

LESSON SLIDES

Python Variables and Data Types

Beginner lesson · Names, values and Python's built-in types

By the end of this lesson you can

- 🎯 Create variables with valid, readable names
- 🎯 Recognise int, float, str, bool and None
- 🎯 Use list, tuple, dict and set
- 🎯 Check types with `type()` and convert them
- 🎯 Explain dynamic typing

What is a variable?

- A name that refers to a value
- Created the first time you assign to it
- No type declaration needed: `age = 15`
- Reassign any time: `age = 16`

Naming rules

- Letters, digits and underscores only
- Cannot start with a digit: 2nd_place is invalid
- Case-sensitive: score and Score are different
- Keywords such as class, if and for are not allowed
- Style (PEP 8): snake_case, e.g. total_marks

Good and bad names

Clear names

- student_name
- total_marks
- is_passed
- max_speed

Avoid

- 2nd_place: starts with a digit
- total-score: hyphen means minus
- class: a keyword
- x, tm: unclear meaning

Numbers: int and float

- int: whole numbers such as 42, -7, 0
- float: decimals such as 3.14 and 98.6
- $7 / 2$ gives 3.5; $7 // 2$ gives 3
- Mixing them gives a float: $2 + 1.5$ is 3.5
- $0.1 + 0.2$ shows 0.30000000000000004

Text: str

- Text in single or double quotes: 'Delhi', "Namaste"
- '10' is text, not a number
- '10' + '5' gives '105'
- len('Python') gives 6

True, False and None

- bool has two values: True and False
- Comparisons give a bool: $5 > 3$ is True
- Capitals matter: true gives a NameError
- None means no value yet; its type is NoneType

Collections: list and tuple

- list: ordered and changeable: `marks = [78, 91, 85]`
- `marks.append(66)` adds an item
- Positions start at 0: `marks[0]` is 78
- tuple: ordered but fixed: `point = (3, 4)`

Collections: dict and set

- dict: key-value pairs: `student = {'name': 'Asha', 'age': 14}`
- Look up by key: `student['name']` gives 'Asha'
- set: unique items, no fixed order
- `{2, 3, 3}` becomes `{2, 3}`

Can it change after it is made?

Mutable

- list
- dict
- set

Immutable

- int, float and bool
- str
- tuple
- None

Checking types with type()

- `type(42) → <class 'int'>`
- `type(3.14) → <class 'float'>`
- `type('42') → <class 'str'>`
- `type([1, 2]) → <class 'list'>`

Dynamic typing

- A name can refer to a value of any type
- `x = 5`, then `x = 'five'` is allowed
- The value has the type, not the name
- Check with `type(x)` when unsure

Converting between types

- `int('25')` gives 25
- `float('3.5')` gives 3.5
- `str(45)` gives '45'
- `int('abc')` raises a `ValueError`
- `input()` always returns a `str`

 Example

A canteen bill in code

- 1 `price = 45`
- 2 `qty = int('3')`
- 3 `total = price * qty`
- 4 `print(f'Total: Rs {total}')`
- 5 Output: Total: Rs 135

 Quick check

Which of these is a valid variable name in Python?

A 2nd_place

B total-score

C student_name

D class

Key takeaways

- ✓ A variable is a name that refers to a value
- ✓ Names: letters, digits, underscores; no leading digit
- ✓ Core types: int, float, str, bool, None
- ✓ Collections: list, tuple, dict, set
- ✓ Use `type()` to check and `int()`, `float()`, `str()` to convert