🐍 Python BreakFast Practice by: RANJITSINGH MUGAVEKAR

📊 MACHINE LEARNING USING PYTHON

📈 DATA SCIENCE USING PYTHON

🧠 MindWave LAB_3 HOLKHAR NAGAR, MUGAON

#Python Data Types Explained with Examples

1. Numeric Types

Integer (int)

- Whole numbers (positive or negative) without decimals
- Unlimited precision in Python 3

In [1]:
```
#Example01.
x = 10
y = -5
z = 9876543210

print(type(x))  # <class 'int'>
print(x + y)    # 5
print(z * 2)    # 19753086420
```

```
<class 'int'>
5
19753086420
```

In [ ]:
```

```

Float (float)

- Numbers with decimal points or in exponential form
- Implemented as double-precision (64-bit) floating-point numbers

In [9]:
```
#Example
a = 3.14
b = -0.5
c = 2.5e3  # Scientific notation (2.5 × 10³ = 2500.0)

print(type(a))  # <class 'float'>
print(a * b)    # -1.57
print(c)        # 2500.0
```

```
<class 'float'>
-1.57
2500.0
```

In [ ]:

2. Boolean (bool)

Boolean

- Represents truth values: True or False
- Subclass of integers where True == 1 and False == 0

In [6]:
```python
#Example:

is_active = True
has_permission = False

print(type(is_active))     # <class 'bool'>
print(is_active and has_permission)  # False
print(int(True))           # 1
print(int(False))          # 0
```

```
<class 'bool'>
False
1
0
```

In [ ]:

3. String (str)

String

- Sequence of Unicode characters (immutable)
- Can be created with single, double, or triple quotes

In [7]:
```
#Example
name = "Alice"
greeting = 'Hello'
multiline = """This is a
multi-line string"""

print(type(name))          # <class 'str'>
print(greeting + " " + name)  # Hello Alice
print(len(name))           # 5
print(name[0])             # 'A' (indexing)
print(name[1:4])           # 'lic' (slicing)
```

```
<class 'str'>
Hello Alice
5
A
lic
```

In [ ]:
```

```

4. Sequence Types

Sequence Type

- List (list)
  - Ordered, mutable sequence of items
  - Can contain mixed data types

In [10]:
```
#Example
fruits = ['apple', 'banana', 'cherry']
numbers = [1, 2.5, 3]
mixed = [1, 'hello', True, 3.14]

print(type(fruits))        # <class 'list'>
fruits.append('orange')    # Add item
print(fruits[1])           # banana (indexing)
fruits[1] = 'blueberry'    # Modify item
print(fruits)              # ['apple', 'blueberry', 'cherry', 'orange']
print(len(numbers))        # 3
```

```
<class 'list'>
banana
['apple', 'blueberry', 'cherry', 'orange']
3
```

In [ ]:
```

```

- Tuple (tuple)
  - Ordered, immutable sequence of items
  - Faster than lists for fixed data

In [11]:
```
#Example
```

```
coordinates = (10.0, 20.5)
colors = ('red', 'green', 'blue')
single_item = (42,)  # Note the comma for single-item tuples

print(type(coordinates))   # <class 'tuple'>
print(colors[1])           # green
# colors[1] = 'yellow'     # TypeError (immutable)
x, y = coordinates         # Unpacking
print(y)                   # 20.5
```

```
<class 'tuple'>
green
20.5
```

In [ ]:

5. Mapping Type

- **Dictionary (dict)**
  - Unordered collection of key-value pairs
  - Keys must be immutable (strings, numbers, tuples)

In [12]:
```
#Example
person = {
    'name': 'Alice',
    'age': 30,
    'is_student': False
}

squares = {1: 1, 2: 4, 3: 9}

print(type(person))        # <class 'dict'>
print(person['name'])      # Alice
person['age'] = 31         # Modify value
person['city'] = 'Paris'   # Add new key-value
print(person.keys())       # dict_keys(['name', 'age', 'is_student', 'city'])
print(3 in squares)        # True (key membership)
```

```
<class 'dict'>
Alice
dict_keys(['name', 'age', 'is_student', 'city'])
True
```

In [ ]:

6. Set Types

- **Set (set)**
  - Unordered collection of unique items
  - Mutable version (can add/remove items)

In [13]:
```
#Example
unique_numbers = {1, 2, 3, 3, 2}
vowels = {'a', 'e', 'i', 'o', 'u'}

print(type(unique_numbers)) # <class 'set'>
print(unique_numbers)       # {1, 2, 3} (duplicates removed)
vowels.add('y')            # Add item
vowels.remove('e')         # Remove item
print('a' in vowels)       # True (membership test)
```

```
<class 'set'>
{1, 2, 3}
True
```

In [ ]:

- **Frozen Set (frozenset)**
  - Immutable version of set
  - Can be used as dictionary keys

In [17]:
```
#Example
constants = frozenset([3.14, 2.71, 1.618])
# constants.add(0)  # AttributeError (immutable)
print(constants)
```

```
frozenset({1.618, 2.71, 3.14})
```

In [ ]:

- **Type Conversion Examples**

In [22]:
```
#Example
# int to float
print(float(5))      # 5.0

# float to int (truncates decimal)
rint(int(3.9))      # 3 (not rounded!)

# string to int/float
print(int("42"))     # 42
print(float("3.14")) # 3.14

# bool to int
print(int(True))     # 1

# list to set
print(set([1,2,2,3])) # {1, 2, 3}

# tuple to list
print(list((1,2,3)))  # [1, 2, 3]
```

👨‍💻 Author

✨ Ranjitsingh mugavekar ✨

🧠 MindWave HOLKHAR NAGAR,MUGAON

📈 Beginner to AI Journey

In [ ]: