Recap

Yesterday

Variables

Declaring & initializing Variables

Globally Typed Vs Statically Typed

Data Types

Exercises

Today

Python Keywords

Python Comments

Python Input

Python Conditional Statements

Logical Operators

Pass Statement

Version Control

Python Keywords

Python has a set of keywords that are reserved words that cannot be used as variable names, function names, or any other identifiers.

Keywords Examples

and

as

break

class

continue

def

del

for

Etc etc

Python Keywords

Which keywords we used in our yesterday session?

Python Keywords

```
$ python3.12
>>> help()
help> keywords
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

Python Comments

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code.

Python Comments

Comments starts with a #, and Python will ignore those

Python Comments

```
#This is a comment
print("Hello, World!")
```

Python Comments - Multiline

```
#This is a comment
# written in
# more than just one line
print("Hello, World!")

"""
This is a comment
written in
more than just one line
"""
```

Python input function

The `input()` function allows user input.

```
x = input('Enter your name:')

print('Hello, ' + x)
```

Python Conditional Statements

Python supports the usual logical conditions from mathematics:

Comparison Operators

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

Python Conditional Statements

Conditions can be used in several ways, most commonly in "**if statements**" and loops.

If Statement

An "if statement" is written by using the `if` keyword.

Elif Statement

The `elif` keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

Else Statement

The `else` keyword catches anything which isn't caught by the preceding conditions.

If – Elif - Else

```
a = 200
b = 33
if b > a: #False
  print("b is greater than a")
elif a == b: #False
  print("a and b are equal")
else:
  print("a is greater than b")
```

Short hand If

```
if a > b: print("a is greater than b")
```

Short Hand If ... Else

```
a = 2
b = 330
print("A") if a > b else print("B")
```

This technique is known as **Ternary Operators**, or **Conditional Expressions**.

```
a = 330
b = 330
[print("A") if a > b ] else [ print("=") if a == b else print("B")]
```

And Operator

The and keyword is a logical operator, and is used to combine conditional statements:

Condition 1 and condition 2 and condition 3 and ……

1. Multiple conditions
2. All conditions should be true to proceed = AND

   2 conditions
   username should be correct
   password should be correct

   to login both should be true

And Operator

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

Or Operator

The or keyword is a logical operator, and is used to combine conditional statements:

1. Multiple conditions
2. Only one condition needs to be true to proceed = OR

   2 conditions
   username or password should be correct

   to login both should be true

Or Operator

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

Not Operator

The `not` keyword is a logical operator, and is used to reverse the result of the conditional statement:

Not Operator

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

Nested If

You can have `if` statements inside `if` statements, this is called *nested* `if` statements.

Nested If

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 100:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

The pass Statement

`if` statements cannot be empty, but if you for some reason have an `if` statement with no content, put in the `pass` statement to avoid getting an error.

The pass Statement

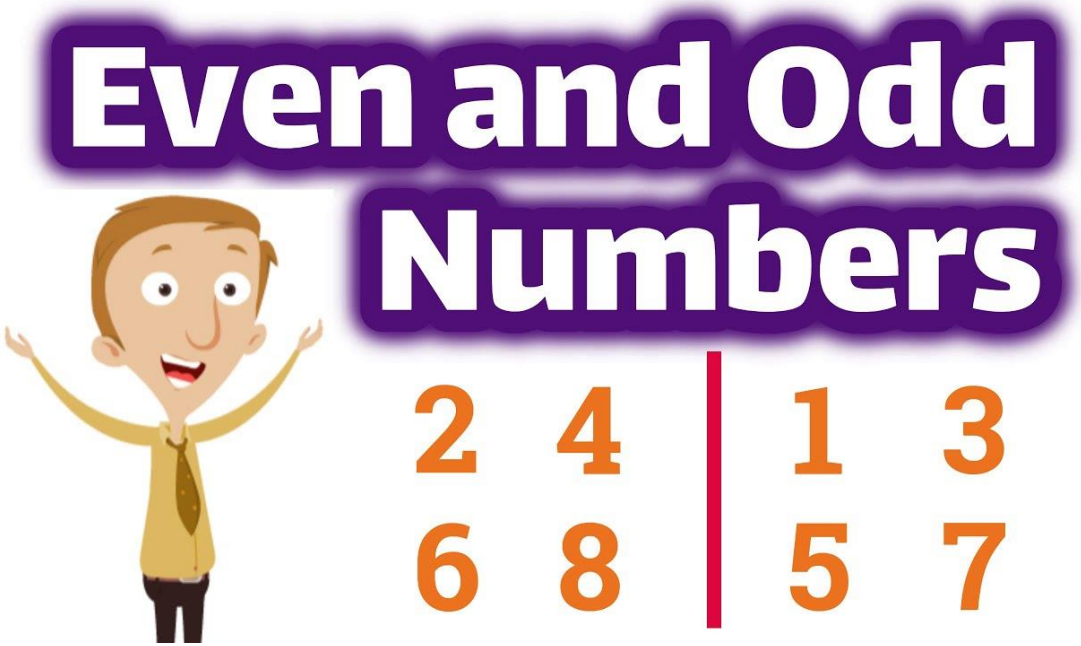
```
a = 33
b = 200

if b > a: #True
  pass
```

Activity 1

Write a program that takes input from user and determine if the given number is odd or even?

% = Remainder
5 % 2 = 1



Activity 2

A company decided to give bonus of 7% to employee if his/her year of service is more than 3 years. Ask user for their salary and year of service and print the net bonus amount.