Introduction to Python

T5 Bootcamp by SDAIA

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

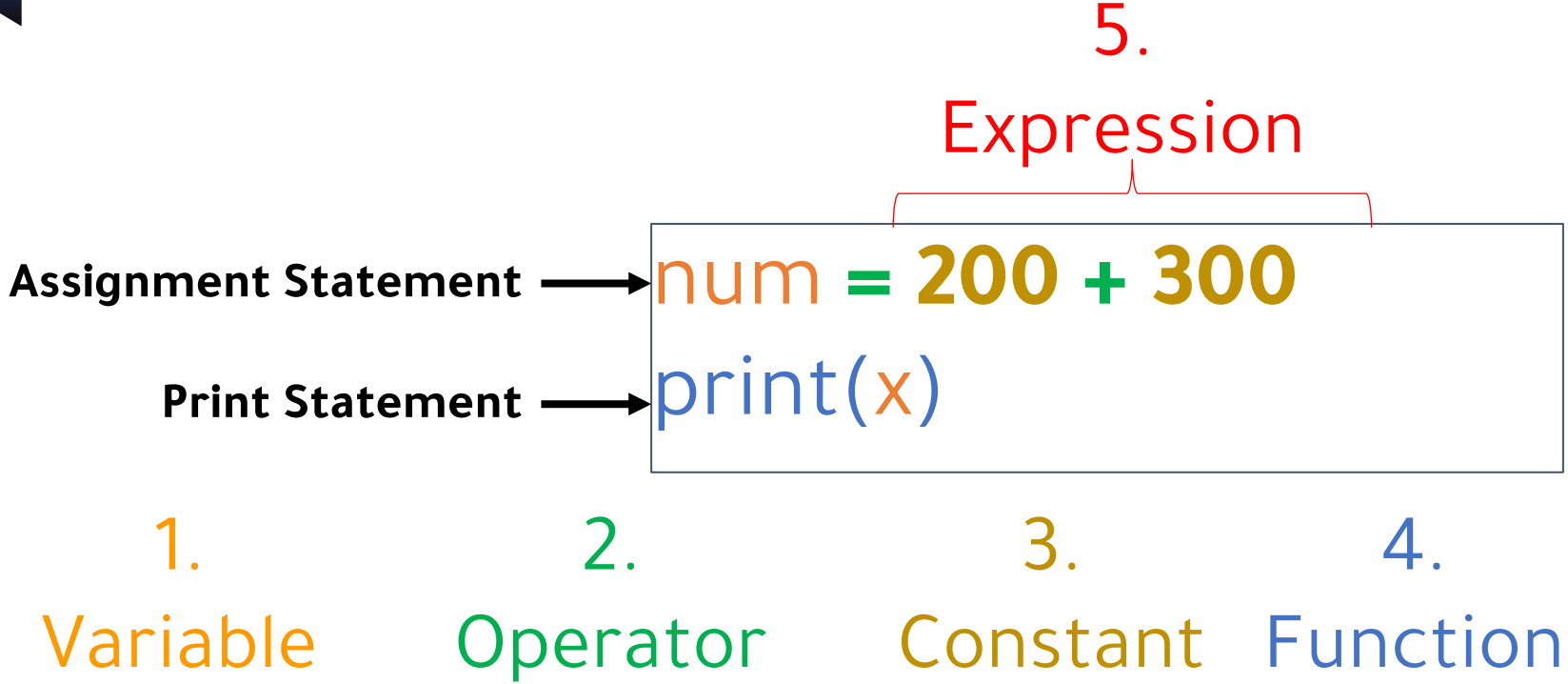
Variables
&
Data Types

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

Outline

- Variables and Values in Python
- Data Types
  - Numeric Types
    - Order Precedence
    - Bitwise Operations
  - String Types
    - Indexing and Slicing
  - Boolean Types
    - Truth Tables
  - Collection Types
    - Mixed Containers
  - None Type
- Naming Rules and Conventions

Terminology



Values

- Every ***Expression*** evaluates to a **value**
- Every value has a data ***Type***

	expression	value	type
literal	500	500	integer
literal	3.14	3.14	float
+	200 + 300	500	integer
+	10.0 + 5.0	15.0	float

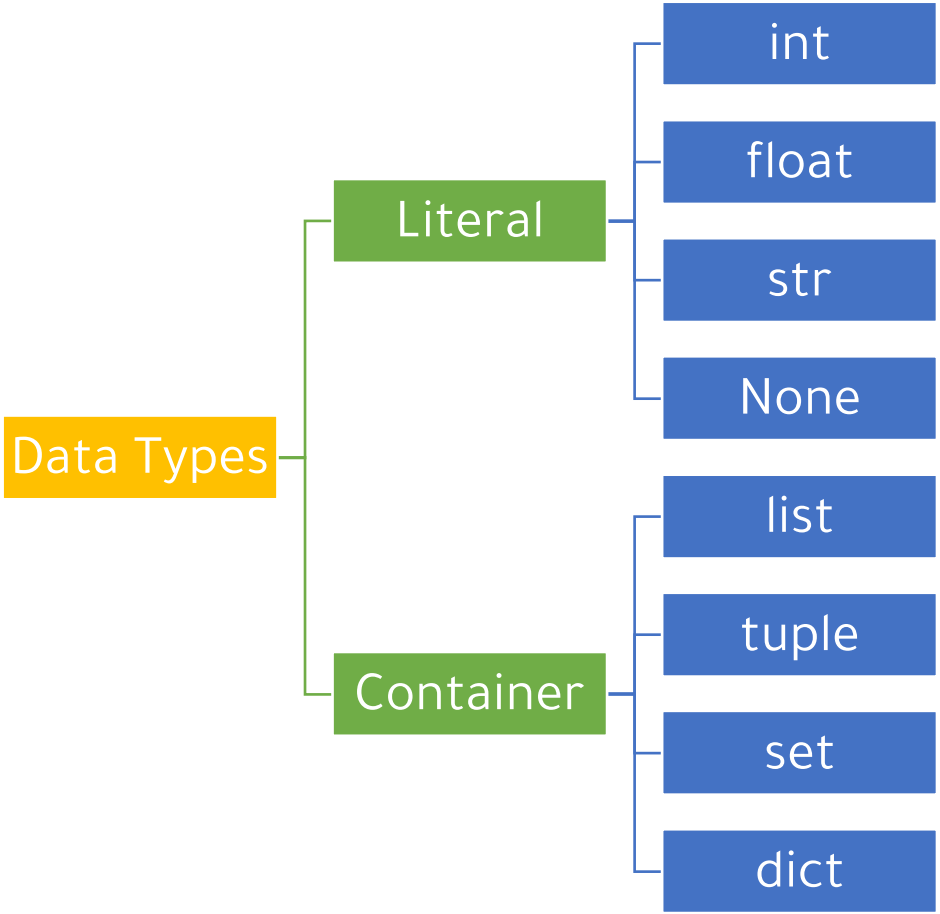
Data Types

- **Data types** represents the kind of value that tells what operations can be performed on a particular data. For example:

- **Numeric Types** can be added, subtracted, multiplied, …etc.
- **String Type (str)** can be concatenated (joined) repeated, …etc.

- Python differentiates between **5 + 5** and **“5” + “5”**
- Also, it will complain if you try: **“5” + 5**.

Python built-in data type hierarchy



Numeric Types

Numeric types are essential for performing

- mathematical calculations
- Coordinates
- representing quantities in scientific and engineering applications

- **Integers (int):** Represent whole numbers, positive, negative, or zero (e.g., -5, 0, 1024).
- **Floats (float):** Represent real numbers with decimal points (e.g., 3.14, -1.2345).
- **Complex numbers (complex):** Represent numbers with a real and imaginary part (e.g., 3+2j, where j represents the imaginary unit).

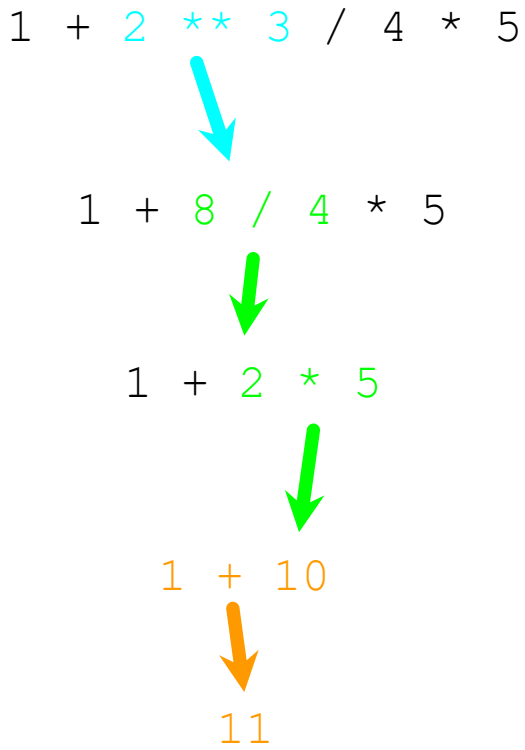
Order Precedence

Precedence	Operators	Associativity
1	Parenthesis (())	Left to Right
2	Exponentiation (**)	Right to left
3	Multiplication (*) and Division (/)	Left to right
4	Floor division (//) and Modulo (%)	Left to right
5	Addition (+) and Subtraction (-)	Left to right

Example: Order Precedence

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11.0
>>>
```

Parenthesis
Power
Multiplication
Addition
Left to Right



Bitwise Operations on Integer Types

```
int a = 0 1 0 1
int b = 1 1 0 0 l
---------
int c = 1 1 0 1
```

```
int a = 0 1 0 1
int b = 1 0 0 1 ^
---------
int c = 1 1 0 0
```

```
int a = 0 1 1 0
int b = 1 0 1 0 &
---------
int c = 0 0 1 0
```

```
int a = 1; // 0001b
int b = a << 1; // 0010b
```

```
int a = 2;        // 0010b
int b = a >> 1;   // 0001b
```

Operation	Result	Notes
`x | y`	bitwise *or* of *x* and *y*	(4)
`x ^ y`	bitwise *exclusive or* of *x* and *y*	(4)
`x & y`	bitwise *and* of *x* and *y*	(4)
`x << n`	*x* shifted left by *n* bits	(1)(2)
`x >> n`	*x* shifted right by *n* bits	(1)(3)
`~x`	the bits of *x* inverted	

String Type

- Strings are used to store text data, create user interfaces, and process textual information like file names, website URLs, and user input.

```
first_name = "Ahmad"
address = "Riyadh, Saudi Arabia"
phone = "00966555555555"

message = """Hello everyone,
I hope you are enjoying the course,

Thank you.
"""
```

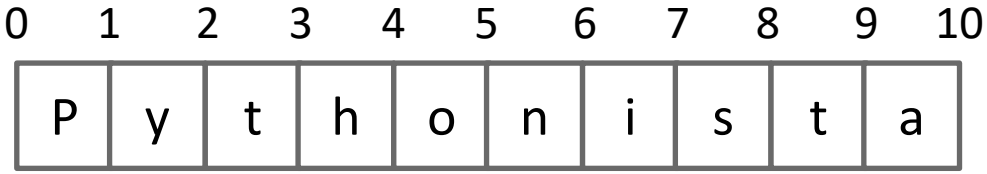
- **str**: Represents sequences of characters, including letters, numbers, symbols, and spaces (e.g., "Hello, world!", "My name is Bard").

- There is no **char** type in Python. We simply use an **str** with length 1
  - Example: “C”

String: Indexing

- A string is a sequence (collection) of characters
- Sequences can be indxed using []
  - 1st element is at index: 0
  - last element is at index: -1

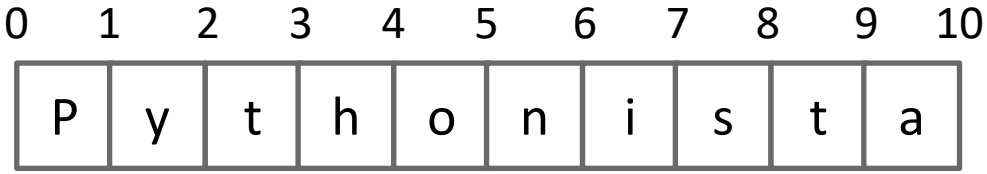
String: Indexing




```
name = "Pythonista"

name[0]  # P
name[1]  # y
name[9] # a
name[10] # IndexError: string index out of range
```

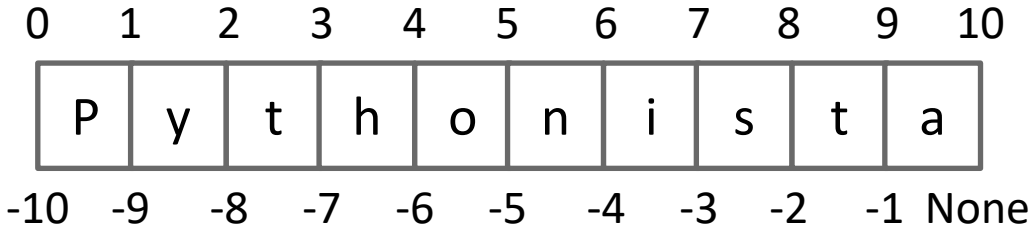
String: Slicing




```
name = "Pythonista"

name[2:5] # tho
name[:5]  # Pytho
name[2:]  # thonista
name[2:None] # thonista
```

String: Negative Indexing and Slicing




```
name = "Pythonista"

name[-1] # a
name[-4:-2] # is
name[:-4]  # Python
name[-4:] # ista
name[-4:None] # ista
```

Boolean

- Represents logical truth values, either **True** or **False**.
- Booleans are crucial for making decisions based on conditions, implementing logical operations (e.g., "if" statements, loops), and representing binary states (e.g., on/off, enabled/disabled).

x	y	x and y
F	F	F
F	T	F
T	F	F
T	T	T

x	y	x or y
F	F	F
F	T	T
T	F	T
T	T	T

x	not x
F	T
T	F

Fig

Comparisons

- This table summarizes the comparison operations:

Operation	Meaning
`<`	strictly less than
`<=`	less than or equal
`>`	strictly greater than
`>=`	greater than or equal
`==`	equal
`!=`	not equal
`is`	object identity
`is not`	negated object identity

Collection Types

- **Collection Types:** Python offers various data structures to organize and store collections of data, including: **Lists**, **Tuples**, **Dictionaries**, and **Sets**.
    - **list:** Ordered, mutable collections of items enclosed in square brackets [] (e.g., [1, "apple", True]).
    - **tuple:** Ordered, immutable collections of items enclosed in parentheses () (e.g., (10, "orange", False)).
    - **dict:** Unordered collections of key-value pairs enclosed in curly braces {} (e.g., {"name": "Alice", "age": 30}).
    - **set:** Unordered collections of unique items enclosed in curly braces {} (e.g., {1, "banana", 1}).

Heterogenous

- Container data types are *Heterogenous*; meaning the data they hold doesn't have to be of one type.

```
my_mixed_list = ["abc", 123, False, ['AABB', 'CC’]]

my_mixed_tuple = ("abc", 123, False, ['AABB', 'CC’])

my_mixed_set = {"abc", 123, False, ('AABB', 'CC’)}

my_mixed_dict = {"abc": 123, 123: "abc",
                 "def": [1, 2, 3], "ghi": {"a": 1, "b": 2}}
```


None

- **None**: A special data type representing the absence of a value.
- For example, it is used to indicate that a variable has no value if it has a value of None.

Naming Rules and Conventions

- **Variables are case-sensitive**
  - my_variable = 5
  - My_Variable = 10
- **Variables cannot start with a number**
  - 2my_variable = 5
- **You cannot use reserved keywords**
  - class = 5
  - True = 1
  - break = True

Better not use built-in function names

```
sum = 5
max = 10
list = [10, 20, 30]
```


Name Variables

- “There are only two hard things in Computer Science: cache invalidation and naming things”. - Phil Karlton

- Which of the two is better?

```
x2 = 35.0
y = 12.50
mne = x2 * y
print(mne)
```

```
hours = 35.0
rate = 12.50
pay = hours * rate
print(pay)
```


Exercise 1: E-commerce

Guess the data type:

- **product_id:** (e.g., 12345)
- **product_name:** (e.g., "T-Shirt")
- **price:** (e.g., 19.99)
- **in_stock:** (e.g., True)
- **customer_reviews:** (e.g., ["Great product!", "Fast shipping"])
- **product_rating**: (e.g., 1, 2, 3, 4, 5)

Exercise 2: Weather Data

Guess the data type:

- **city_name:** (e.g., "London")
- **temperature:** (e.g., 23.5)
- **weather_condition:** (e.g., "Sunny")
- **humidity:** (e.g., 65%)
- **chance_of_rain:** (e.g., 0.2)
- **wind_direction**: (e.g., “N”, “S”, “E”, “W”)

Thank you

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority