CONDITIONS

comparison operators:

```
<   (less than),
>   (greater than),
==  (equal to),
<=  (less than or equal to),
>=  (greater than or equal to)
!=  (not equal to)
```

If statement:

```
x = int(input("Please enter an integer: "))

if x < 0:
print('Negative changed to zero')

elif x == 0:

     print('Zero')

elif x == 1:

     print('Single')

else:

     print('More')
```

logical operator:

and	Determines whether both operands are true.	True and True is True True and False is False False and True is False False and False is False
or	Determines when one of two operands is true.	True or True is True True or False is True False or True is True False or False is False
not	Negates the truth value of a single operand. A true value becomes false and a false value becomes true.	not True is False not False is True

LOOPS

for:

```
words = ['cat', 'dog', 'mouse']

for w in words:
       print(w, len(w))

for letter in 'geek':
      print(letter)
```

while:

```
a = 0
b = 1
while b < 10:
     print(b)
     temp = b
     b = a + b
     a = temp
```

useful iterate tools:

range:

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

```
range(5, 10)    #5 through 9
range(0, 10, 3) #0, 3, 6, 9
```

enumerate:

```
list = ['a', 'b', 'c']
for counter, element in enumerate(list):
        print(counter, element)
```

continue:

```
iloscLiter = 0
slowo = "Geek Academy"

for i in slowo:
    if i == " ":
        print ("znalazlem spacje")
        continue
    iloscLiter += 1

print (iloscLiter)
```

break:

```
while True:
    var = input ("Enter something, or 'q' to quit:
    print (var)
    if var == 'q':
        break
```

FUNCTIONS

```
def send_message():
       print ("that is my first function")
```

passing and return arguments:

```
def powerNumber(x):
      print("doing complicated calculation…")
      return x**2

print(powerNumber(16))
```

passing *args and **kwargs

```
def my_function(a, b, *args, **kwargs):
      pass
```

*args is used to pass variable number of arguments to a function

**kwargs is used to pass keyworded, variable-length argument list

CLASS EXAMPLE

```
class Animals():
    def breathe(self):
        print("oddycham")
    def move(self):
        print("biegam")
    def eat_food(self):
        pass

class Mammals(Animals):
    def feed_young_with_milk():
        print("karmi")

class Giraffes(Mammals):
    def eat_leaves_from_trees(self):
        print('je liscie')

x = Giraffes()
x.eat_leaves_from_trees()
```

DATA TYPES

BASE TYPES

int, float:

```
people = 12
tax = 12.5 / 100
price = 100.50
```

conversion:

```
>>>int(10)
10
>>>float(10)
10.0
>>>str(10)
'10'
>>bin(2)
'0b10'
```

bytes:

```
binary_number = 0b10
octal_number = 0o10
hexdecimal_number = 0x10
```

string:

```
Text_variable = "my text example"
```

String operations:

concatenate:

```
text = "textA" + "textB"
```

immutable:

word[0] = 'J' #ERROR!

indexing:

```
word = "Python"
word[1] # character 'y'
word[-1]  # last character 'n'
```

slicing:

```
word[0:2]
# characters from position 0 (included) to 2 (excluded)
'Py'

 word[2:5]
# characters from position 2 (included) to 5 (excluded)
'tho'
```

CONTAINER TYPES

list:

```
squares = [1, 2, 4, 9, 16, 25]
names= ['Paweł', 'Grazyna']
```

tuple:

```
zwierzeta = ('pies', 'kot', 'mysz')
pusty = ()          t1 = (10, )
```

Lists operations:

concatenation:

```
>>> squares = [1, 2, 4, 9, 16]
>>> squares + [36, 49, 64, 81]
[1, 2, 4, 9, 16, 36, 49, 64, 81]
```

indexing:

```
squares[0]  #returns the item1
squares[-1] #returns last element
```

replace:

```
letters = ['a', 'b', 'c', 'd', 'e']
letters[2:5] = ['C', 'D', 'E']
letters['a', 'b', 'C', 'D', 'E']
```

remove:

```
letters[2:5] = []
letters['a', 'b', 'f', 'g']
```

Main differences between lists and tuples:

-Lists can be modified (enable handy operations), tuples are immutable.
-Tuples are **faster** than lists. If you're defining a constant set of values and all you're ever going to do with it is iterate through it, use a tuple instead of a list.
-It makes your code safer if you "**write-protect**" data that does not need to be changed.

list comprehensions:

```
S = [x**2 for x in range(8)]
M = [x for x in S if % 2 == 0]

S ->[0, 1, 4, 9, 16, 25, 36, 49]
M ->[0, 4, 16, 36]
```

dict:

```
pythons = {'Cleese': 'John', 'Gilliam': 'Terry'}

#useful oferations:
others = { 'Marx': 'Groucho'}
pythons.update(others)
>>>pythons:{'Cleese': 'John', 'Gilliam': 'Terry',
'Marx': 'Groucho', 'Chapman': 'Graham'}
```

set:

```
a = {1, 2}
b = {2, 3}

>>>a & b -> {2}
>>> a | b -> {1, 2, 3}
>>> a.union(b) -> {1, 2, 3}
 >>> a – b ->{1}
>>> a.difference(b) -> {1}
```