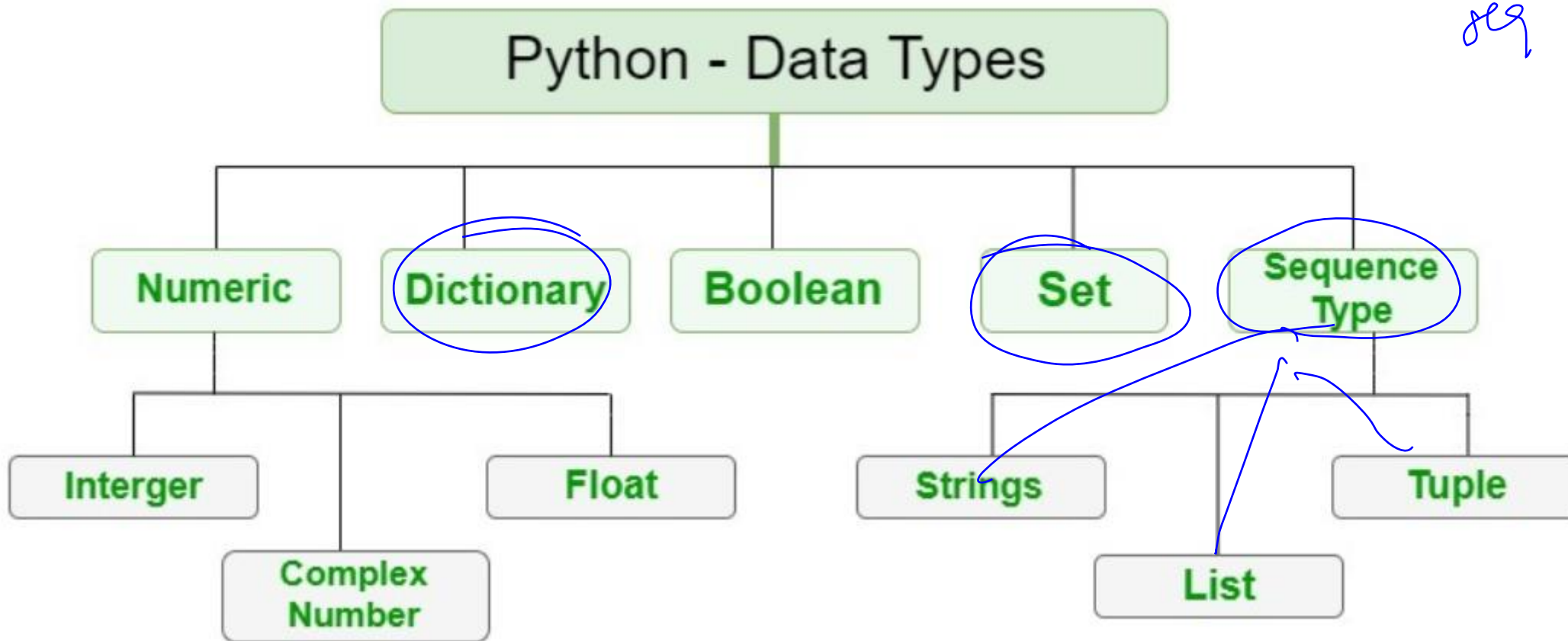




Python Data Types

Following are the standard or built-in data type of Python:

Dictionary, Set, seq are not data type





Sequence Types

In Python, sequence is the ordered collection of similar or different data types. Sequences allows to store multiple values in an organized and efficient fashion.

There are 3 sequence types in Python:

- String
- List
- Tuple



Strings

Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes ' the same as double quotes ".

Creating strings is as simple as assigning a value to a variable.

Example:

```
name = 'Prime Intuit'
```

```
print(name)
```

```
phrase = "Prime Intuit one of the finest finishing school"
```

```
Print(phrase)
```

```
Multiline = """ There was a boy named Raja
```

```
    who was a very good sprinter
```

```
    He could run faster then a deer"""
```



Accessing Values in Strings

Python does not support a character type; these are treated as strings of length one, thus also considered a substring.

To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring.

Example:

```
Name = 'Prime Intuit'
```

```
Phrase = "Prime Intuit one of the finest finishing school"
```

```
print(name[0])
```

```
print(name[0:5])
```

Length Functions: `len()`

```
Print(len(Name))
```



String Operators

a = "Hello"

b = "Prime Intuit"

c = "Finishing School"

+ or **concatenation operator** will concatenate 2 or more strings

print(a+b)

output: HelloPrime Intuit

print(a+b+c)

output: HelloPrime IntuitFinishing School

print(a + b + c[9:]) # concatenation with slicing

output: HelloPrime Intuit School



String Operators

a = "Hello"

b = "Prime Intuit"

c = "Finishing School"

*****, **Repetition Operator**, concatenates **multiple copies** of same string

```
print (a*3 ) # repetition operator
```

output: Hello Hello Hello

```
print (a*3 +b) # repetition with concatenation
```

output: HelloHelloHelloPrime Intuit

```
print ( a*2 + " " + b + c[9:]*2) # repetition with concatenation & slicing
```

output: HelloHello Prime Intuit School School



String Operators

a = "Hello"

b = "Prime Intuit"

c = "Finishing School"

[], [:] Slicing and Range slicing operators, will return characters from string

print(a[1])

output e

print(a[0])

output H

print(b[:6])

output Prime

print(b[6:])

output Intuit

print(b[6:9])

output Int

Print(b[-1])

Negative indexing

o/p



String Operators

```
a = "Hello"
```

```
b = "Prime Intuit"
```

```
c = "Finishing School"
```

in , Membership Operator, Returns true (Boolean) if a character exists in the given string

```
d = 'l' in b
```

```
print(d)          # output True
```

```
print(type(d))    # output <class 'bool'>
```



String Operators

```
a = "Hello"
```

```
b = "Prime Intuit"
```

```
c = "Finishing School"
```

in , Membership Operator, Returns true (Boolean) if a character exists in the given string

```
d = 'l' in b
```

```
print(d)
```

```
# output True
```

```
print(type(d))
```

```
# output <class 'bool'>
```



String Operators

```
a = "Hello"
```

```
b = "Prime Intuit"
```

```
c = "Finishing School"
```

not in, Membership Operator, **Returns true (Boolean)** if a character does not exists in the given string

```
d = 'l' not in b
```

```
print(d)           # output False
```

```
print(type(d))     # output <class 'bool'>
```



String Operators

a = "Hello"

b = "Prime Intuit"

c = "Finishing School"

r/R, Raw String – Suppresses actual meaning of Escape characters

`print('This will continue printing \n in next line')`

**# output: This will continue printing
in next line**

`print('This will continue printing' R'\n' 'in next line')`

output: This will continue printing\nin next line



String Operators

a = "Hello"

b = "Prime Intuit"

c = "Finishing School"

%, Format - Performs String formatting, works in combination with **Format Symbol**

Example:

```
print ("My name is %s and weight is %d kg!" % ('Niranjan', 71))
```

output: My name is Niranjan and weight is 71 kg! in next line

%c – character, **%d** – signed decimal integer, **%u** – unsigned decimal integer, **%f** – floating point real number



Python – Built in String Methods

e = "Hello Hello Hello This is a grand sale"

b = "Prime Intuit"

c = "Finishing School"

f = 'there was a big Tiger'

Method	Description	Example
<code>capitalize()</code>	Capitalizes first letter of string	<code>print(f.capitalize())</code>
<code>center(width, fillchar)</code>	Returns a string padded with <i>fillchar</i> with the original string centered to a total of <i>width</i> columns.	<code>print(e.count("Hello"))</code>
<code>len(string)</code>	Returns the length of the string	<code>print(len(a))</code>
<code>count(str, beg=0, end=len(string))</code>	Counts how many times <i>str</i> occurs in string or in a substring of string if starting index <i>beg</i> and ending index <i>end</i> are given.	<code>e.count("Hello")</code>



Python – Built in String Methods

Method	Description	Example
find()	Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.	
Index()	Same as find(), but raises an exception if str not found.	
Isalnum()	Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.	<pre>txt = "Company 12" x = txt.isalnum() print(x) #output True</pre>
Isalpha()	Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.	



Python – Built in String Methods

Method	Description	Example
isdigit()	Returns true if the string contains only digits and false otherwise.	
islower()	Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.	
isnumeric()	Returns true if a unicode string contains only numeric characters and false otherwise.	
isspace()	Returns true if string contains only whitespace characters and false otherwise.	



Python – Built in String Methods

Method	Description	Example
isupper()	Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.	
join()	Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.	
lower()	Converts all uppercase letters in string to lowercase.	
lstrip()	Removes all leading whitespace in string.	