1- Using Python as a Calculator

In [6]:
```
8 / 5  # division always returns a floating point number
```

Out[6]:
```
1.6
```

In [7]:
```
(50 - 5*6) / 4
```

Out[7]:
```
5.0
```

```
Division in python
1- Classic division returns a float, eg: 17 / 3 = 5.666666666666667
2- Floor division discards the fractional part, eg: 17 // 3 = 5
3- The % operator returns the remainder of the division, eg: 17 % 3 = 2
4- Floored quotient * divisor + remainder, 5 * 3 + 2 = 17
```

In [10]:
```
2 ** 7  # 2 to the power of 7
```

Out[10]:
```
128
```

In [12]:
```
width = 20
height = 5 * 9
width * height
```

Out[12]:
```
900
```

In [2]:
```
tax = 12.5 / 100
price = 100.50
price * tax
```

Out[2]:
```
12.5625
```

In [23]:
```
price + _     #last printed expression is assigned to the variable _
```

Out[23]:
```
113.0625
```

In [24]:
```
round(_, 2)
```

Out[24]:
```
113.06
```

2- Strings

In [26]:
```
#Strings are enclosed into single or double quota
'spam eggs'  # single quotes
```

Out[26]:
```
'spam eggs'
```

In [34]:
```
string1 = 'doesn\'t'  # use \' to escape the single quote...
string2 = "doesn't"   # ...or use double quotes instead
string3 = '"Yes," they said.'
string4 = "\"Yes,\" they said."
string5 = '"Isn\'t," they said.'

print(string1)
print(string2)
print(string3)
print(string4)
print(string5)
```

```
doesn't
doesn't
"Yes," they said.
"Yes," they said.
"Isn't," they said.
```

In [36]:
```
s = 'First line.\nSecond line.'  # \n means newline
s  # without print(), \n is included in the output
```

Out[36]:
```
'First line.\nSecond line.'
```

In [39]:
```
print(s)  # with print(), \n produces a new line
```

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

In [42]:
```
print('C:\some\name')  # here \n means newline!
print(r'C:\some\name')  # r before the quote means that you don't \ to be interpreted as special characters
```

```
C:\some
ame
C:\some\name
```

In [60]:
```
print(3 * 'un' + 'ium') # Strings can be concatenated with the + operator, and repeated with *

print('''Py
            thon''')
print("""py
            thon""")

print('py'      'thon') # Two or more string next to each other are automatically concatenated
text = ('Put several strings within parentheses '
        'to have them joined together.') #'Put several strings within parentheses to have them joined together.'
print(text)

#This only works with two literals though, not with variables or expressions:
#prefix = 'Py'
#prefix 'thon'  # can't concatenate a variable and a string literal
#('un' * 3) 'ium'
```

```
Py
            thon
py
            thon
python
Put several strings within parentheses to have them joined together.
unununium
```

In [ ]:
```
prefix + 'thon'    #If you want to concatenate variables or a variable and a literal, use +
```

Strings can be indexed

```
 +---+---+---+---+---+---+
 | P | y | t | h | o | n |
 +---+---+---+---+---+---+
 0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

In [67]:
```
word = 'Python'
print(word[0])  # character in position 0
print(word[-6])

print(word[5])  # character in position 5
print(word[-1])  # last character

print(word[-2])  # second-last character

print(word[0:2])  # characters from position 0 (included) to 2 (excluded)
print(word[:2])   # character from the beginning to position 2 (excluded)

print(word[2:5])
print(word[4:])   # characters from position 4 (included) to the end

print(word[-2:])

print(word[:2] + word[2:])

print(word[:4] + word[4:])

print(len(word))

#print(word[42]) #error: index out of range

print(word[4:42])
print(word[42:])
```

```
P
P
n
n
o
Py
Py
tho
on
on
Python
Python
6
on
```

Strings are immutable -can't be changed-

In [68]:
```
word[0] = 'J'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-68-91a956888ca7> in <module>
----> 1 word[0] = 'J'

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

In [71]:
```
#If you need a different string, you should create a new one
print('J' + word[1:])
print(word[:2] + 'py')
```

```
Jython
Pypy
```

3- Lists

In [3]:
```
#Lists might contain items of different types, but usually the items all have the same type.
squares = [1, 4, 9, 16, 25]
print(squares)
print(squares[:])
print(squares[0])     # indexing returns the item
print(squares[-1])
print(squares[-3:])   # slicing returns a new list
```

```
[1, 4, 9, 16, 25]
[1, 4, 9, 16, 25]
1
25
[9, 16, 25]
```

In [5]:
```
#Lists are mutable and can be concatenated
print(squares + [36, 49, 64, 81, 100])
print(squares)
squares[-1] = 0
print(squares)

#add to the end of list, using append
squares.append(1)
print(squares)
squares.append(9 ** 2)
print(squares)
```

```
[1, 4, 9, 16, 0, 1, 81, 36, 49, 64, 81, 100]
[1, 4, 9, 16, 0, 1, 81]
[1, 4, 9, 16, 0, 1, 0]
[1, 4, 9, 16, 0, 1, 0, 1]
[1, 4, 9, 16, 0, 1, 0, 1, 81]
```

In [7]:
```
letters = ['a', 'b', 'c', 'd']
print(len(letters))

letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print(letters)
['a', 'b', 'c', 'd', 'e', 'f', 'g']

# replace some values
letters[2:5] = ['C', 'D', 'E']
print(letters)

# now remove them
letters[2:5] = []
print(letters)

# clear the list by replacing all the elements with an empty list
letters[:] = []
print(letters)
```

```
4
['a', 'b', 'c', 'd', 'e', 'f', 'g']
['a', 'b', 'C', 'D', 'E', 'f', 'g']
['a', 'b', 'f', 'g']
[]
```

In [9]:
```
a = ['a', 'b', 'c']
n = [1, 2, 3]
x = [a, n]

print(x)
print(x[0])
print(x[0][1])
```

```
[['a', 'b', 'c'], [1, 2, 3]]
['a', 'b', 'c']
b
```

Task1: Fibonacci series: (0,1,1,2,3,5,8) the sum of two elements defines the next

In [15]:
```
a, b = 0, 1
while a < 10:
    print(a, end=' ')
    a, b = b, a+b
```

```
0 1 1 2 3 5 8
```

In [4]:
```
fib = [0, 1]
for i in range(5):
    fib.append(fib[-1]+fib[-2])
print(fib)
```

```
[0, 1, 1, 2, 3, 5, 8]
```

4- User Input

In [1]:
```
x = input()
print(x+5)
```

```
5

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-6fb94e335c20> in <module>
      1 x = input()
----> 2 print(x+5)

TypeError: can only concatenate str (not "int") to str
```

In [2]:
```
x = int(input())
print(x+5)
```

```
5
10
```

5- Control Flow (if Statements)

In [7]:
```
x = int(input("Please enter an integer: "))
if x < 0:
    print('Negative')
elif x == 0:
    print('Zero')
else:
    print('positive')
```

```
Please enter an integer: 5
positive
```

Task2: check either number is even or odd

In [34]:
```
x = int(input("Please enter an integer: "))
if x % 2 == 0:
    print('even')
else:
    print('odd')

for num in range(2, 10):
    if num % 2 == 0:
        print("Found an even number", num)
        continue
    print("Found an odd number", num)
```

```
Please enter an integer: 6
even
Found an even number 2
Found an odd number 3
Found an even number 4
Found an odd number 5
Found an even number 6
Found an odd number 7
Found an even number 8
Found an odd number 9
```

For Statements

In [15]:
```
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))
```

```
cat 3
window 6
defenestrate 12
```

In [29]:
```
for i in range(5):
    print(i, end=' ')
print()

print(list(range(5, 10)))
print(list(range(0, 10, 3)))
print(list(range(-10, -100, -30)))

a = ['Mary', 'had', 'a', 'little', 'lamb']
for i in range(len(a)):
    print(i, a[i])
```

```
0 1 2 3 4
[5, 6, 7, 8, 9]
[0, 3, 6, 9]
[-10, -40, -70]
0 Mary
1 had
2 a
3 little
4 lamb
```

In [31]:
```
print(range(10))
print(sum(range(4)))
```

```
range(0, 10)
6
```

Task 3:

In [32]:
```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'equals', x, '*', n//x)
            break
        else:
            # loop fell through without finding a factor
            print(n, 'is a prime number')
```

```
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

In [42]:
```
users = {'Hans': 'active', 'Éléonore': 'inactive', '景太郎': 'active'}


#Iterate over a copy
for user, status in users.copy().items(): #iterating over users.items() produces error, cause size changes after delet
    if status == 'inactive':
        del users[user]

print(users)

active_users = {}
for user, status in users.items():
    if status == 'active':
        active_users[user] = status

print(active_users)
```

```
{'Hans': 'active', '景太郎': 'active'}
{'Hans': 'active', '景太郎': 'active'}
{'Hans': 'active', '景太郎': 'active'}
```

6. Defining Functions

In [44]:
```
def fib(n):    # write Fibonacci series up to n
    """Print a Fibonacci series up to n."""
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b

# Now call the function we just defined:
print(fib(2000))
```

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 None
```

In [50]:
```
def fib2(n):  # return Fibonacci series up to n
    """Return a list containing the Fibonacci series up to n."""
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)    # see below
        a, b = b, a+b
    return result

f100 = fib2(100)    # call it
print(f100)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

7- Printing

In [14]:
```
import math
print(f'The value of pi is approximately {math.pi:.3f}.') #f-string, for string formatting literals
print('The value of pi is approximately %5.3f.' % math.pi) #old string formatting
```

```
The value of pi is approximately 3.142.
The value of pi is approximately 3.142.
```

In [9]:
```
table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 7678}
for name, phone in table.items():
    print(f'{name:10} ==> {phone:10d}')
```

```
Sjoerd     ==>       4127
Jack       ==>       4098
Dcab       ==>       7678
```

In [11]:
```
print('{0} and {1}'.format('spam', 'eggs'))
print('{1} and {0}'.format('spam', 'eggs'))
```

```
spam and eggs
eggs and spam
```

Task4:

In [7]:
```
for x in range(1, 11):
     print('{0:2d} {1:2d} {2:4d}'.format(x, x*x, x*x*x))
```

```
 1  1    1
 2  4    8
 3  9   27
 4 16   64
 5 25  125
 6 36  216
 7 49  343
 8 64  512
 9 81  729
10 100 1000
```

Data structure

list

In [41]:
```
fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
print(fruits.count('apple'))
print(fruits.count('tangerine'))
print(fruits.index('banana'))
print(fruits.index('banana', 4))  # Find next banana starting a position 4
fruits.reverse()
print(fruits)
fruits.append('grape')
print(fruits)
fruits.insert(2, 'strawberry')
print(fruits)
fruits.sort()
print(fruits)
fruits.remove('strawberry')
print(fruits)
fruits.pop()
print(fruits)
fruits.pop(4)
print(fruits)
fruitsCopy = fruits.copy()
print(fruitsCopy)
fruitsCopy.clear()
print(fruitsCopy)
del fruitsCopy[:]
print(fruitsCopy)
```

```
2
0
3
6
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
['banana', 'apple', 'strawberry', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear', 'strawberry']
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange']
['apple', 'apple', 'banana', 'banana', 'kiwi', 'orange']
['apple', 'apple', 'banana', 'banana', 'kiwi', 'orange']
[]
[]
```

In [24]:
```
squares = []
for x in range(10):
    squares.append(x**2)
print(squares)

squares2 = [x**2 for x in range(10)]
print(squares2)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [30]:
```
print( [x for x in [1,2,3]] )
print( [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y] )
```

```
[1, 2, 3]
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

In [34]:
```
combs = []
for x in [1,2,3]:
    for y in [3,1,4]:
        if x != y:
            combs.append((x, y))
print(combs)
```

```
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

In [36]:
```
[(x, x**2) for x in range(6)]
```

Out[36]:
```
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
```

In [39]:
```
from math import pi
[str(round(pi, i)) for i in range(1, 6)]
```

Out[39]:
```
['3.1', '3.14', '3.142', '3.1416', '3.14159']
```

Sets

In [49]:
```
basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print(basket)                      # show that duplicates have been removed
print('orange' in basket)          # fast membership testing
print('crabgrass' in basket)

a = set('abracadabra')
print(a)
b = set('alacazam')                    # unique letters in a
print(b)

# Demonstrate set operations on unique letters from two words
print(a - b)                           # letters in a but not in b
print(a | b)                           # letters in a or b or both
print(a & b)                           # letters in both a and b
print(a ^ b)                           # letters in a or b, but not both

a = {x for x in 'abracadabra' if x not in 'abc'}
print(a)
```

```
{'pear', 'banana', 'apple', 'orange'}
True
False
{'a', 'b', 'd', 'c', 'r'}
{'a', 'z', 'c', 'l', 'm'}
{'d', 'b', 'r'}
{'a', 'b', 'd', 'z', 'c', 'l', 'r', 'm'}
{'a', 'c'}
{'b', 'l', 'r', 'd', 'z', 'm'}
{'d', 'r'}
```

Dictionary

In [9]:
```
tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
print(tel)
print(tel['jack'])
del tel['sape']
print(tel)
print(list(tel))
print(sorted(tel))
print('guido' in tel)
print('jack' not in tel)

x = dict(sape=4139, guido=4127, jack=4098)
print(x)

{x: x**2 for x in (2, 4, 6)}
```

```
{'jack': 4098, 'sape': 4139, 'guido': 4127}
4098
{'jack': 4098, 'guido': 4127}
['jack', 'guido']
['guido', 'jack']
True
False
{'sape': 4139, 'guido': 4127, 'jack': 4098}
```

Out[9]:
```
{2: 4, 4: 16, 6: 36}
```

looping techniques

In [22]:
```
knights = {'gallahad': 'the pure', 'robin': 'the brave'}
for k, v in knights.items():
     print(k, v)
```

```
gallahad the pure
robin the brave
```

In [23]:
```
for i, v in enumerate(['tic', 'tac', 'toe']):
     print(i, v)
```

```
0 tic
1 tac
2 toe
```

In [15]:
```
questions = ['name', 'quest', 'favorite color']
answers = ['lancelot', 'the holy grail', 'blue']
for q, a in zip(questions, answers):
    print('What is your {0}?  It is {1}.'.format(q, a))
```

```
What is your name?  It is lancelot.
What is your quest?  It is the holy grail.
What is your favorite color?  It is blue.
```

In [17]:
```
for i in reversed(range(1, 10, 2)):
    print(i)
```

```
9
7
5
3
1
```

In [19]:
```
basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
for i in sorted(basket):
    print(i)
```

```
apple
apple
banana
orange
orange
pear
```

In [27]:
```
basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
print(set(basket))
for f in sorted(set(basket)):
    print(f)
```

```
{'banana', 'pear', 'orange', 'apple'}
apple
banana
orange
pear
```

End