APSC-5984 Lab 2: Python Basics I

Due: 2023-01-30 (Monday) 23:59:59

0. Overview

In this lab, you will learn basic Python syntax. We will cover the essential concepts of Python:

- Variables and data types
- Operators
- Control structures: If-else statements

You will need to open the `labs/lab_02/assignment.ipynb` file in VS Code and follow the instruction to complete this lab assignment.

(Back to top)

1. Variables

There are several rules for naming variables in Python:

- A variable name can only contain letters (`A-Za-z`), numbers(`0-9`), and underscores(`_`).
- A variable name is case sensitive. For example, `first_var` and `First_var` are two different variables.
- There are two ways to name a variable: `snake_case` and `camelCase`. In `snake_case`, all letters are lowercase and words are separated by underscores. In `camelCase`, the first letter of each word is capitalized. For example, `first_var` and `firstVar` are both valid variable names.

Things you cannot do:

- A variable name cannot start with a number.
- A variable name cannot contain spaces.

Examples:

- Valid variable names: `first_var`, `firstVar`, `first_var_1`, `firstVar1`, `first_var_1_2_3`, `firstVar123`
- Invalid variable names: `1st_var`, `first var`, `first-var`

In Python, you can use `=` to assign a value to a variable. For example, you can assign an integer value `3` to a variable `first_var` by running the following code:

```
first_var = 3
```

And we can print the value of `first_var` by running the following code:

```
print(first_var) # 3
```

(Back to top)

2. Data Types

There are several data types in Python:

- `int`: an integer number, e.g., `3`, `0`, `-1`
- `float`: a floating point number, e.g., `3.14`, `0.0`, `-1.0`
- `bool`: a boolean value, e.g., `True`, `False`
- `str`: a string, e.g., `"hello"`, `"2023-01-30"`

```
var_int = 3
var_float = 3.14
var_bool = True
var_str = "hello"
```

You can use `type()` function to check the type of a variable. For example, you can check the type of `a` by running the following code:

```
print(type(var_float)) # <class 'float'>
```

It is possible to convert a variable from one type to another. For example, you can convert a `float` to an `int` by running the following code:

From `float` to `int`:

```
var_float = 3.14
var_int = int(var_float)
print(var_int) # 3
```

From `int` to `string`:

```
var_int = 3
var_str = str(var_int)
print(var_str) # "3"
```

(Back to top)

3. Operators

3.1 Arithmetic Operators

In Python, you can use arithmetic operators to perform arithmetic operations. For example, you can use the `+` operator to add two numbers, and use the `-` operator to subtract two numbers. We use `a = 7` and `b = 4` as examples in the following table to illustrate the usage of arithmetic operators.

Operator	Description	Example	Result
`+`	Addition	`a + b`	`11`
`-`	Subtraction	`a - b`	`3`
`*`	Multiplication	`a * b`	`28`
`**`	Exponentiation	`a ** b`	`2401`
`/`	Division	`a / b`	`1.75`
`//`	Floor division	`a // b`	`1`
`%`	Modulus	`a % b`	`3`

3.2 String Operators

You can also use operators to perform operations on strings. For example, you can use the `+` operator to concatenate two strings, and use the `*` operator to repeat a string. We use `a = "hello"` and `b = "world"` as examples in the following table to illustrate the usage of string operators.

Operator	Description	Example	Result
`+`	Concatenation	`a + b`	`"helloworld"`
`*`	Repetition	`a * 3`	`"hellohellohello"`

3.3 Comparison Operators

You can use comparison operators to compare two values. For example, you can use the `==` operator to check if two values are equal. We use `a = 7` and `b = 4` as examples in the following table to illustrate the usage of comparison operators.

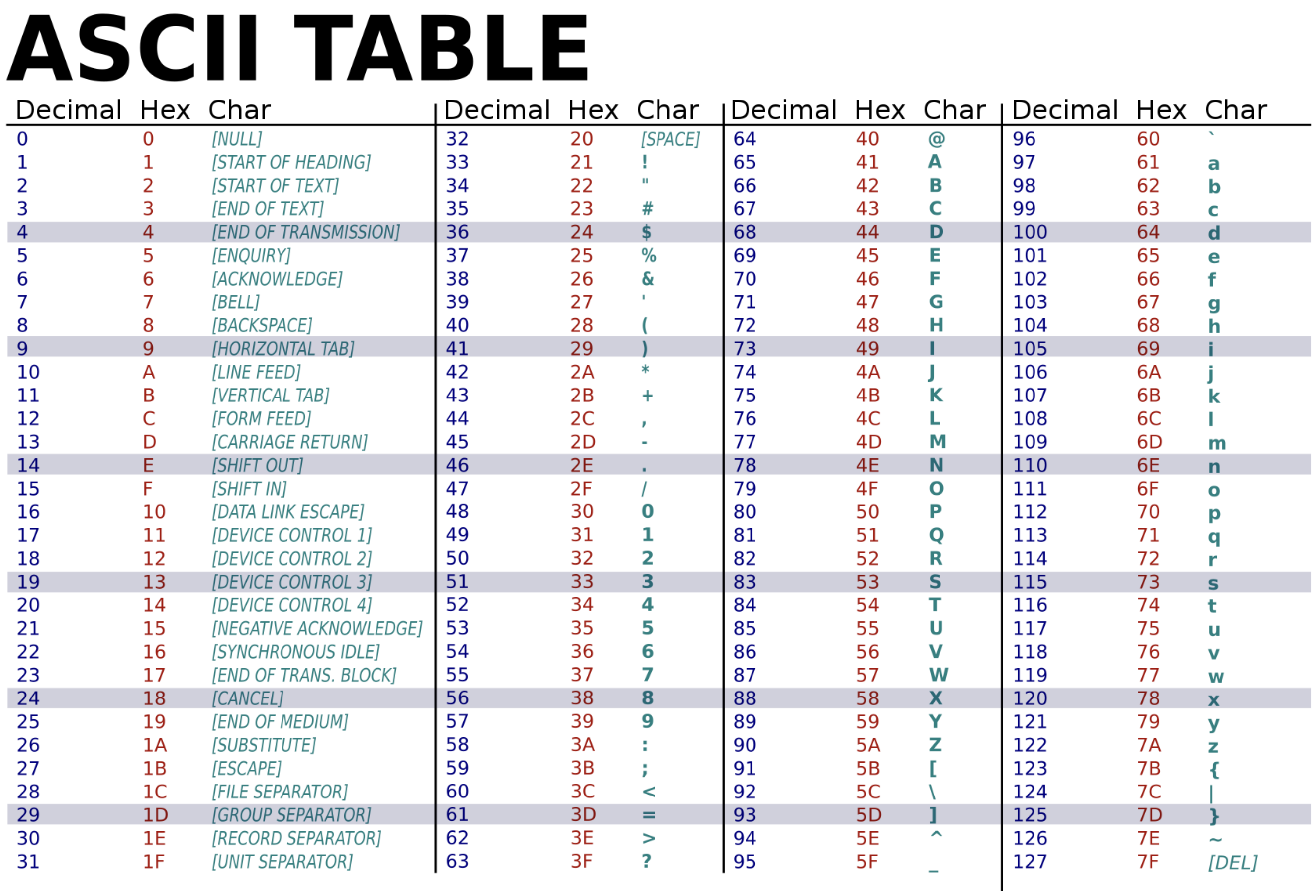
Operator	Description	Example	Result

Operator	Description	Example	Result
`==`	Equal to	`a == b`	`False`
`!=`	Not equal to	`a != b`	`True`
`>`	Greater than	`a > b`	`True`
`<`	Less than	`a < b`	`False`
`>=`	Greater than or equal to	`a >= b`	`True`
`<=`	Less than or equal to	`a <= b`	`False`

Similar logic applies to strings. For example, you can use the `==` operator to check if two strings are equal. We use `a = "hello"` and `b = "world"` as examples in the following table to illustrate the usage of comparison operators. We are also able to use the `>` and `<` operators to compare two strings. The comparison is based on the ASCII table and the alphabetical order of the strings. For example, `"hello"` is smaller than `"world"` because the first letter of `"hello"` is `h` (ASCII code 104) is smaller than the first letter of `"world"` is `w` (ASCII code 119).

```
a = "hello"
b = "world"
print(a != b) # True
print(a > b) # False
```

ASCII Table from Wikipedia:



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

3.4. Logical Operators

You can use logical operators to perform logical operations. For example, you can use the `and` operator to check if two conditions are both `True`. We use `a = True` and `b = False` as examples in the following table to illustrate the usage of logical operators. It is noted that some operators are interchangeable, e.g., `and` and `&&`.

Operator	Description	Example	Result
`and`	Logical AND	`a and b`	`False`
&&	Logical AND	`a && b`	`False`
`or`	Logical OR	`a or b`	`True`
||	Logical OR	`a || b`	`True`
`not`	Logical NOT	`not a`	`False`
!	Logical NOT	`!a`	`False`

(Back to top)

4. Conditional Statements

You can use `if` statement to check a condition. For example, you can check if `10` is greater than `0` by running the following code:

```
if 10 > 0:
    print("10 is greater than 0")

# Output:
# 10 is greater than 0
```

It is noted that the Python syntax requires no parentheses around the condition, and the code block in the `if` statement is indented by a tab (4 spaces). Inappropriate indentation will cause syntax errors. For example, the following code will cause a syntax error:

```
if 10 > 0:
print("10 is greater than 0")

# Output:
# IndentationError: expected an indented block
```

You can use `else` statement to run a block of code if the previous conditions are not satisfied. Although the `else` statement is optional. It is recommended to use `else` statement to avoid confusion. FOr example, you can check if `7` is even or odd by running the following code:

```
if 7 % 2 == 0:
    print("7 is even")
else:
    print("7 is odd")

# Output:
# 7 is odd
```

This code should print `"7 is odd"` because `7` is not divisible by `2`.

When it comes to multiple conditions, you can use `elif` statement to check as many conditions as you want.

```
if 27 % 2 == 0:
    print("27 is even")
elif 27 % 3 == 0:
    print("27 is divisible by 3")
else:
    print("27 is neither even nor divisible by 3")

# Output:
# 27 is divisible by 3
```

This code should print `"27 is divisible by 3"` because `27` is divisible by `3` but not `2`.

One more thing, the if-else statements can be rewritten using the ternary operator. For example, you can rewrite the below code as follows:

```
value = ""
if 10 > 0:
    value = "10 is greater than 0"
else:
    value = "10 is not greater than 0"
```

Which is equivalent to:

```
value = "10 is greater than 0" if 10 > 0 else "10 is not greater than 0"
```

(Back to top)