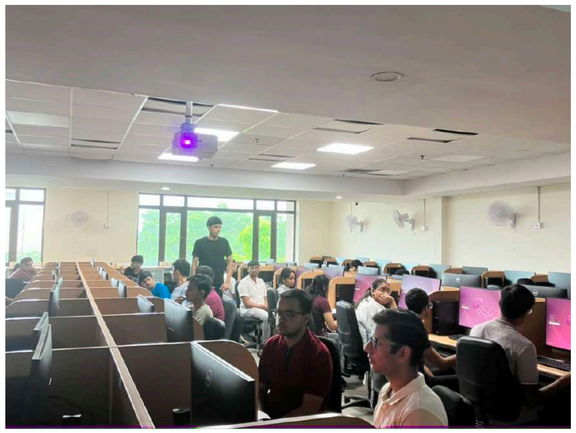
COL1000
Introduction to Programming

Priyanka Golia

Most (if not all) of the content is borrowed from Prof. Subodh Kumar's slides



*COL*1000 so far!!

Hello Python!

```
priyanka@Priyankas-MacBook-Pro ~ % python3
Python 3.13.1 (main, Dec  3 2024, 17:59:52) [Clang
16.0.0 (clang-1600.0.26.4)] on darwin
Type "help", "copyright", "credits" or "license" fo
r more information.
>>> input("name")
namepriyanka
'priyanka'
>>> print("hello", name)
Traceback (most recent call last):
  File "<python-input-1>", line 1, in <module>
    print("hello", name)
                   ^^^^
NameError: name 'name' is not defined
```

To allow program to read the keyboard input

input(“some prompt”)

We took the input, but
didn’t ask program to memorize it

Hello Python!

```
priyanka@Priyankas-MacBook-Pro ~ % python3
Python 3.13.1 (main, Dec  3 2024, 17:59:52) [Clang
16.0.0 (clang-1600.0.26.4)] on darwin
Type "help", "copyright", "credits" or "license" fo
r more information.
>>> name = input("name: ")
name: Priyanka
>>> print("hello!", name)
hello! Priyanka
>>>
```

"name" is now a variable which memorizes a value

One or more value (separated by a comma), a list

Key Concepts

Variables and Objects

Objects: In python, everything is object!

That includes, texts, lists, functions.

Each object in python has three key properties:

1. Type — Describes what kind of thing the object is.
    - 5 is integer, 5.5 is float, “5” is string, “hello” is string, [1,2,3] is list.
2. Value — the content the object holds.
    - 5 is object which is of type integer and has value 5
3. Identity — A unique ID for each object — like its memory address. (More on this later)

Key Concepts

Variables and Objects

Python builds objects using:

- Literals — directly written values:

  10, “hello”, [1,2,3]

- Operators — combine or manipulate other objects:

  3 + 4, “hi" + “there", [1] + [2]

- Functions — create or convert objects:

  int(“5”), list(range(3))

  int(“5”) — Object of type string is changed to object of type integer.

Key Concepts

Variables and Objects

Variable : A “human friendly” name that refers to a value (i.e., to an object). Think of it as a label attached to a box — — the box contains the value, and the label helps you refer to it.

```
x = 10
```

Now, variable x refers to the value 10.

- Variables hold references to objects, not the actual value.
    - Creates an object with value 10 and assigns (“=“) the name **x** to refer it
- You don’t declare the type of variable (like, int x = 10 in other languages!)
    - Type of the variable is the type of the object it is referring to!

```
x = 5
name = "Priyanka"
```

Key Concepts

Variables and Objects

A variable name must:

- Start with a letter (A–Z or a–z) or an underscore _
- Be followed by letters, digits, or underscores
- Be case-sensitive (name and Name are different)
- Not be a Python keyword (like if, while, class, etc.)

ProTip: Use descriptive names: *total_price*, not just *x*

✅

```
x = 5
name = "Alice"
_name = "Hidden"
count2 = 10
```

❌

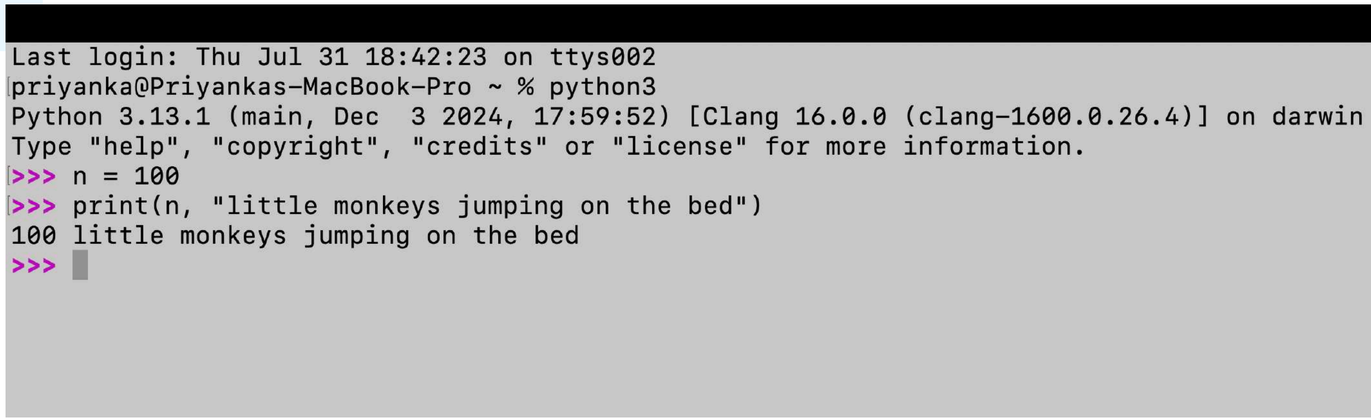
```
2count = 5
my-name = "A"
class = 3
```

Working with Variables

Variables let you store and reuse values in your code:

```
>>> n = 100
>>> print(n, "little monkeys jumping on the bed.")
```

```
100 little monkeys jumping on the bed.
```



Variables in Python:

- Don't need to be declared with a type
- Can be used directly in print statements
- Can be changed at any time

Key Concepts

Statement

A statement is a line of code that tells to do something.

```
x = 10    # This is an assignment statement
print(x)  # This is an function call statement
```

"#" —This is comment, to tell user about the statement, it doesn't "do" anything — doesn't get executed.

Statement could:

- Create a variable
- Perform a calculation } With the help of operators
- Print something
- Make a decision
- Repeat something } Conditional and Loops — more on this later!

Key Concepts

Statement

Type	Example	What it does
Assignment	x= 5	Binds a name (variable) to a value (object)
Expression	3 + 4	Computes a value
Functional Call	print(“hi”)	Calls a function
Conditional	If x > 0:	Runs code only if condition is true
Loop	For i in range (3):	Repeats a block of code.

Quick Quiz?

	Variable	Object	Statement
a= 3	✅	✅	✅
“Hello”	❌	✅	❌
print(“hello”)	❌	✅	✅

Key Concepts

Variables and Objects

Python builds objects using:

- Literals — directly written values:

  10, “hello”, [1,2,3]

- Operators — combine or manipulate other objects:

  3 + 4, “hi" + “there", [1] + [2]

- Functions — create or convert objects:

  int(“5”), list(range(3))

  int(“5”) — Object of type string is changed to object of type integer.

Key Concepts

Types

Primitive

Integer, Float, Boolean

Type	Example	Description
`int`	`5, -2, 100`	Whole numbers
`float`	`3.14, -0.5`	Decimal numbers
`bool`	`True, False`	Logical values

Compound

String, Collection, Range

Type	Example	Description
Str	“Hello”	Collection of characters
`list`	`[1, 2, 3]`	Changeable collection
`dict`	`{"a": 1, "b": 2}`	Key-value pairs (like a mini-database)

Can build new types (and define operations on those) **(more on this later!)**

Built-in conversion (when appropriate):

```
>>> print(int("hello"))
Traceback (most recent call last):
  File "<python-input-7>", line 1, in <module>
    print(int("hello"))
          ~~~^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'hello'
>>>
```

```
>>> x = 3
>>> x = float(x)
>>> print(x)
3.0
>>> x = 4.5
>>> print(int(x))
4
>>> print(int(4.3))
4
>>> print(int(4.7))
4
>>>
```

Key Concepts

Variables and Objects

Python builds objects using:

- Literals — directly written values:

  10, “hello”, [1,2,3]

- Operators — combine or manipulate other objects:

  3 + 4, “hi" + “there", [1] + [2]

- Functions — create or convert objects:

  int(“5”), list(range(3))

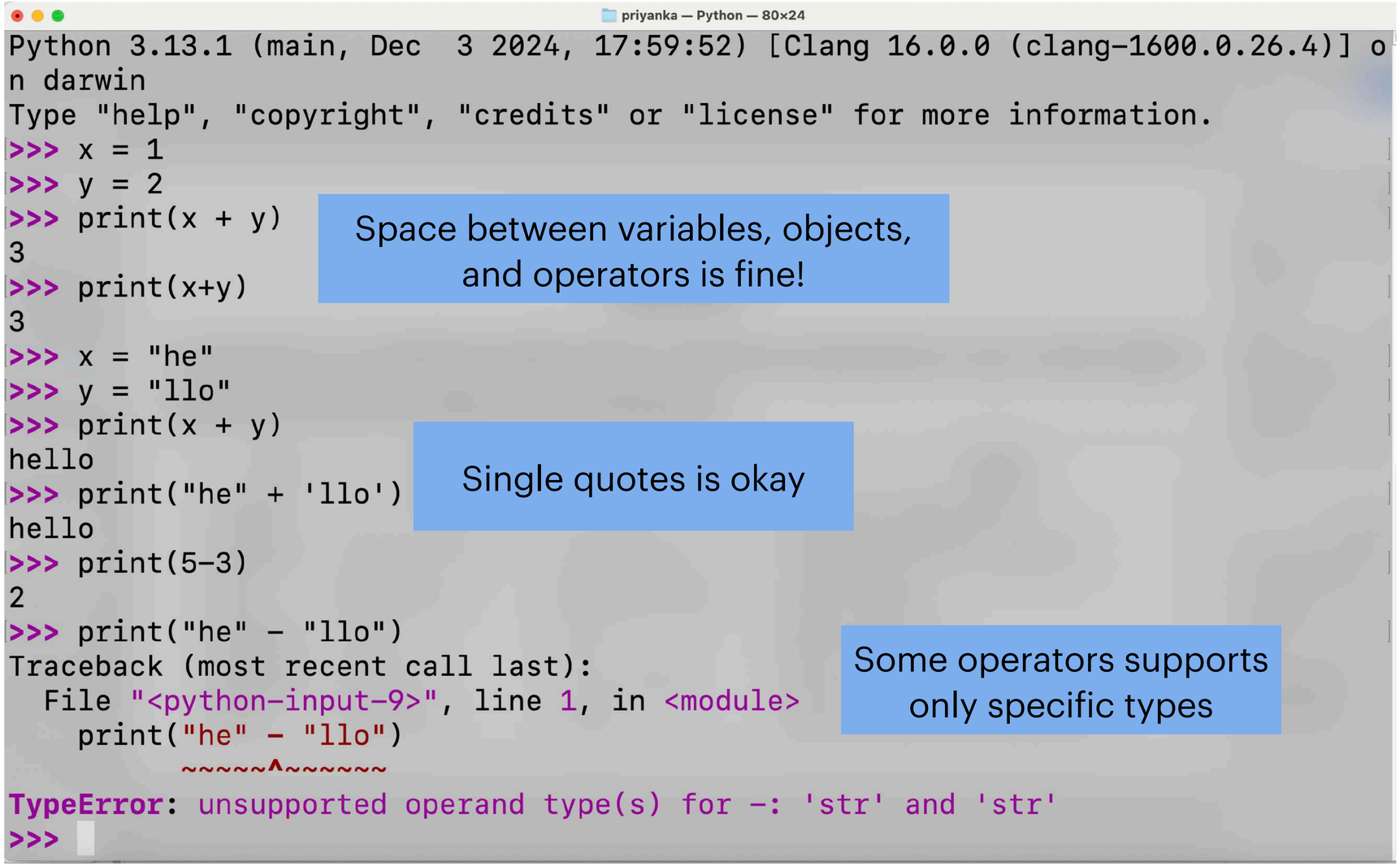
  int(“5”) — Object of type string is changed to object of type integer.

Key Concepts

Operators

- An operator is a symbol that tells Python to perform an operation on one or more objects.
- Operators create new objects from existing ones.

Type	Operator	Example	Meaning
Arithmetic	`+, -, *, /, //, %, **`	`3 + 4, 5 * 2`	Add, subtract, multiply, divide, etc.
Comparison	`==, !=, <, >, <=, >=`	`x == 10, x < y`	Check conditions (returns `True`/`False`)
Logical	`and, or, not`	`True and False`	Combine conditions
Assignment	`=, +=, -=`	`x += 1`	Update variable values
Membership	`in, not in`	`"a" in "cat"`	Check if value exists
Identity	`is, is not`	`x is y`	Check if two names refer to the same object


priyanka — Python — 80×24
Python 3.13.1 (main, Dec 3 2024, 17:59:52) [Clang 16.0.0 (clang-1600.0.26.4)] o
n darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 1
>>> y = 2
>>> print(x + y)
3
>>> print(x+y)
3
>>> x = "he"
>>> y = "llo"
>>> print(x + y)
hello
>>> print("he" + 'llo')
hello
>>> print(5-3)
2
>>> print("he" - "llo")
Traceback (most recent call last):
File "<python-input-9>", line 1, in <module>
print("he" - "llo")
~~~~~^~~~~~~
TypeError: unsupported operand type(s) for -: 'str' and 'str'
>>>
Space between variables, objects, and operators is fine!
Single quotes is okay
Some operators supports only specific types
~~~~~

```
>>> print (7/3)
2.3333333333333335
>>> print(7//3)
2
>>> print(-7//3)
-3
>>> print(-7.0//3)
-3.0
>>> print(7.0//3)
2.0
>>>
```

// — integer division.
Lower down to the closest integer.
can handle negatives and “floats”!

```
>>> print(2 +"hello")
Traceback (most recent call last):
  File "<python-input-0>", line 1, in <module>
    print(2 +"hello")
          ~~^~~~~~~~
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> print(2 * "hello")
hellohello
>>>
```

```
>>> print(4 ** 2)
16
>>> print(4 * 2)
8
>>> print("hello" * 2)
hellohello
>>> print("hello" ** 2)
Traceback (most recent call last):
  File "<python-input-3>", line 1, in <module>
    print("hello" ** 2)
          ~~~~~~~~^^~~
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
>>>
```

** — power operator, $a**b$ is a^b