Language Fundamentals

Character set of python

A character set defines encoding and decoding standards. There are two encoding and decoding standards.

1. ASCII
2. UNICODE

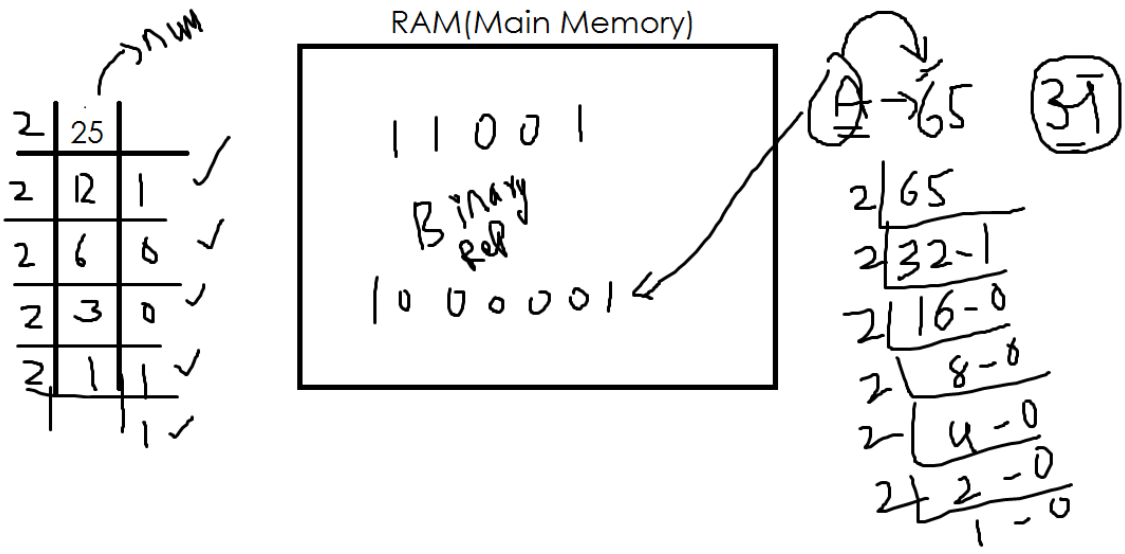
ASCII stands American Standard Code for information Interchange. ASCII supports 256 characters. (0-255).

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]

UNICODE

Unicode is a super set of ASCII.
Unicode support all the characters of ASCII and also support characters from other languages. UNICODE support more than one lakh characters.



Python Tokens

What is a token?
A Token is smallest individual unit within program.

1. Keywords
2. Identifiers
3. Literals
4. Data types
5. Operators

Keywords
Keywords are reserved words (OR) language related words. These words are having special meaning within language. The meaning of these words cannot change by programmer.

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>> len(keyword.kwlist)
```

All keywords are in lowercase except 3 keywords

1. True
2. False
3. None

These keywords represent values in python.

Python case-sensitive language, it understands the difference between uppercase and lowercase.

Identifiers

Identifier is a user defined word.
Identifier is used to identify programming elements

1. Variable names
2. Function names
3. Program name
4. Data type name
5. Package name

Identifier is a user defined word and this word is created using all the alphabets (A-Z/a-z), digits(0-9) and one special character _

Identifier is created using the following rules and regulations

1. Identifier should not be keyword

```
>>> and=100
SyntaxError: invalid syntax
>>> if=1
SyntaxError: invalid syntax
>>> assert=1
SyntaxError: invalid syntax
>>> pass=100
SyntaxError: invalid syntax
```

2. Identifier should not start with digit

```
>>> no1=10
>>> no2=20
>>> no1
10
>>> no2
20
>>> 3no=30
SyntaxError: invalid decimal literal
```

3. Identifier can be defined in uppercase or lowercase

```
>>> a=10
>>> a
10
>>> A=20
>>> A
20
>>> a
10
```

4. Identifier allows only one special character _

```
>>> rollno_number=1
>>> rollno_number
1
>>> student_fees=4000
>>> student_fees
4000
>>> _a=100
>>> _a
100
>>> a_=200
a_
200
>>> _=300
```

```
_
300
$amt=20
SyntaxError: invalid syntax
>>> amt$=300
SyntaxError: invalid syntax
>>> @a=100
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of
'='?
>>> a@=100
Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    a@=100
TypeError: unsupported operand type(s) for @=: 'int' and 'int'
```

5. There should not be any space between identifier

```
>>> student rollno=1
SyntaxError: invalid syntax
```

6. The length of identifier is unlimited

```
>>> aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa=1
>>> bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb=2
>>> aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
1
>>> bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
2
```

pythonbygupta@gmail.com

codewithsatishgupta