



# Diploma in Python Programming

Basic features of Python programming



Examine three basic Python data types and their use cases



Introduce variables and literals and show how they are applied



Explain Python statements and comments



Discuss the importance of indentations in Python



## Objectives

# Important facts!

- Everything in Python programming is an object
- Every value in Python has a data type.
- Data types are called ‘instances of classes’
- An ‘instance of a class’ is then referred to as an ‘object’



# Basic data types





# Three basic data types

- Number data types
- String data types
- Boolean data types



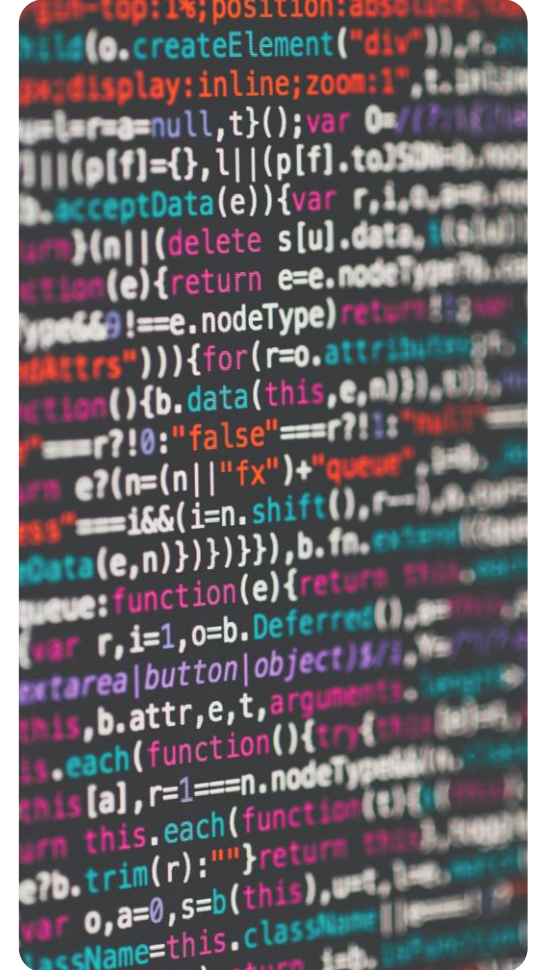
01



02



03





## Number data types

- Integers
- Floating-point numbers
- Complex numbers

**DID YOU  
KNOW?**

# Code process

- Input
- Process
- Output

Input (1)

`print(2 + 3)` (2)

Output (3)

5 (4)

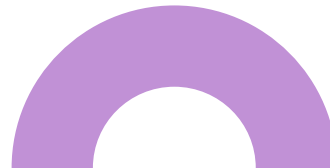




# Integers

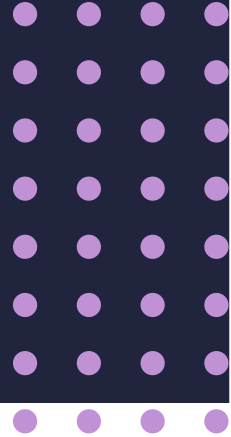


- Positive and negative whole numbers (including 0)
- No limit to length of an integer value
- Can be used for standard addition, subtraction, division and multiplication operations in Python





# Integers



## Addition

---

Input	(1)
<code>print(2 + 3)</code>	(2)
Output	(3)
5	(4)

## Subtraction

---

Input	(5)
<code>print(5 - 2)</code>	(6)
Output	(7)
3	(8)

## Division

---

Input	(13)
<code>print(10/2)</code>	(14)
Output	(15)
5	(16)

## Multiplication

---

Input	(9)
<code>print(3 * 2)</code>	(10)
Output	(11)
6	(12)



# Integers



## Type() Function

---

Input	(17)
<code>print(type(4))</code>	(18)
Output	(19)
<code>&lt; class<sup>0</sup>int<sup>0</sup> &gt;</code>	(20)

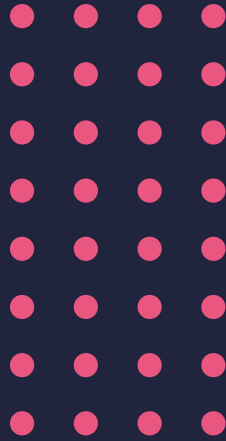
## Class integer

---

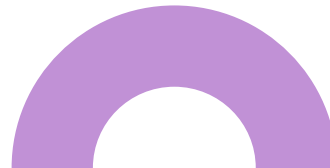
Input	(21)
<code>print(isinstance(2,int))</code>	(22)
Output	(23)
True	(24)



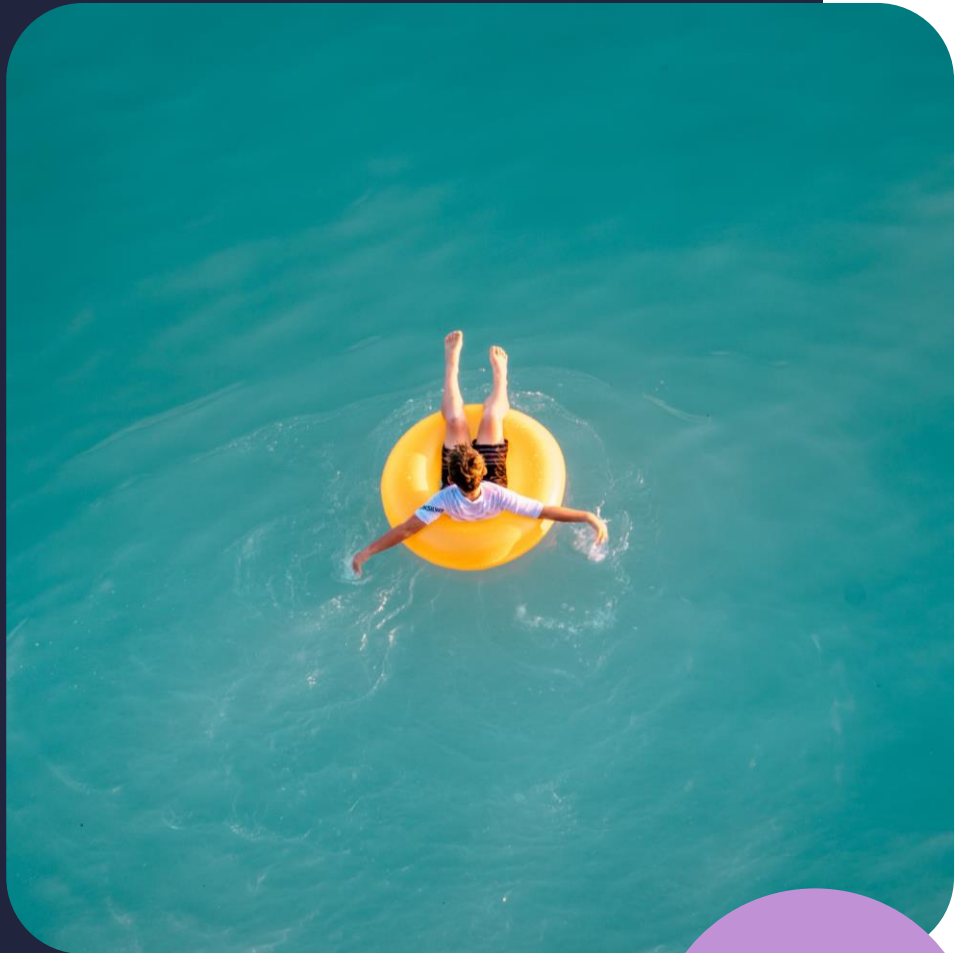
# Floating-point numbers



- Numbers are values with decimal points
- For example, 5.1



# Floating-point numbers



## Type[]

Input (25)

print(type(5.0)) (26)

Output (27)

< class<sup>0</sup>float<sup>0</sup> > (28)

Input (29)

print(type(3.2)) (30)

Output (31)

< class0float0 > (32)

## Isinstance[]

Input (33)

print(isinstance (34)  
(5.0,float))

Output (35)

True (36)

Input (37)

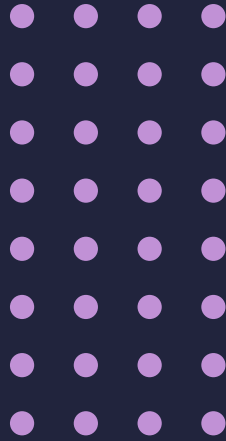
print(isinstance (38)  
(5.0,int))

Output (39)

False (34)



# Complex numbers



**<real part> + <imaginary part>j**

---

Input (41)

`print(type(2 + 3j))` (42)

Output (43)

`< class0complex0 >` (44)

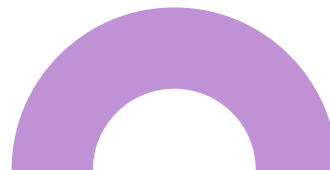




# Strings data types



- Sequences of character data
- Immutable
- Represented by single or double quotes
- Triple quotes for multi-line strings



# Examples of a string



## Single quote

---

Input (45)

`print('Hello I am a string')` (46)

Output (47)

Hello I am a string (48)

# Examples of a string



## Confirmation

---

Input (49)

Print(type('Hello I am a string')) (50)

Output (51)

<class 'string'> (52)



# Examples of a string



## Double quote

---

Input (53)

Print(type('Hello I am a string')) (54)

Output (55)

<class 'string'> (56)

# Examples of multi-line strings



## Triple quote

---

Input		(57)
print(type('Hello I		(58)
am a		(59)
string'))		(60)
Output		(61)
<class'string'>		(62)

# Examples of multi-line strings



## Double triple quote

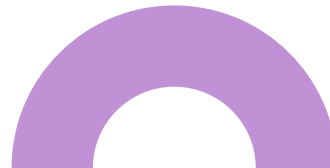
```
Input (63)
print(type("""Hello I (64)
          am a (65)
          string""")) (66)

Output (67)
<class'0string'0> (68)
```



# Boolean data types

- Objects have TRUE or FALSE value





# Examples of a Boolean data type



*Input* (69)

*print(type(True))* (70)

*Output* (71)

*<class 'bool'>* (72)

and

*Input* (73)

*print(type(False))* (74)

*Output* (75)

*<class 'bool'>* (76)

## DID YOU KNOW?

You can convert between data types using different type conversion functions such as *int()*, *float()* and *str()*.



# Examples of a Boolean data type



## Data type

---

Input	(77)
<code>print(float(5))</code>	(78)
Output	(79)
5.0	(80)

## Truncate

---

Input	(81)
<code>print(int(10.6))</code>	(82)
Output	(83)
10	(84)

# Examples of a Boolean data type



## Compatible values

---

Input	(85)
<code>print(float(03.40))</code>	(86)
Output	(87)
3.4	(88)

## String data type

---

Input	(89)
<code>print(str(34))</code>	(90)
Output	(91)
<code>0340</code>	(92)





# Python variables and literals



Pic needs to be of containers



**A variable is a named location with memory that stores data.**

# Examples of a Boolean data type



## Containers

---

Input	(93)
num = 5	(94)
print(num)	(95)
Output	(96)
5	(97)

# Examples of a Boolean data type



## Value assignment

Input	(98)
num = 5	(99)
num = 10	(100)
print(num)	(101)
Output	(102)
10	(103)



# Variables

Variables can contain any type of data types

```
surname = 'Smith' (104)
```



# Examples of a Boolean data type



## Containers

---

Input	(105)
surname = 'Smith'	(106)
surname = 'Denga'	(107)
print(surname)	(108)
Output	(109)
Denga	(110)

# Examples of changing a string data type to a number or Boolean data type

## Example 1

---

Input  
surname = 'Smith'  
surname = 10  
print(surname)  
Output  
10

(111)  
(112)  
(113)  
(114)  
(115)  
(116)

## Example 2

---

Input  
surname = 10  
surname = 'Smith'  
print(surname)  
Output  
Smith

(117)  
(118)  
(119)  
(120)  
(121)  
(122)

## Example 3

---

Input  
surname = 10  
surname = True  
print(surname)  
Output  
True

(123)  
(124)  
(125)  
(126)  
(127)  
(128)



**DID YOU  
KNOW?**

**You can assign the  
same value to  
multiple variables**

## Example

Input	(129)
age,name,salary = 77	(130)
print(age)	(131)
print(num )	(132)
print(salary)	(133)
Output	(134)
77	(135)
77	(136)
77	(137)

# Examples of assigning the same value to multiple variables



Input

```
age,name,salary = 77,"Smith",1000
```

```
print(age)
```

```
print(num)
```

```
print(salary)
```

Output

77

Smith

1000

(138)

(139)

(140)

(141)

(142)

(143)

(144)

(145)

(146)

# 6 rules to remember when naming your variables

01

Variable names should have a combination of letters in uppercase (A to Z), or lowercase (a to z), or digits (0 to 9) AND you use an underscore (\_) when you are combining two different words.

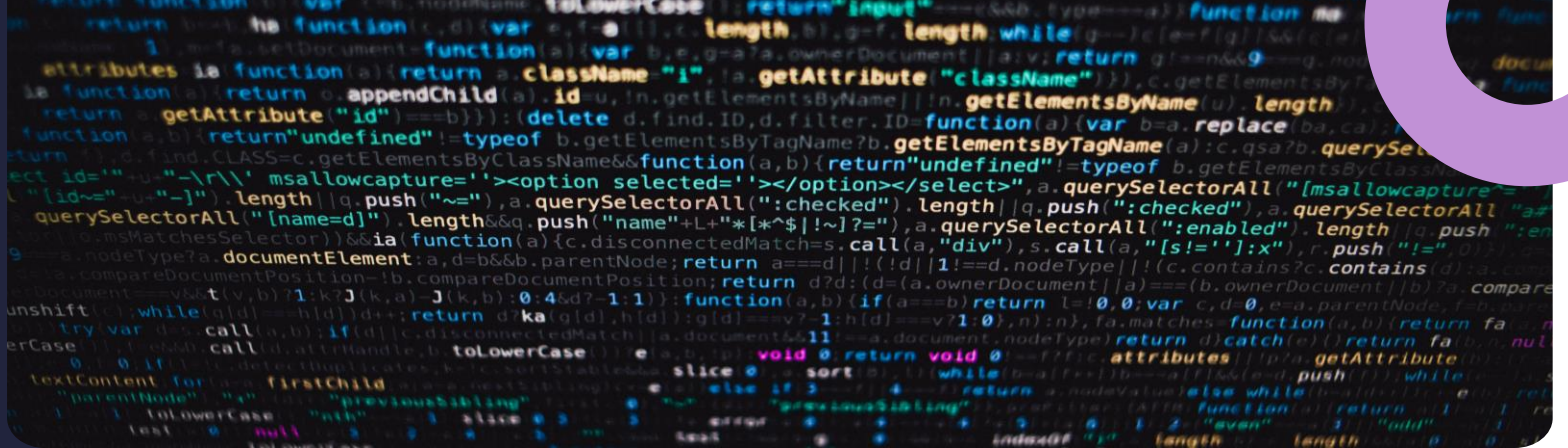
lucky\_number (148)

MySurname (149)

LuckyNumber (150)

mySurname (151)

luckyName (152)



# 6 rules to remember when naming your variables

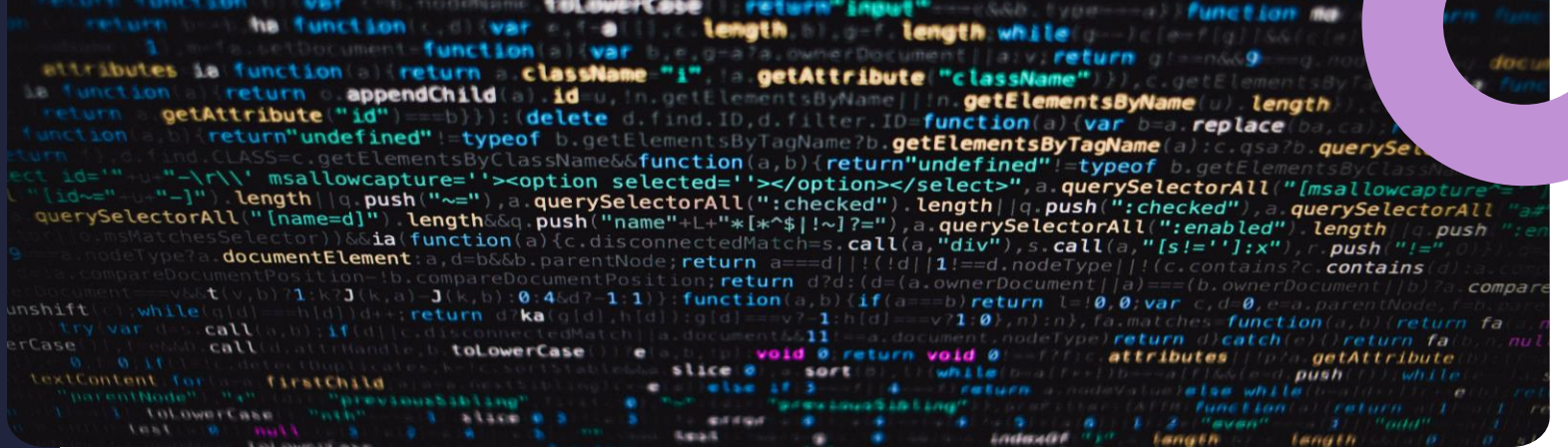
02

Variable names should make sense. For example, if you want to create a variable that will store your age, creating your variable as.

age = 77 (153)

and not

a = 77 (154)



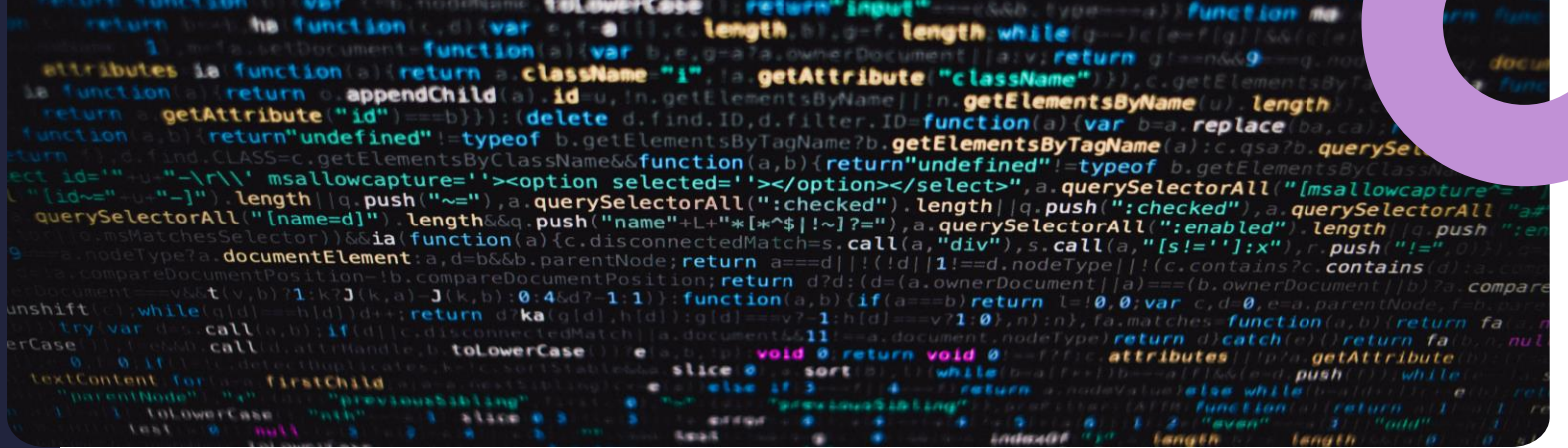
# 6 rules to remember when naming your variables

03

Variable names with two words must include an underscore to separate the words. Let's say you want to create a variable name for your lucky number, you can name your variable like this.

lucky\_number = 10

(155)





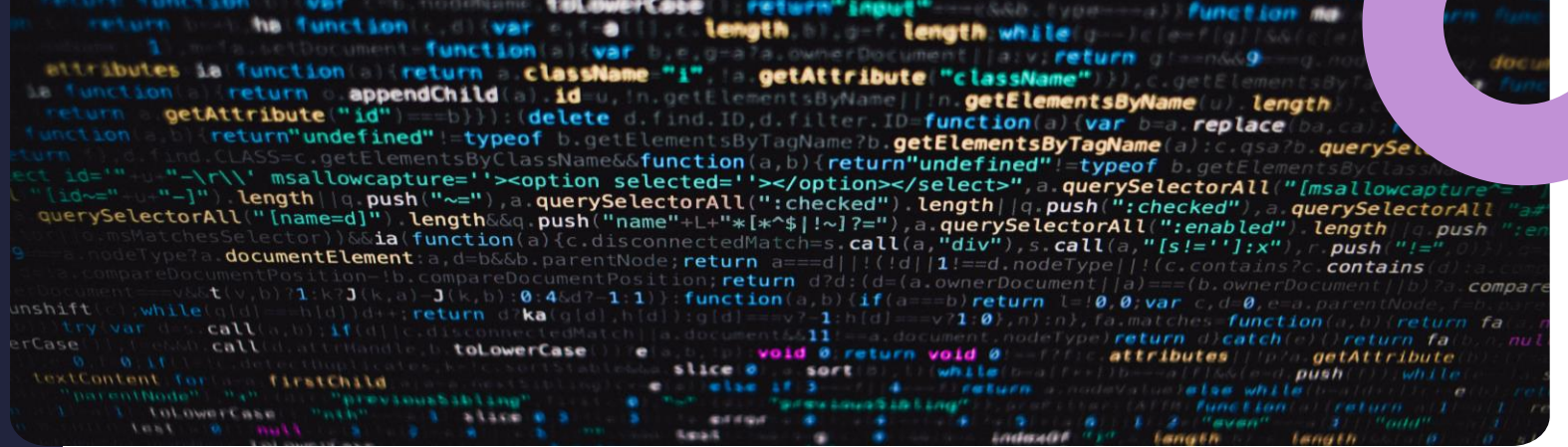
# 6 rules to remember when naming your variables

04

Variable names in Python are case sensitive. For example, the two variable names, Lucky-number and lucky-number will be treated differently in Python even though they are pronounced the same.

GRAVITY = 9.8 (156)

SPEED\_OF\_LIGHT = 299792458 (157)



[illegible]

05



96 year

4people



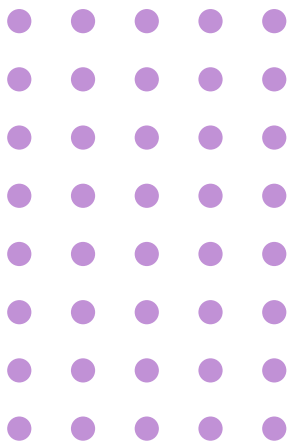
```
function function(a) {  
    return b; }  
    he function(c,d){ var e,f,a.length,g,h,q.f.length while(g-->c[e-f]=g(f=g+1));  
        i,j,e,f,c.setDocument=function(a){var b,c,g;a=a.ownerDocument||a|v;return g==null?g.no  
attributes is function(a){return a.className=="I";},a.getAttribute("className"));c.getElementsByT  
is function(a){return c.appendChild(a).id=u,n.getElementsByIdName||(n.getElementsByTagName(u).length  
return a.getAttribute("id")==b)}:(delete d.find.ID,d.filter.ID=function(a)(var b=a.replace(ba,ca);  
function(a,b){return undefined!=typeof b.getElementsByTagName?b.getElementsByTagName(a):c.qsa?b.querySel  
return a.c.find.CLASS=c.getElementsByClassName&&function(a,b){return undefined!=typeof b.getElementsByClassNameByClassNa  
ect id="'+'-'\r\\' msallowcapture='>option selected='</option></select">,a.querySelectorAll(["msallowcapture"  
("[id~='"+'-"]").length]|q.push("<"),a.querySelectorAll(":checked").length]|q.push(":checked"),a.querySelectorAll("a?  
querySelectorAll(["name=d"].length&q.push("name"+L+"*[*$]?")=),a.querySelectorAll(":enabled").length]|q.push("<en  
9     .matchesSelector)&&iia(function(a){c.disconnectedMatches.call(a,"div"),s.call(a,"[s]='':x"),r.push("!");  
e.compareDocumentPosition-lb.compareDocumentPosition;return d;d=(d=a.ownerDocument)|j==(b.ownerDocument)|jb.a.compare  
document=v&t,v,l:kJ(k,a)-J(k,b):0.4&d7-1}):function(a,b){if(a===b)return l=[0,0];var c,d=0,e=a.parentNode,f=b.p  
unshift(i).while(g[d]!==h[d]&&d++return ka[g[d],h[d]]:gld]-v7-1:h[d]==v7-1:n):n),fa.matches=function(a,b){return fa.m  
try var d=a.call(a,b);if(d){c=c.disconnectedMatch[a.document+ll]=a.document.nodeType)return d}catch(e){return fa.nu  
erCase}f=f.b.call(d,handleB,c.toLowerCase,e,void 0)return void 0}-ffff.attributes}|(p?a.getAttribute(n)  
0,0;if(!c.determineAttributes,k=k.sortStable);a=a.b,p(void 0)-ffff.attributes}|(p?a.getAttribute(n)  
textContent for a.firstChild  
"parentNode":"a","previousSibling": " "  
1 if 1.toLowerCase "nth" else 0,3 no 3 endgrd 0 + * indexof "o" length "even" 1 "odd" 0 1
```

06

lucky\_#  
@lucky\_number  
@lucky\_#

@lucky\_number

@lucky\_#



# Important tip

Python special functionalities

False	def	if	raise
None	del	import	return
True	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

# Object identity



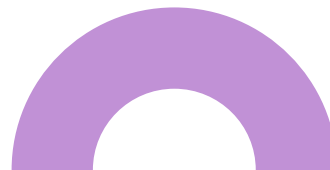
Input	(158)
num	(159)
print(id(num))	(160)
Output	(161)
94668063081792	(162)

Please note that the value of this identity number will differ from program to program.



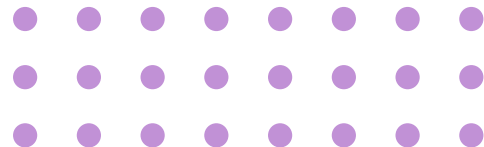
# What is a Literal?

- Numeric literals
- String literals
- Boolean literals



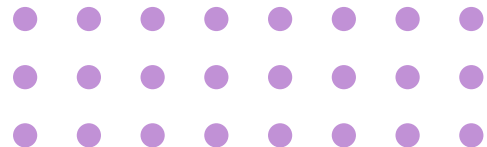
# Numeric Literals

- Integer
- Float
- Complex



# Integer literal

- **Binary literal**
- **Decimal literal**
- **Octal literal**
- **Hexadecimal literal**



Input	(163)
binary_literal = 0b1010	(164)
decimal_literal = 100	(165)
octal_literal = 0o310	(166)
hexa_decimal = 0 × 12c	(167)
print(binary_literal)	(168)
print(decimal_literal)	(169)
print(octal_literal)	(170)
print(hexa_decimal)	(171)
Output	(172)
10	(173)
100	(174)
200	(175)
300	(176)
	(177)

# Float literal



Input	(178)
<code>float_literal = 2.5</code>	(179)
<code>print(float_literal)</code>	(180)
Output	(181)
<code>2.5</code>	(182)



# Complex literal



Input	(183)
<code>complex_literal = 2 + 3.4j</code>	(184)
<code>print(complex_literal)</code>	(185)
Output	(186)
<code>2 + 3.4j</code>	(187)

# String Literals



Input

```
print(0Hello I am a string0)
```

Output

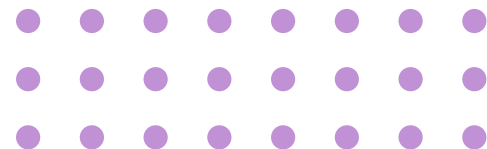
```
I am a string
```

(188)

(189)

(190)

(191)



# Boolean Literals



Input

```
boolean_literal = True
```

```
print(boolean_literal)
```

Output

```
True
```

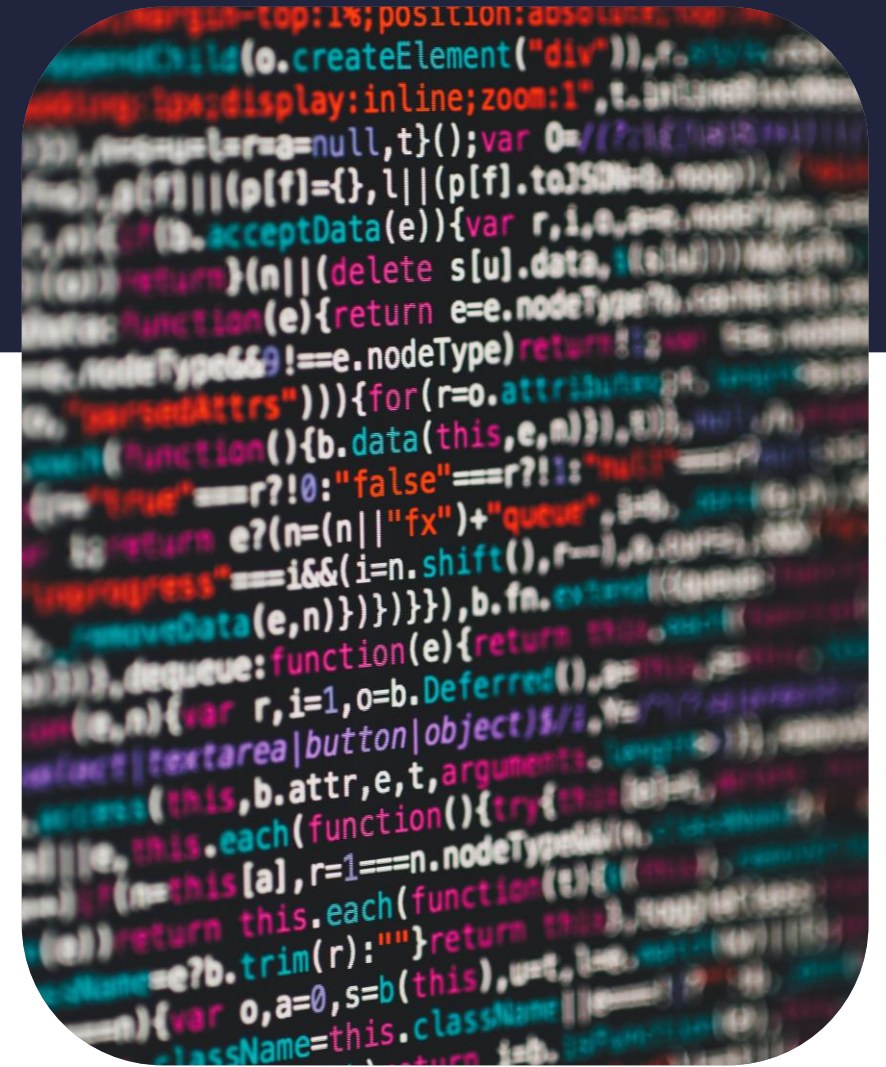
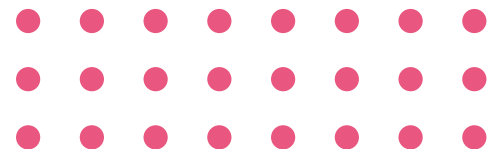
(192)

(193)

(194)

(195)

(196)





# Python Statement

---

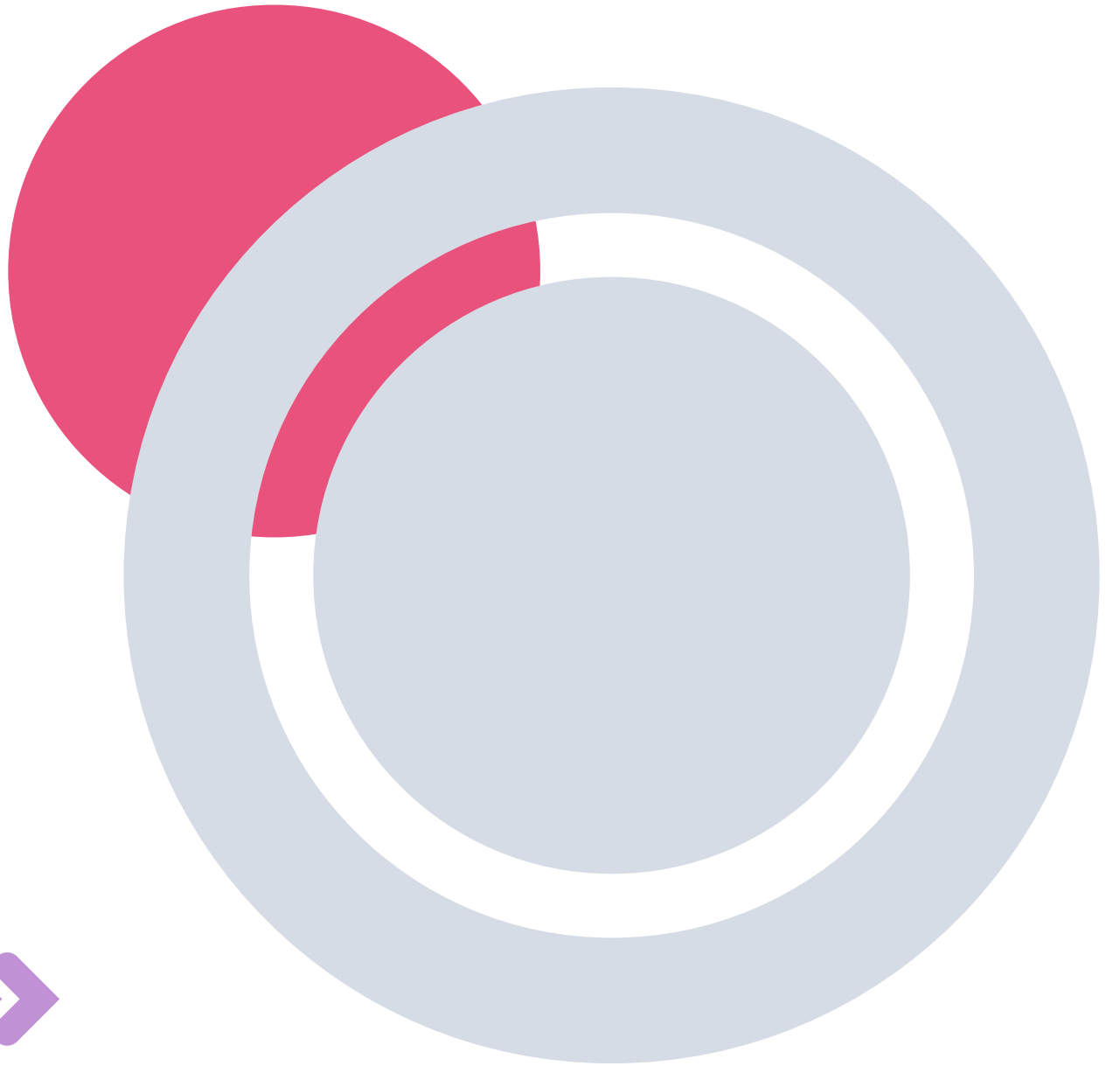
Coming up  
In Lesson **3**

---

# Python Statements

A Python statement is an instruction that a Python interpreter can execute.

**surname = 'Smith'** (197)



# Multi-line Statements



Input	(198)
num = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8	(199)
print(num)	(200)
Output	(201)
36	(202)

# Multi-line Statements



Input	(203)
num = 1 + 2 + 3 + \	(204)
4 + 5 + 6 + \	(205)
7 + 8	(206)
print(num)	(207)
Output	(208)
36	(209)



# Multi-line Statements



Input	(210)
num = 1 + 2 + 3 +	(211)
4 + 5 + 6 +	(212)
7 + 8	(213)
print(num)	(214)
Output	(215)
SyntaxError: invalid syntax	(216)

# Multi-line Statements

## Example 1



Parenthesis (), {}, []

Input	(217)
num = (1 + 2 + 3 +	(218)
4 + 5 + 6 +	(219)
7 + 8)	(220)
print(num)	(221)
Output	(222)
36	(223)

# Multi-line Statements

## Example 2



Input	(224)
num = {1 + 2 + 3 +	(225)
4 + 5 + 6 +	(226)
7 + 8}	(227)
print(num)	(228)
Output	(229)
36	(230)

# Multi-line Statements

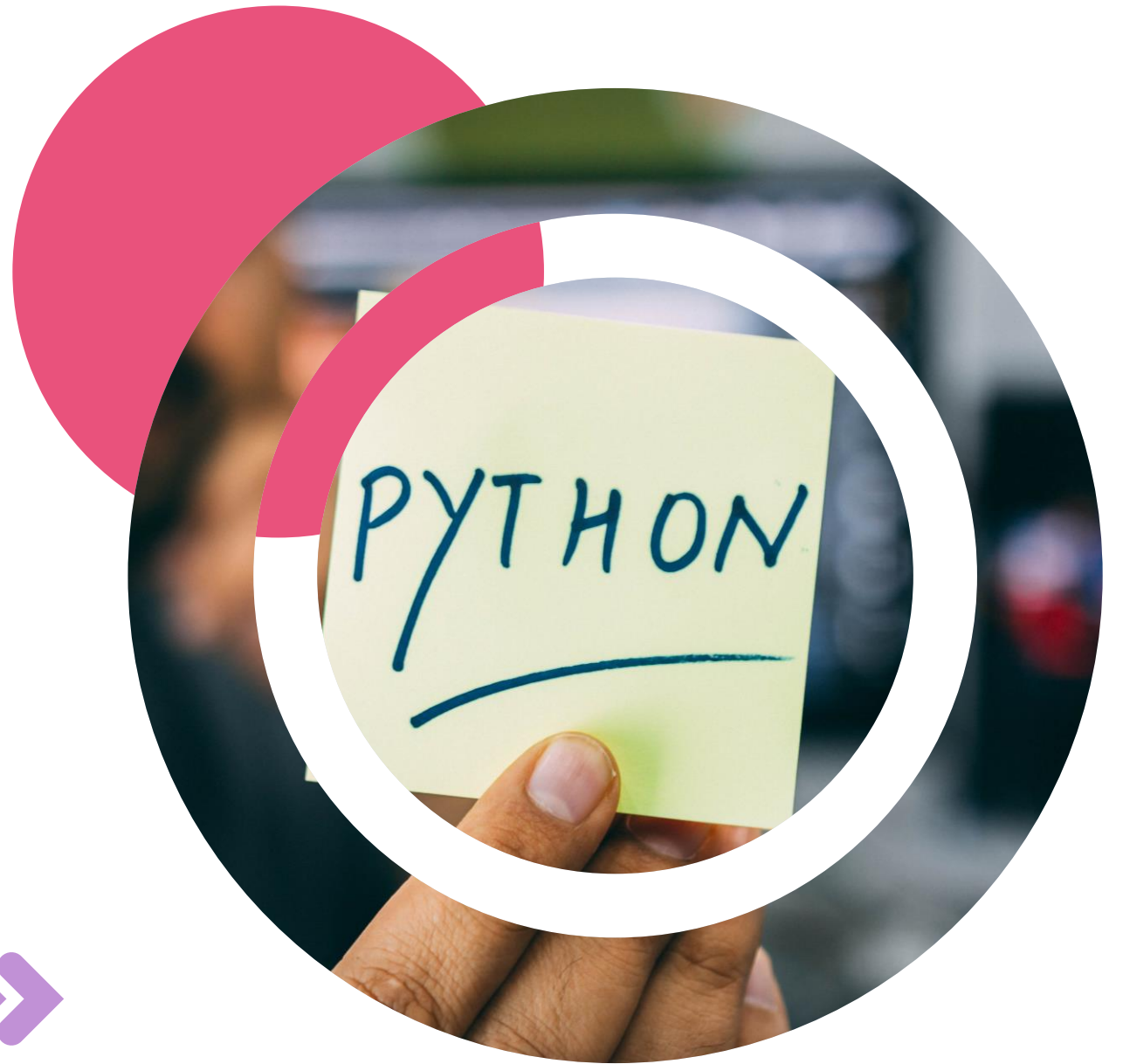
## Example 3



Input	(231)
num = [1 + 2 + 3 +	(232)
4 + 5 + 6 +	(233)
7 + 8]	(234)
print(num)	(235)
Output	(236)
36	(237)

# Python Indentation

Python uses indentation to define and write a code block.



# Multi-line Statements



Input	(238)
if True :	(239)
Num=10	(240)
print(num)	(241)
Output	(242)
10	(243)

# Multi-line Statements



Input	(244)
if True:	(245)
num=10	(246)
print(num)	(247)
Output	(248)
IndentationError	(249)

# Multi-line Statements



Input	(250)
<code>if True; num=10; print(num)</code>	(251)
Output	(252)
10	(253)



# Multi-line Statements



Input

```
if True:
```

```
    num=10
```

```
    print(num)
```

Output

10

(254)

(255)

(256)

(257)

(258)

(259)



# IndentationError

A common error when indentation is not consistent.



# Python comments

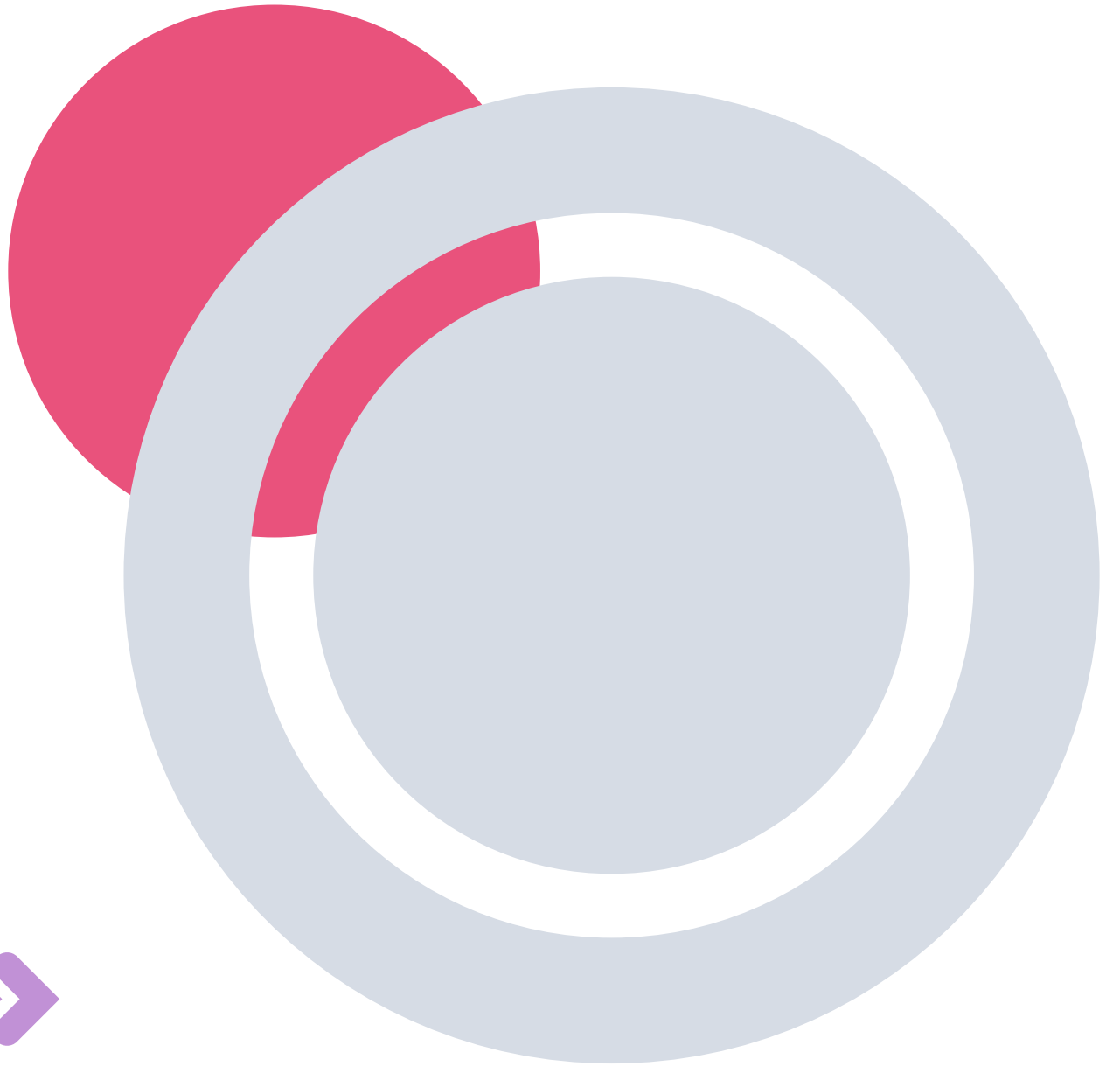
# Writing a comment within a code



Input	(260)
#This is just a comment	(261)
#This is just a comment	(262)
num=10	(263)
print(num)	(264)
Output	(265)
10	(266)

# Multiple line comments

Sometimes we have comments  
that extend over multiple lines.



# Writing a comment within a code block

## Example 1



Input	(267)
#First comment	(268)
#Second comment	(269)
#Third comment	(270)
num='Hello'	(271)
print(num)	(272)
Output	(273)
Hello	(274)

# Writing a comment within a code

## Example 2



Input	(275)
<sup>000</sup> First comment	(276)
Second comment	(277)
Third comment <sup>000</sup>	(278)
num="Hello" <sup>00</sup>	(279)
print(num)	(280)
Output	(281)
Hello	(282)



# Writing a comment within a code

## Example 3



Input	(283)
000000First comment	(284)
Second comment	(285)
Third comment000000	(286)
num="Hello"00	(287)
print(num)	(288)
Output	(289)
Hello	(290)