python basics

In [ ]:
```
# numbers
```

In [1]:
```
2 + 2
```

Out[1]: 4

In [2]:
```
5-2
```

Out[2]: 3

In [3]:
```
8/2 #division always returns a float
```

Out[3]: 4.0

In [4]:
```
5*9
```

Out[4]: 45

In [5]:
```
17/3 # classic division returns a float
```

Out[5]: 5.666666666666667

In [6]:
```
17//3 # floor division returns a float
```

Out[6]: 5

In [7]:
```
17 % 3  # modulus returns the remainder of the division
```

Out[7]: 2

In [8]:
```
5**2 #  means sqaure of 5
```

Out[8]: 25

In [9]:
```
2**7 # 2 to the power of 7
```

Out[9]: 128

In [10]:
```
3**3 # same example
```

Out[10]: 27

In [11]:
```
width = 20
height = 5 * 9
width * height
```

Out[11]: 900

In [12]:
```
45 * 20
```

Out[12]: 900

In [14]:
```
tax = 12.5 / 100
price = 100.50
price * tax
```

Out[14]: 12.5625

In [15]:
```
price + _  # _ saves last printed expresion
```

Out[15]: 113.0625

In [16]:
```
round (_ , 2)
```

Out[16]: 113.06

str can be represented in ' ', " ", ''' '''.

In [17]:
```
' hello world'
```

Out[17]: ' hello world'

In [18]:
```
" welcome to Naresh it "
```

Out[18]: ' welcome to Naresh it '

In [19]:
```
''' paris rabbit got your back ;)! yay !!!!!!'''
```

Out[19]: ' paris rabbit got your back ;)! yay !!!!!!'

In [20]:
```
"19 04 " # digits and numerals enclosed in quotes are also strings
```

Out[20]: '19 04 '

To use quotation marks in the string follow the following syntax

In [21]:
```
'aren\'t' # with single quotes  for single quotes
```

Out[21]: "aren't"

In [22]:
```
"aren't" #  through double quotes its very easy
```

Out[22]: "aren't"

In [25]:
```
' "yes", she said ' # for double quotes using single quotes
```

Out[25]: ' "yes", she said '

In [26]:
```
"\"yes,\" they said" # through double quotes
```

Out[26]: '"yes," they said'

In [27]:
```
'"Isn\'t," He said' # by single quotes
```

Out[27]: '"Isn\'t," He said'

In [33]:
```
print(" wow !!, how nice is \"this\"")  # syntax to add "" in the print statemen
```

```
wow !!, how nice is "this"
```

\n means new line

In [36]:
```
s = 'First line. \n Second line.'
s           #\n means new line
            #without print() function , special characters are included in the s
```

Out[36]: 'First line. \n Second line.'

In [40]:
```
# with print(), special characters are interpreted, so \n produces new line
s = 'First line.\nSecond line.'
print(s)
```

```
First line.
Second line.
```

If we dont want \ to be interpreted as special character -- use an r before the first quote:

In [46]:
```
print("c:\some\name")
```

```
c:\some
ame
```

In [44]:
```
print(r"c:\some\name") # note the r before the quote
```

```
c:\some\name
```

Use """----""" or ''' -----or''' triple quotes for multi line print statement

In [48]:
```
print("""\
      usage: thingy [OPTIONS]
      -H                         Display the usage message
      -H hostname                hostname connect to
      """)
```

```
usage: thingy [OPTIONS]
-H                         Display the usage message
-H hostname                hostname connect to
```

Strings can be concatenated with the + operator and repeated with * :

In [50]:
```
3 * "san"+ "ia"
```

Out[50]: 'sansansania'

In [51]:
```
"py"+ "thon"
```

Out[51]: 'python'

In [ ]:
```
# when we want to break long strings
```

In [57]:
```
text = ("put several strings within parentheses"
        " to have them joined together.")
text
```

Out[57]: 'put several strings within parentheses to have them joined together.'

In [59]:
```
prefix ="py"
prefix + "thon"
```

Out[59]: 'python'

In [60]:
```
word="sania"
word
```

Out[60]: 'sania'

In [65]:
```
word[0]
```

Out[65]: 's'

In [64]:
```
word[1]
```

Out[64]: 'a'

In [66]:
```
word[-1] # last character
```

Out[66]: 'a'

In [67]:
```
word[-2] # last second character
```

Out[67]: 'i'

In [69]:
```
word[-5]
```

Out[69]: 's'

In [70]:
```
word[0:2]
```

Out[70]: 'sa'

In [71]:
```
word[:5]
```

Out[71]: 'sania'

In [72]:
```
word[2:5]
```

Out[72]: 'nia'

In [73]:
```
word[0:]
```

Out[73]: 'sania'

In [74]:
```
word[3:]
```

Out[74]: 'ia'

s[:i] + s[i:]= s

In [78]:
```
word[:2] + word[2:]
```

Out[78]: 'sania'

In [79]:
```
word[:1] + word[1:]
```

Out[79]: 'sania'

In [80]:
```
word[0] = "T" # python strings are immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[80], line 1
----> 1 word[0] = "T"

TypeError: 'str' object does not support item assignment
```

In [81]:
```
len(word) # returns length of a string
```

Out[81]: 5

In [83]:
```
# simple assingment
rgb = ["red", "green", "blue"]
rgba = rgb
id(rgb) == id(rgb) # they reference the same object
```

Out[83]: True

In [84]:
```
rgba.append("alph")
rgb
```

Out[84]: ['red', 'green', 'blue', 'alph']

In [87]:
```
correct_rgba = rgba[:]
correct_rgba[-1] = "alpha"   # shallow copy of the list
correct_rgba
```

Out[87]:
```
['red', 'green', 'blue', 'alpha']
```

In [88]:
```
rgba
```

Out[88]:
```
['red', 'green', 'blue', 'alph']
```

In [89]:
```
letters =['a','b','c','d','e','f','g']
letters
```

Out[89]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

In [90]:
```
letters[2:5]=['C','D,','E']
letters                          # it replaces some values
```

Out[90]:
```
['a', 'b', 'C', 'D,', 'E', 'f', 'g']
```

In [91]:
```
# now we are removing them
letters[2:5]=[]
letters
```

Out[91]:
```
['a', 'b', 'f', 'g']
```

In [92]:
```
#clear the list by replacing all the elements with an empty list
letters[:]=[]
letters
```

Out[92]:
```
[]
```

In [93]:
```
# print statement
i = 5*5
print("The value of i is",i)
```

```
The value of i is 25
```

In [ ]:
```

```