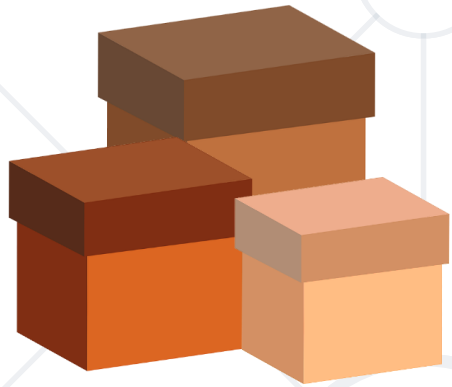


Data Types and Variables

Types of Operators



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

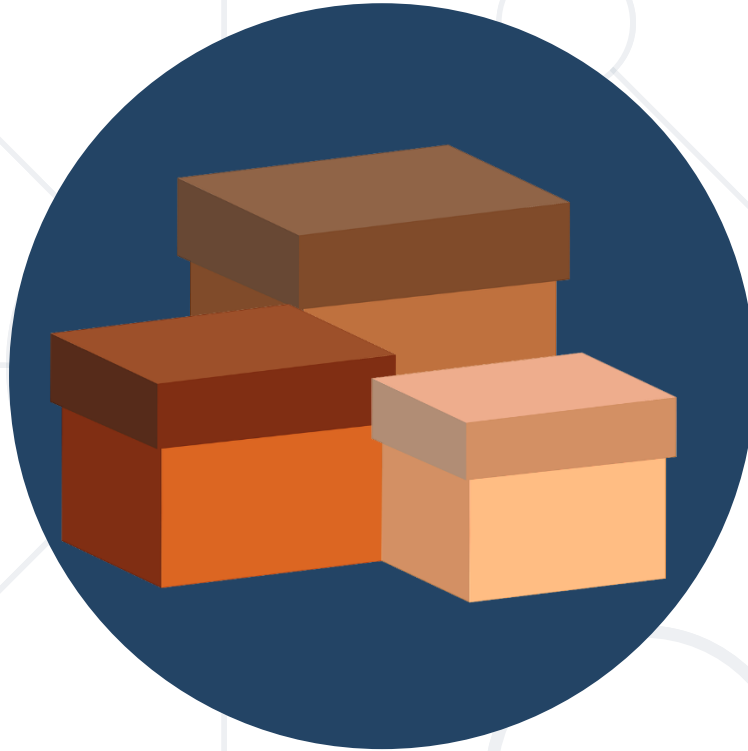
sli.do

#fund-python

Table of Contents

1. What is a **Data Type**
2. **Strings**
3. **Numbers**
4. **Booleans**
5. Additional Data Types



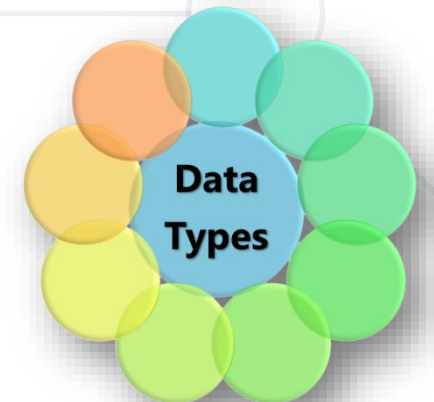


What is Data Type

Definition and Examples

What is Data Type?

- A classification that specifies **which type of value** a variable has and **what type of operations** can be applied to it
- In Python we have the following data types:
 - Numeric Types: int, float, complex, decimal
 - Text Type: str
 - List, Set, Tuple, Dictionary
 - Boolean



```
int_num = 10           # int value
float_num = 10.2       # float value
a_str = 'Hello world'  # str value
is_true = True         # bool value
list = [123, 'abcd', 10.2, 'd'] # list
dict = {'name': 'Amy', 'age': 10} # dictionary
```

- Python is a **dynamic** language
- Variables are **not** directly associated with any particular value type
- Any variable can be **assigned** (and **re-assigned**) values of all types

```
variable = 42           # variable is now an int
variable = 'bar'        # variable is now a string
variable = True         # variable is now a boolean
```

Check the Type of a Variable

- The **type()** function helps you find the type of the variable

```
print(type('123'))      # <class 'str'>
print(type(123))         # <class 'int'>
print(type(123==123))   # <class 'bool'>
```

- The **isinstance()** function checks if the specified object is of the specified type

```
print(isinstance('123', str))    # True
print(isinstance(123, str))       # False
print(isinstance(123==123, bool)) # True
```




'ABC'

Strings

Sequence of Characters

What is a String?

- Used to represent **textual data**
- Each element in the String occupies a **position** in the String
- The **first** element is at **index 0**, the next at index 1, and so on
- The **length** of a String is the number of elements in it

```
name = 'George'  
print(name[0])  #'G'
```

Accessing element at index



String Literal

- String literals in Python are surrounded by either single quotation marks or double quotation marks: 'hello' is the same as "hello"
- The `len()` method returns the length of a string

```
a = "Hello, World!"  
print(len(a)) # 13
```



Strings are Immutable

- Unlike in languages like C, Python strings are **immutable**
 - This means that once a string is created, it is **not** possible to **modify** it

```
name = 'George'  
name[0] = 'P' # Error
```



String Interpolation

- From Python 3.6+ we can use **string interpolation**
- These are string literals that allow **embedded** expressions

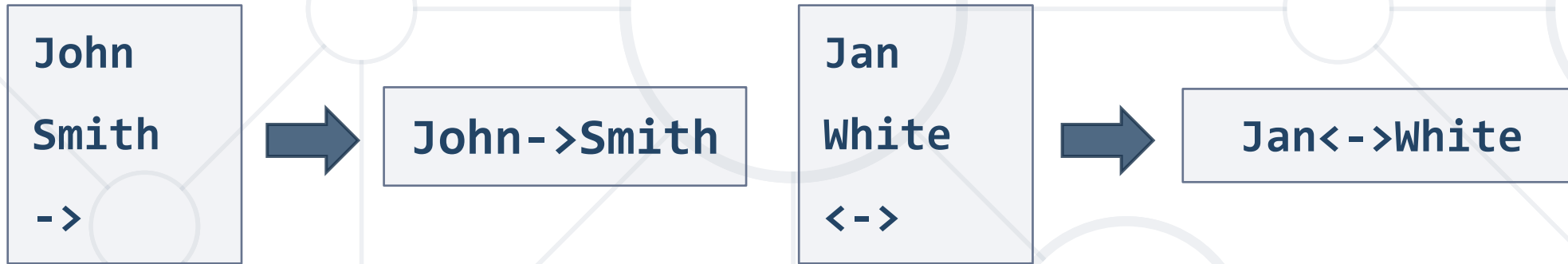
```
name = 'Rick'  
age = 18  
print(f'{name} = {age}') # Rick = 18
```

Place your **variables** inside {}



Problem: Concatenate Names

- Receive two **names** as **string parameters** and a **delimiter**
- Print the names **joined** by the delimiter



```
first_name = input()
second_name = input()
delimiter = input()
print(f'{first_name}{delimiter}{second_name}')
```




123

Numbers

Integer, Float

Integer

- **Int** or **integer** is a whole number, positive or negative, without decimals, of unlimited length
- Python integers are **immutable**



```
x = 1           # int
y = 231223423352 # int
z = -2312312     # int
```


Float

- **Float** is a floating-point real number, positive or negative, written with a decimal point dividing the integer and fractional parts, of unlimited length
- Python floats are **immutable**

```
x = 1.1           # float
y = 231223423352.24 # float
z = -2312312.689  # float
```



Problem: Meters to Kilometers

- Write a program that converts meters to kilometers formatted to the second decimal point.

1852 → 1.85

798 → 0.80

Solution: Meters to Kilometers

```
meters = int(input())  
kilometers = meters/1000  
print(f'{kilometers:.2f}')
```

Problem: Pounds to Dollars

- Write a program that converts British pounds to US dollars formatted to the 3rd decimal point
 - 1 British Pound = 1.31 Dollars

80 → 104.800

39 → 51.090

Solution: Pounds to Dollars

```
pounds = int(input())  
dollars = pounds * 1.31  
print(f'{dollars:.3f}')
```

Problem: Centuries to Minutes

- Write a program to enter an integer number of centuries and convert it to years, days, hours, and minutes

Centuries = 1



1 centuries = 100 years = 36524 days = 876576 hours = 52594560 minutes

Solution: Centuries to Minutes

```
centuries = int(input())
years = centuries * 100
days = int(years * 365.2422)
hours = 24 * days
minutes = 60 * hours

print(f"{centuries} centuries = {years} years = {days} days = {hours} hours = {minutes} minutes")
```

Tropical year has
365.2422 days

int() converts float
to int



True
False

Booleans

Conditions

What is a Boolean?

- Boolean represents a logical entity and can have two values: **True** and **False**
- You can use the **bool()** function to find out if an expression (or a variable) is true:

```
print(bool(10 > 9)) # True
```

- Or even easier:

```
print(10 > 9) # True
```



Comparisons and Conditions

Operator	Description	Example
<code>==</code>	equal	<code>if (day == 'Monday')</code>
<code>></code>	greater than	<code>if (salary > 9000)</code>
<code><</code>	less than	<code>if (age < 18)</code>
<code>>=</code>	greater than or equal	<code>if (6 >= 6)</code>
<code>!=</code>	not equal	<code>if (5 != 5)</code>
<code>in</code>	item is in sequence	<code>'a' in 'abc' # True</code>

- Everything with a "value" is **True**

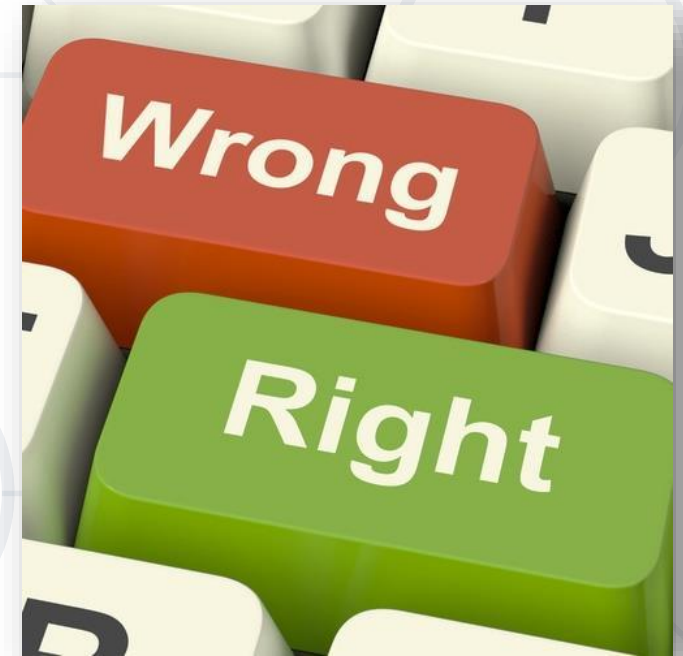
```
number = 1
if (number):
    print(number)    # 1
```

True

- Everything without a "value" is **False**

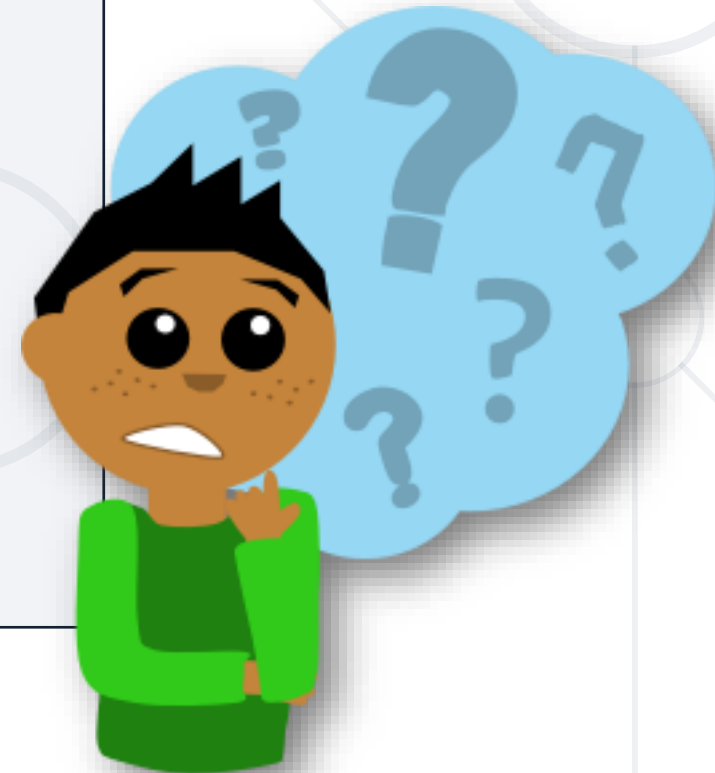
```
number = None
if (number):
    print(number)
else:
    print('false')    # False
```

False



Booleans Examples

```
x = 0
bool(x)    # False
x = -0
bool(x)    # False
x = ''
bool(x)    # False
x = False
bool(x)    # False
x = None
bool(x)    # False
```



Problem: Special Numbers

- Write a program that reads an integer **n**. For all numbers in the range **1...n** print the number and if it is special or not (**True** / **False**)
 - A number is **special** when the **sum** of its digits is **5**, **7**, or **11**

20



1 -> False

2 -> False

3 -> False

4 -> False

5 -> True

6 -> False

7 -> True

8 -> False

9 -> False

10 -> False

11 -> False

12 -> False

13 -> False

14 -> True

15 -> False

16 -> True

17 -> False

18 -> False

19 -> False

20 -> False

Solution: Special Numbers

```
n = int(input())

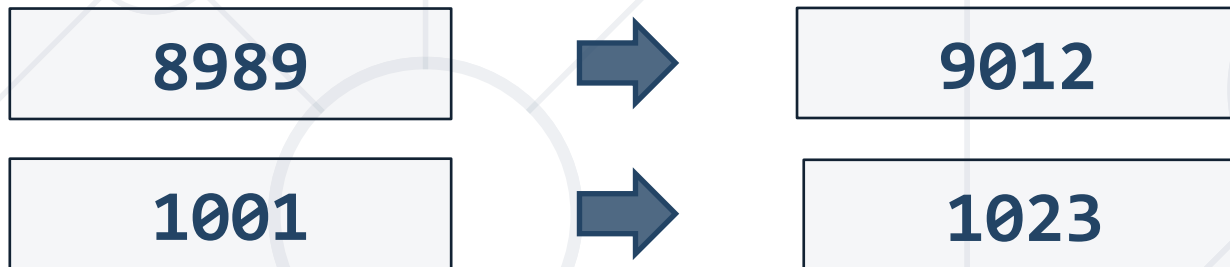
for num in range(1, n + 1):
    sum_of_digits = 0
    digits = num

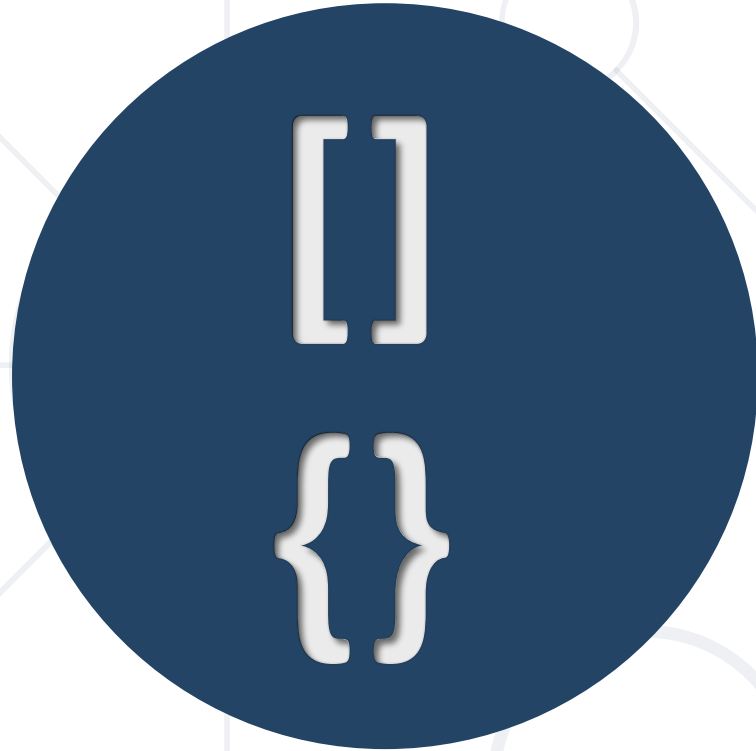
    while digits > 0:
        sum_of_digits += digits % 10
        digits = int(digits / 10)

# TODO: check whether the sum is special
```

Problem: Next Happy Year

- Happy Year is the year with **only distinct digits**
 - for example, **2018**
- Write a program that receives an **integer** number and **finds the next happy year**





Additional Datatypes

List, Tuple, Set, Dict

Definition and Examples

- A **list** contains items separated by commas and enclosed within square brackets

```
cars = ["Saab", "Volvo", "BMW"]
```

- A **tuple** is a collection which is ordered and **unchangeable**. In Python, tuples are written with round brackets

```
example_tuple = ("apple", "banana", "cherry")  
print(example_tuple)
```



Definition and Examples

- A **set** is a collection which is unordered and unindexed. Sets are written with curly brackets.

```
example_set = {"apple", "banana", "cherry"}  
print(example_set)
```

- A **dictionary** is a collection that is **ordered** (python 3.7+), changeable, and indexed
 - They have keys and values

```
example_dict = {"brand": "Ford", "model": "Mustang"}  
print(example_dict)
```





None

None Keyword

What is None?

- The **None** keyword is used to define a null value or no value at all
- There are two ways to check if a variable is None
 - One way can be performed by using the **is** keyword
 - Another is using the **==** syntax

```
if null_variable is None:  
    print('null_variable is None')
```

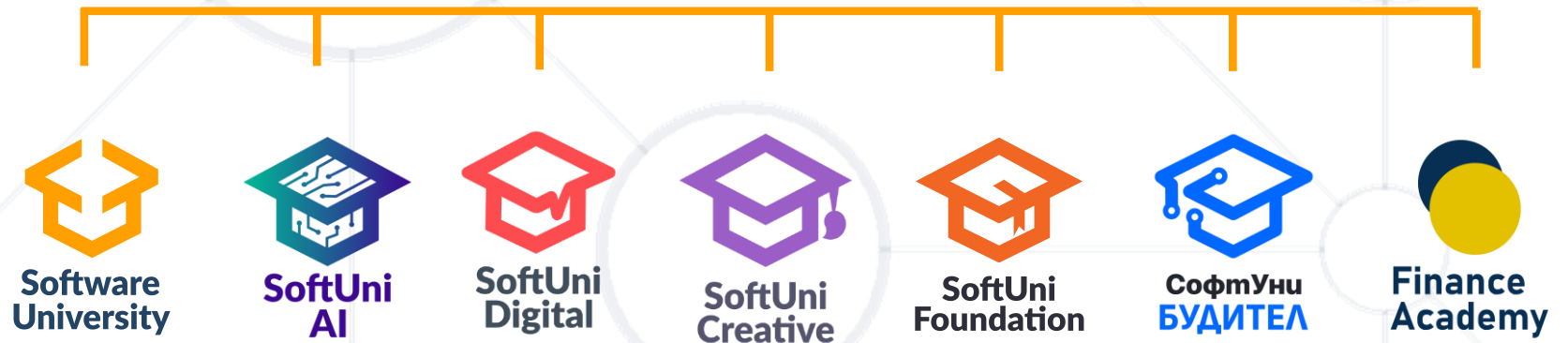
```
if null_variable == None:  
    print('null_variable is None')
```



- Python supports the following data types:
 - String
 - Bool
 - Int
 - Float
 - List, Tuple, Set, Dict
- **None** is nothing



Questions?



SoftUni Diamond Partners



**SUPER
HOSTING
.BG**



INDEAVR
Serving the high achievers



THE CROWN IS YOURS

VIVACOM

- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, softuni.org
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos, and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

