LabDrill02_yourFirstnameLastname

Instructions:

3. An Informal Intro to Python

In [1]:
```
#this is the first comment
spam = 1  # nad this is the second comment
          #... and this is the third
text = "#this is not a comment because it's in quotes"
```

In [2]:
```
spam
```

Out[2]:
```
1
```

In [3]:
```
text
```

Out[3]:
```
"#this is not a comment because it's in quotes"
```

3.1 Using Python as a Calculator

In [4]:
```
2+2
```

Out[4]:
```
4
```

In [5]:
```
50-5*6
```

Out[5]:
```
20
```

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

```
Out[6]: 5.0
```

```
In [7]:  8/5 # division always returns a floating point number
```

```
Out[7]: 1.6
```

3.1.1 Numbers

```
In [8]:  17/3     #classic division returns a float
```

```
Out[8]: 5.666666666666667
```

```
In [9]:  17//3    #floor division discards the fractional part
```

```
Out[9]: 5
```

```
In [10]:  17%3    #the modulus operator returns the remainder of the division
```

```
Out[10]: 2
```

```
In [11]:  5*3+2      # result * divisor + remainder
```

```
Out[11]: 17
```

```
In [12]:  5 ** 2    # squared
```

```
Out[12]: 25
```

```
In [13]:  2**7  # 2 to the power of 7
```

```
Out[13]: 128
```

```
In [14]:  width = 20
          height = 5*9
          width * height
```

```
Out[14]: 900
```

```
In [15]:  n
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-ab0680a89434> in <module>
----> 1 n

NameError: name 'n' is not defined
```

3.1.2 Strings

```
In [17]:  'spam eggs'   #single quotes strings

Out[17]:  'spam eggs'

In [18]:  'doesn\'t'    # use \' to escape the single quote

Out[18]:  "doesn't"

In [19]:  "doesn't"

Out[19]:  "doesn't"

In [20]:  '"Yes", they said'

Out[20]:  '"Yes", they said'

In [21]:  "\"Yes, \" they said."

Out[21]:  '"Yes, " they said.'

In [22]:  '"Isn\'t, they said'

Out[22]:  '"Isn\'t, they said'

In [23]:  '"Isn\'t," they said.'

Out[23]:  '"Isn\'t," they said.'
```

In [24]:
```
print('"Isn\'t," they said.')
```

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

In [25]:
```
s = "Frist line.\nSecond line."
s
print(s)
```

```
Frist line.
Second line.
```

In [26]:
```
print('C:\some\name')
```

```
C:\some
ame
```

In [27]:
```
print(r'C:\some\name')  #note the r before the quote
```

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

In [28]:
```
print("""\
Usage: thingy [OPTIONS]
    -h                      Display this usage message
    -H hostname           Hostname to connect to
""")
```

```
Usage: thingy [OPTIONS]
    -h                      Display this usage message
    -H hostname           Hostname to connect to
```

In [29]:
```
# 3 times 'un', followed by 'ium'
3*'un'+'ium'
```

Out[29]:
```
'unununium'
```

In [30]:
```
'Py' 'thon'
```

Out[30]:
```
'Python'
```

```
In [31]:  text = ('Put several strings withing parentheses '
                  'to hav ethem joined together.')
          text

Out[31]: 'Put several strings withing parentheses to hav ethem joined together.'

In [32]:  prefix = 'Py'

In [33]:  prefix 'thon'

  File "<ipython-input-33-7edf08bff78f>", line 1
    prefix 'thon'
                ^
SyntaxError: invalid syntax

In [34]:  prefix+'thon'

Out[34]: 'Python'

In [35]:  word = 'Python'

In [36]:  word[0]     #character in position 0

Out[36]: 'P'

In [37]:  word[5]        #char in 5th position

Out[37]: 'n'

In [38]:  word[-1]

Out[38]: 'n'

In [39]:  word[-2]

Out[39]: 'o'
```

```
In [40]: word[-6]

Out[40]: 'P'

In [41]: word[0:2]

Out[41]: 'Py'

In [42]: word[2:5]

Out[42]: 'tho'

In [43]: word[:2]+word[2:]

Out[43]: 'Python'

In [44]: word[:4]+word[4:]

Out[44]: 'Python'

In [45]: word[:2]

Out[45]: 'Py'

In [46]: word[4:]

Out[46]: 'on'

In [47]: word[-2:]

Out[47]: 'on'

In [48]: word[42]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-48-4d0f20275732> in <module>
----> 1 word[42]

IndexError: string index out of range
```

```
In [49]:  word[4:42]

Out[49]:  'on'

In [50]:  word[42:]

Out[50]:  ''

In [51]:  word[0]='J'

          ---------------------------------------------------------------------------
          TypeError                                 Traceback (most recent call last)
          <ipython-input-51-36bea27fec3d> in <module>
          ----> 1 word[0]='J'

          TypeError: 'str' object does not support item assignment

In [52]:  word[2:] = 'py'

          ---------------------------------------------------------------------------
          TypeError                                 Traceback (most recent call last)
          <ipython-input-52-6488bbf78f5a> in <module>
          ----> 1 word[2:] = 'py'

          TypeError: 'str' object does not support item assignment

In [ ]:   'J'+ word[1:]

In [ ]:   word[:2]+'py'

In [53]:  s='supercalifragislisticexpiladocious'
          len(s)

Out[53]:  34
```

Lists

```
In [54]:  squares = [1,4,9,16,25]
          squares

Out[54]:  [1, 4, 9, 16, 25]

In [55]:  squares[0]

Out[55]:  1

In [56]:  squares[-1]

Out[56]:  25

In [57]:  squares[-3]

Out[57]:  9

In [58]:  squares[:]

Out[58]:  [1, 4, 9, 16, 25]

In [59]:  squares + [36,49,64,81,100]

Out[59]:  [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

In [60]:  cubes = [1,8,27,65,125]

In [61]:  4**3

Out[61]:  64

In [63]:  cubes[3]=64

In [64]:  cubes

Out[64]:  [1, 8, 27, 64, 125]
```

In [65]:
```
cubes.append(216)
```

In [66]:
```
cubes.append(7**3)
```

In [67]:
```
cubes
```

Out[67]:
```
[1, 8, 27, 64, 125, 216, 343]
```

In [68]:
```
letters = ['a','b','c','d','e','f','g']
letters
```

Out[68]:
```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

In [69]:
```
letters[2:5] = ['C','D','E']
letters
```

Out[69]:
```
['a', 'b', 'C', 'D', 'E', 'f', 'g']
```

In [70]:
```
letters[2:5]=[]
letters
```

Out[70]:
```
['a', 'b', 'f', 'g']
```

In [71]:
```
letters[:]=[]
letters
```

Out[71]:
```
[]
```

In [72]:
```
letters=['a','b','c','d']
len(letters)
```

Out[72]:
```
4
```

In [73]:
```
a = ['a','b','c']
n = [1,2,3]
x = [a,n]
x
```

Out[73]:
```
[['a', 'b', 'c'], [1, 2, 3]]
```

```
In [74]: x[0]
```

```
Out[74]: ['a', 'b', 'c']
```

```
In [75]: x[0][1]
```

```
Out[75]: 'b'
```

3.2 First Steps towards programming

```
In [76]: #fibonacci series
         a,b = 0,1
         while a< 10:
             print(a)
             a,b=b,a+b
```

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

```
In [77]: i=256**2
         print('The value of i is', i)
```

```
The value of i is 65536
```

```
In [78]: a,b = 0,1
         while a< 1000:
             print(a,end=',')
             a,b = b, a+b
```

```
0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,
```

4. More Control Flow Tools

4.1 if statemnets

In [79]:
```python
x = int(input('Please enter an integer: '))

if x<0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x ==1:
    print("Single")
else:
    print("More")
```

```
Please enter an integer: 4
More
```

4.2 for statements

In [80]:
```python
words = ['cat','windows','defenstrate']
```

In [81]:
```python
for w in words:
    print(w, len(w))
```

```
cat 3
windows 7
defenstrate 11
```

In [82]:
```python
for w in words[:]:
    if len(w) >6:
        words.insert(0,w)
words
```

Out[82]:
```
['defenstrate', 'windows', 'cat', 'windows', 'defenstrate']
```

4.3 the range() function

In [83]:

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

In [84]:

```
for i in range(5,10):
    print(i)
```

```
5
6
7
8
9
```

In [85]:

```
for i in range(0,10,3):
    print(i)
```

```
0
3
6
9
```

In [86]:

```
for i in range(-10,-100, -30):
    print(i)
```

```
-10
-40
-70
```

In [88]:

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

```
0 Mary
1 had
2 a
3 little
4 lamb
```

In [90]:

```
print(range(10))
```

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

In [91]:
```python
list(range(10))
```

Out[91]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

4.4 break and continue Statements and else clauses on loops

In [92]:
```python
for n in range(2,10):
    for x in range(2,n):
        if n % x == 0:
            print(n,'equals',x,'*',n//x)
            break
    else:
        print(n, 'is prime number')
```

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

In [94]:
```python
for num in range(2,10):
    if num % 2 == 0:
        print("Found and even number", num)
        continue
    print("Found a number", num)
```

```
Found and even number 2
Found a number 3
Found and even number 4
Found a number 5
Found and even number 6
Found a number 7
Found and even number 8
Found a number 9
```

In [*]:
```python
while True:
    pass
```

In [*]:
```
class MyEmptyClass:
    pass
```

In [*]:
```
def initlog(*args):
    pass   # Remember to implement this!
```

In [*]:
```
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
    print()

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

4.6 Defining functions

In [*]:
```
fib
```

In [*]:
```
f = fib
```

In [*]:
```
f(2000)
```

In [*]:
```
fib(0)
```

In [*]:
```
print(fib(0))
```

In [*]:

```
def fib2(n):
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result

f100 = fib2(100)
f100
```

4.7 More on difining functions

4.7.1 default arguments values

In [*]:

```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('invalid user response')
        print(reminder)
```

In [*]:

```
ask_ok("do you really want to quit?")
```

In [*]:

```
ask_ok('OK to overwrite the file?', 2)
```

In [*]:

```
ask_ok('OK to overwrite the file?', 2, 'Come on, only yes or no!')
```

In [*]:

```
i = 5

def f(arg=i):
    print(arg)

i = 6
f()
```

In [*]:
```
def f(a, L=[]):
    L.append(a)
    return L

print(f(1))
print(f(2))
print(f(3))
```

4.7.2 keyword arguments

In [*]:
```
def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
    print("-- This parrot wouldn't", action, end=' ')
    print("if you put", voltage, "volts through it.")
    print("-- Lovely plumage, the", type)
    print("-- It's", state, "!")
```

In [*]:
```
parrot(1000)
parrot(voltage=1000)
parrot(voltage=1000000, action='VOOOM')
parrot(action='VOOOOOM', voltage=1000000)
parrot('a million', 'bereft of life', 'jump')
parrot('a thousand', state='pushing up the daisies')
```

In [*]:
```
parrot()
parrot(voltage=5.0, 'dead')
parrot(110, voltage=220)
parrot(actor='John Cleese')
```

In [*]:
```
parrot()
```

In [*]:
```
parrot(voltage=1000)
```

In [*]:
```
parrot(110, voltage=220)
```

In [*]:
```
parrot(actor="john cleese")
```

In [*]:

```python
def function(a):
    pass

function(0, a=0)
```

In [ ]:

```python

```