AN INFORMAL INTRODUCTION TO PYTHON

- USING A PYTHON CALCULATOR

```
In [2]: >>> 2+3
```

```
Out[2]: 5
```

```
In [3]: >>> 50-5*6/4
```

```
Out[3]: 42.5
```

```
In [4]: >>> 8/5 #division always returns as a floating point number
```

```
Out[4]: 1.6
```

division always returns a float. to do floor divisiom and get an integer result you can use // operation, to calculate the remainder you can use%

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

```
Out[5]: 5.666666666666667
```

```
In [6]: >>> 17//3 #floor division discard the fractional part
```

```
Out[6]: 5
```

```
In [8]: >>> 17%3 #the oerator returns the remainder of the division
```

```
Out[8]: 2
```

```
In [ ]: >>> 5*3+2 #floated quotent
```

calculation of power

```
In [10]: >>> 5**2 #5 squared
```

```
Out[10]: 25
```

```
In [12]: >>> 2**7 # 2 to the power 7
```

```
Out[12]: 128
```

Equal sign used to assign a value to a variable

```
In [13]: >>> width=20
         >>> height= 5*9
         >>> width*height
```

```
Out[13]: 900
```

In interactive mode the last printed expression is assigned to the variable_. This means that when you are using python as adesk calculator it is somewhat easier to continue calculation

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

```
Out[14]: 12.5625
```

```
In [15]: >>> price+_
```

```
Out[15]: 113.0625
```

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

```
Out[16]: 113.06
```

STRING

```
In [17]: >>> 'spam eggs' #single quote
```

```
Out[17]: 'spam eggs'
```

```
In [18]: >>> "paris rabbit got your back:)!yay!" #double quote
```

```
Out[18]: 'paris rabbit got your back:)!yay!'
```

```
In [19]: >>> '1978' #digit and numbers enclosed in quotes also string
```

```
Out[19]: '1978'
```

STRING INDEXING

```
In [20]: >>> word= 'python'
```

```
In [21]: >>> word[0] #character in position 0
```

```
Out[21]: 'p'
```

```
In [22]: >>> word[5] #character in 5 position
```

```
Out[22]: 'n'
```

REVERSE INDEXING

```
In [23]: >>> word[-1] #last character
```

```
Out[23]: 'n'
```

```
In [24]: >>> word[-5]

Out[24]: 'y'

In [25]: >>> word[-2] #2nd last character

Out[25]: 'o'
```

SLICING

```
In [27]: >>> word[0:2] #position from 0 to 2

Out[27]: 'py'

In [28]: >>> word[0:5]

Out[28]: 'pytho'

In [30]: >>> word[4:] #character from position 4 to end

Out[30]: 'on'

In [31]: >>> word[-2:] #character from the 2nd last to end

Out[31]: 'on'
```

LIST

list might contain items of different type but usually the items all have same type

```
In [36]: >>> square=[1,4,9,16,25]
         >>> square

Out[36]: [1, 4, 9, 16, 25]
```

LIST INDEXING AND SLICING

```
In [37]: >>> square[0]

Out[37]: 1

In [38]: >>> square[-1]

Out[38]: 25

In [39]: >>> square[-3]

Out[39]: 9
```

you can also add new items at the end of the list by using list.append()

```
In [52]: >>> cubes=[1,8,27,64,125]
         >>> cubes

Out[52]: [1, 8, 27, 64, 125]

In [54]: >>> cubes.append(216)
         >>> cubes

Out[54]: [1, 8, 27, 64, 125, 216]

In [55]: >>> cubes.append(7**3)
         >>> cubes

Out[55]: [1, 8, 27, 64, 125, 216, 343]
```

built_in function len() also applies to list

```
In [57]: >>> letter=['a','b','c','d']
         >>> letter

Out[57]: ['a', 'b', 'c', 'd']

In [59]: >>> len(letter)

Out[59]: 4
```

Itis possible to nest list(create list containing other list

```
In [60]: >>> a=['a','b','c']
         >>> n=[1,2,3]
         >>> x=[a,n]
         >>> x

Out[60]: [['a', 'b', 'c'], [1, 2, 3]]
```

assignment to slices also possible,this can also change the size of the list or clear it entirely

```
In [61]:  >>> letters=['a','b','c','d','e','f']
         >>> letters

Out[61]: ['a', 'b', 'c', 'd', 'e', 'f']

In [62]: >>> letters[2:5]=['C','D','E'] #replace some items
         >>> letters

Out[62]: ['a', 'b', 'C', 'D', 'E', 'f']

In [63]: >>> letters[2:5]=[] #remove items

In [64]: >>> letters
```

```
Out[64]: ['a', 'b', 'f']
```

```
In [65]: >>> letters[:]=[] #clear the list
         >>> letters
```

```
Out[65]: []
```

An example that uses most of the list methods:

```
In [66]: >>> fruits=['orange','apple','pear','banana','kiwi','apple','banana']
         >>> fruits.count('apple')
```

```
Out[66]: 2
```

```
In [67]: >>> fruits.count('tangerine')
```

```
Out[67]: 0
```

```
In [69]: >>> fruits.count('banana')
```

```
Out[69]: 2
```

```
In [70]: >>> fruits.index('banana')
```

```
Out[70]: 3
```

```
In [71]: >>> fruits.index('banana',4)
```

```
Out[71]: 6
```

```
In [72]: >>> fruits.reverse()
         >>> fruits
```

```
Out[72]: ['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
```

```
In [73]: >>> fruits.sort()
         >>> fruits
```

```
Out[73]: ['apple', 'apple', 'banana', 'banana', 'kiwi', 'orange', 'pear']
```

TUPLES

Atuple consist of a number of values separated by commas,for instance

```
In [74]: >>> t=12345, 54321, 'hello'
         >>> t[0]
```

```
Out[74]: 12345
```

```
In [75]: >>> t
```

```
Out[75]: (12345, 54321, 'hello')
```

```
In [76]: >>> u=t,(1,2,3,4,5)
         >>> u
```

```
Out[76]: ((12345, 54321, 'hello'), (1, 2, 3, 4, 5))
```

```
In [77]: >>> #tuples are immutable
```

```
In [78]: >>> t[0]= 8888
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[78], line 1
----> 1 t[0]= 8888

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

```
In [80]: >>> v=([1,2,3],[3,2,1])
         >>> v
```

```
Out[80]: ([1, 2, 3], [3, 2, 1])
```

SET

```
In [83]: >>> basket = {'apple','orange','apple','pear','orange','banana'}
         >>> print(basket)
```

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

```
In [84]: >>> 'orange' in basket
```

```
Out[84]: True
```

```
In [86]: >>> 'crabgrass' in basket
```

```
Out[86]: False
```

demonstrate set operations on unique lettes from two words

```
In [87]: >>> a=set('abracadabra')
```

```
In [89]: >>> b=set('alacazam')
         >>> a
```

```
Out[89]: {'a', 'b', 'c', 'd', 'r'}
```

```
In [90]: >>> a-b
```

```
Out[90]: {'b', 'd', 'r'}
```

```
In [91]: >>> a|b
```

```
Out[91]: {'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}
```

```
In [92]: >>> a &b
```

```
Out[92]: {'a', 'c'}
```

```
In [93]: >>> a^b
```

```
Out[93]: {'b', 'd', 'l', 'm', 'r', 'z'}
```

DICTIONARY

Here is a small example using a dictionary

```
In [94]: >>> tel={'jack':4098,'sape':4139}
         >>> tel['guido']= 4127
         >>> tel
```

```
Out[94]: {'jack': 4098, 'sape': 4139, 'guido': 4127}
```

```
In [95]: >>> tel['jack']
```

```
Out[95]: 4098
```

```
In [96]: >>> del tel['sape']
         >>> tel['irv']= 4127
         >>> tel
```

```
Out[96]: {'jack': 4098, 'guido': 4127, 'irv': 4127}
```

```
In [97]: >>> list(tel)
```

```
Out[97]: ['jack', 'guido', 'irv']
```

```
In [98]: >>> sorted(tel)
```

```
Out[98]: ['guido', 'irv', 'jack']
```

```
In [99]: >>> 'guido' in tel
```

```
Out[99]: True
```

```
In [100… >>> 'jack' not in tel
```

```
Out[100… False
```

```
In [ ]:
```