

Python Variables and Data Types: Beginner Lesson Slides

LESSON SLIDES

Python Variables and Data Types

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

 learnbyslides.com

1 / 17

Python Variables and Data Types

SPEAKER NOTES

Ask: who has written any code before, even a single line? Reassure beginners that Python reads close to plain English. Tell them every code example in this deck has been run on Python 3, so they can type it in and check.

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

By the end of this lesson you can

SPEAKER NOTES

Read the objectives aloud. If computers are available, ask learners to open a Python shell now and try each example as you go.

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`

What is a variable?

SPEAKER NOTES

Compare a variable to a label stuck on a value rather than a box. Ask: what does = mean in maths, and what does it mean here? In Python it means assign, not equals; == tests equality.

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`

Naming rules

SPEAKER NOTES

Type `2nd_place = 1` in the shell to show the `SyntaxError`. Ask: why might `score` and `Score` in the same program cause a bug? Python treats them as two separate variables.

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

Good and bad names

SPEAKER NOTES

The first three names in the right column cause errors; the last one runs but is hard to read. Ask learners to rename `tm` to something meaningful. Code is read far more often than it is written.

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`

Numbers: int and float

SPEAKER NOTES

Run `0.1 + 0.2` live: it surprises everyone. Explain that floats are stored in binary, so some decimals are approximations. Ask: which operator would you use to share 7 sweets equally between 2 people?

Text: str

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

Text: str

SPEAKER NOTES

Show that + joins strings instead of adding them. Ask: what would `'10' * 3` give? It repeats the text: `'101010'`. A common error is trying to add a str and an int, which raises a `TypeError`.

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`

True, False and None

SPEAKER NOTES

Type `true` in the shell to show `NameError: name 'true' is not defined`. Ask: where might you use `None`? For example, a phone number that has not been entered yet.

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: list and tuple

SPEAKER NOTES

After `append`, `marks` is `[78, 91, 85, 66]`. Ask: why would you choose a tuple for map coordinates? Because they should not change by accident.

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}`

Collections: dict and set

SPEAKER NOTES

Relate a dict to a school ID card: each label (key) has a value. Ask: how could a set help count how many different states students come from? It removes duplicates automatically.

Can it change after it is made?

Mutable

- `list`
- `dict`
- `set`

Immutable

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

Can it change after it is made?

SPEAKER NOTES

Show that `t[0] = 9` on a tuple raises `TypeError: 'tuple' object does not support item assignment`. The same happens for a `str`. Reassigning a name is allowed; changing an immutable value in place is not.

Checking types with type()

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

Checking types with type()

SPEAKER NOTES

Ask learners to predict `type(3.0)` before running it; the answer is float, because of the decimal point. `type()` is the quickest way to debug a value that behaves strangely.

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

Dynamic typing

SPEAKER NOTES

Run `x = 5`, `type(x)`, then `x = 'five'`, `type(x)`: the result changes from `int` to `str`. Ask: what problem could this cause in a long program? A name may hold a different type than you expect.

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`

Converting between types

SPEAKER NOTES

This is the most common beginner bug: adding a number typed by the user without converting it. Ask: what does `int('3.5')` do? It raises a `ValueError`; use `float()` first.

</> Example

A canteen bill in code

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

OUTPUT Total: Rs 135

A canteen bill in code

SPEAKER NOTES

In a real program qty would come from `input()`, which returns the text '3', so `int()` is needed. Ask: what would print if we forgot `int()`? `'3' * 45` repeats the text, a classic surprise.

 Quick check

Which of these is a valid variable name in Python?

A 2nd_place

B total-score

C student_name

D class

Quick check

SPEAKER NOTES

Answer: C, student_name. 2nd_place starts with a digit, total-score contains a minus sign, and class is a reserved keyword.

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

Key takeaways

SPEAKER NOTES

Recap with one live example per bullet. Then take the five-question quiz on the page and set a short task: store your name, age and three favourite foods in suitable variables and print their types.