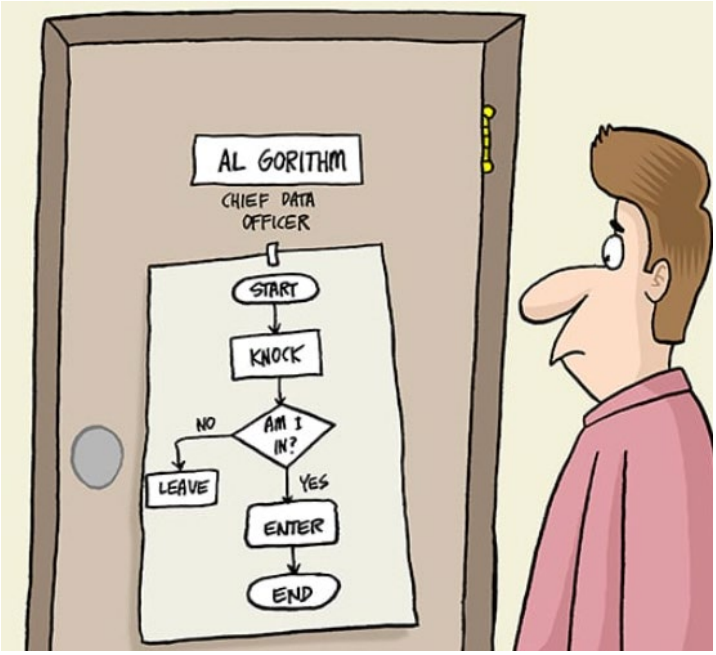
Lösungsalgorithmen & Programmieren

- Block 01 /Abend 04-

Darstellung von Algorithmen



🎯 Leistungs-/Lernziele dieses Blocks:

Die Studierenden …

kennen den Zweck grafischer Darstellungsmethoden für Algorithmen (PAP, Struktogramm nach Nassi-Shneiderman, Pseudocode)
können die wichtigsten Symbole des Programmablaufplans (Start/Ende, Prozess, Entscheidung, Ein-/Ausgabe, Pfeile) korrekt benennen und verwenden
kennen den Aufbau eines Nassi-Shneiderman-Struktogramms und verstehen dessen Elemente (Sequenz, Selektion, Iteration)
können gängige Kontrollstrukturen (Sequenz, Verzweigung, Schleife) in beiden Diagrammformen (PAP und Nassi–Shneiderman) mit Hilfe geeigneter Software darstellen
können aus einem gegebenen PAP-Diagramm und /oder Nassi-Shneiderman-Struktogramm korrekten Python-Code ableiten
können aus bestehendem Python-Code die zugehörige PAP-Darstellung und/oder Nassi-Shneiderman-Strukto-gramm ableiten
können algorithmische Abläufe strukturiert in einer Diagrammform darstellen und anschliessend implementieren

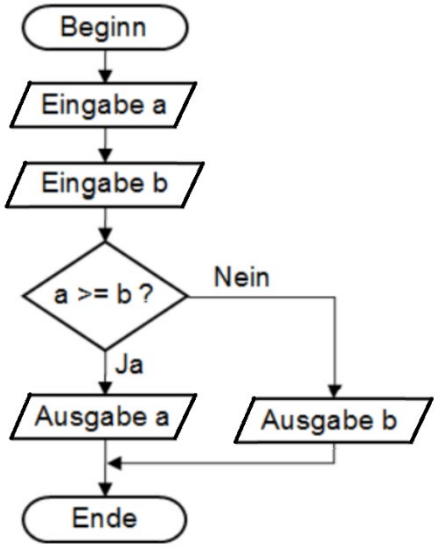
Inhaltsverzeichnis:

1	**Programmablaufplan (PAP)**	**2**
2	**Struktogramme (Nassi-Schneidermann)**	**4**
3	**Pseudocode**	**6**
4	**Miniprojekte realisieren**	**7**
4.1	Projekt 01 (simple code):	7
4.2	Projekt 02 (Geldüberweisung via Twint):	7
4.3	Projekt 03 (Bananaway):	7
4.4	Projekt 04 (PAP umsetzen):	8
4.5	Projekt 05 (Fibonacci Folge):	9

1 Programmablaufplan (PAP)

(Dieses Kapitel ist eine Zusammenfassung ihres Herdt-Buches (Programmieren Grundlagen, Kapitel 3).

Ein Programmablaufplan (PAP) wird auch Ablaufdiagramm, Flussdiagramm oder Blockdiagramm genannt.

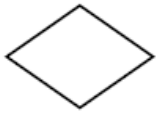
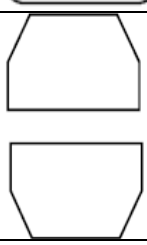


Programmablaufpläne sind grafische Darstellungen mit Hilfe genormter Symbole. Sie sind weit verbreitete Hilfsmittel bei der Programmentwicklung. Die Programmstruktur lässt sich bildhaft als Ablauf darstellen. Die Pfeile zeigen dabei die Richtung der Verarbeitung an.

Der PAP reflektiert nicht alle Details des Programms, sondern stellt den allgemeinen Logikfluss dar.

Wenn Sie den PAP korrekt entwickelt haben, ist es nur noch eine Formsache das endgültige Programm zu schreiben. Nachdem das Programm fertig gestellt wurde, können Sie dieses Diagramm ausserdem für die Dokumentation verwenden.

Die einzelnen Symbole werden nachfolgend kurz erklärt.

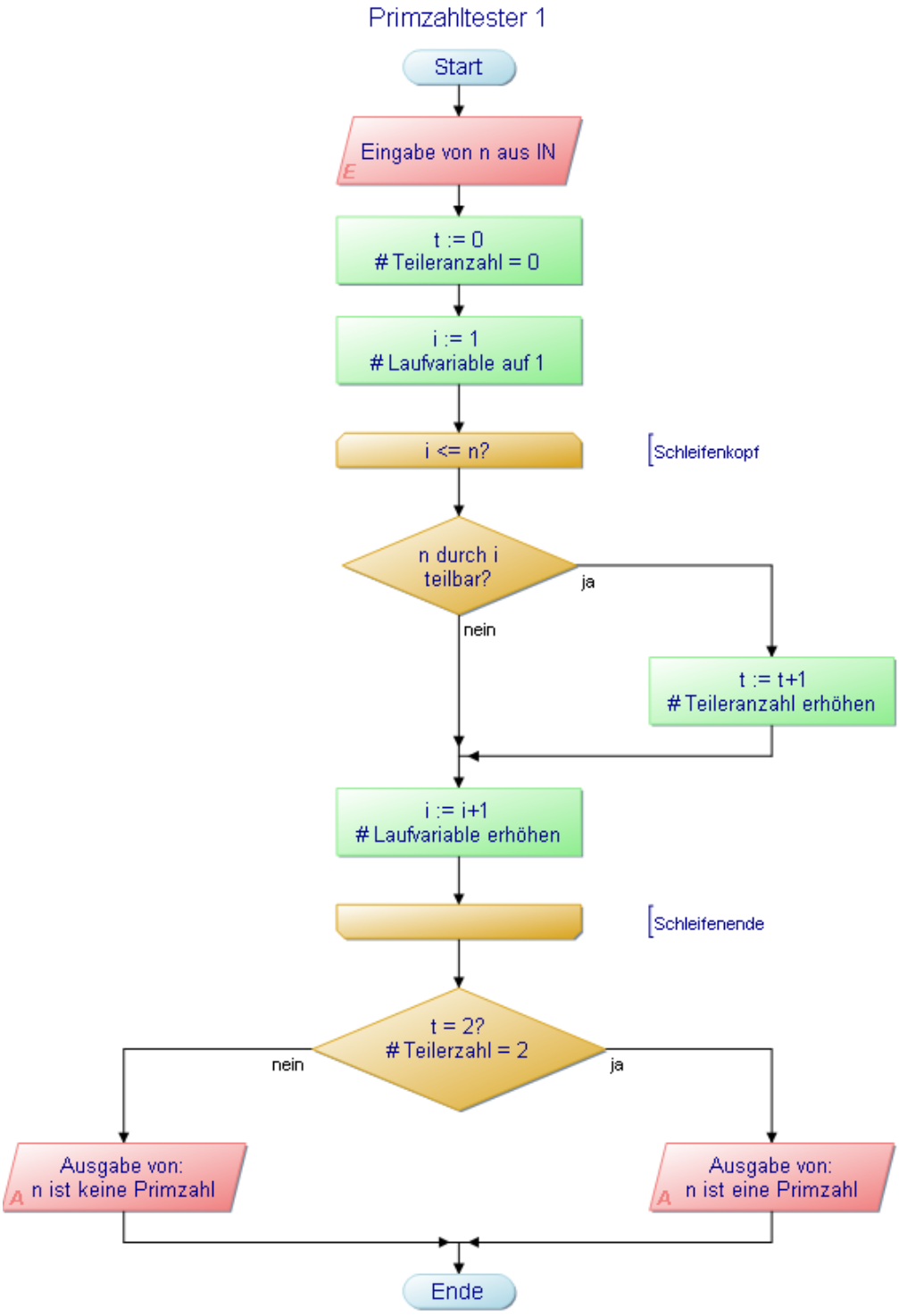
	**Verarbeitung** Das Rechteck ist für Zuweisungen oder Ein- und Ausgabeoperationen vorgesehen.
	**Verbindung** Zur Verdeutlichung der Ablaufrichtung werden Linien mit einer Pfeilspitze benutzt.
	**Verzweigung** Bedingungen werden als Raute dargestellt. Eine Verbindungslinie führt hinein, zwei Ver-bindungslinien führen heraus. Je nach Wahrheitswert der Bedingung wird der Ablauf in die eine oder andere Richtung fortgeführt.
	**Manuelle Verarbeitung** Eingaben des Programmnutzers werden durch ein Trapez oder eine Raute dargestellt.
	**Dokumentation an anderer Stelle** Durch dieses Symbol wird auf einen anderen PAP hingewiesen, der z. B. ein Unterprogramm darstellt.
	**Verbinder** Teilt sich ein Programmablauf und wird dann wieder zusammengeführt, wird der Verbinder (Konnektor) eingesetzt. Somit wird vermieden, dass sich Ablauflinien kreuzen. Damit wird die Übersichtlichkeit erhöht. Für größere PAPs, für die mehrere Seiten benötigt werden, wird der Verbinder am Seitenende der vorherigen und Seitenanfang der folgenden Seite verwendet. Zur Identifizierung wird er mit einer eindeutigen Zahl für diese Verbindungsstelle versehen.
	**Grenzstelle** Eine Grenzstelle kennzeichnet den Anfang und das Ende eines Programmablaufplans.
	**Schleifenbegrenzer** Zur Darstellung von Programmwiederholungen werden diese zwei Symbole benutzt, die den Anfang und das Ende einer Schleife kennzeichnen. In der Praxis verwendet man statt dieser Schleifensymbole oft auch Verbindungen, die wieder zu einem vorherigen Verarbeitungsschritt verweisen.

Aufgabe 01:
Stelle eine Software bereit, mit der du PAP (*Programmablaufpläne) darstellen kannst.*
Empfehlenswerte Freeware **PapDesigner:**

http://friedrich-folkmann.de/papdesigner/Hauptseite.html (Zip-File muss nur entpackt werden)
http://dia-installer.de/ *(GNU General Public License)*

Software PapDesigner.zip

Aufgabe 02:
Betrachte den nachfolgend dargestellten Programmablaufplan „Primzahltester". Wie der Name sagt, soll der Algorithmus für eine eingegebene Zahl überprüfen, ob die Zahl eine Primzahl ist oder nicht. Versuche erst die Logik zu verstehen und setze danach den Programmcode um in Python.

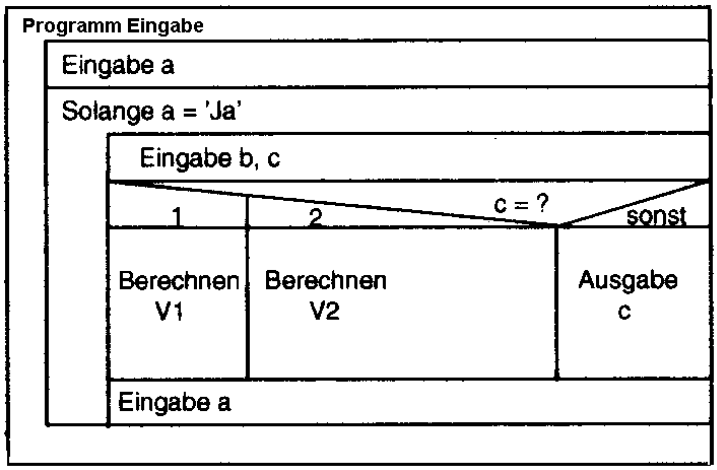


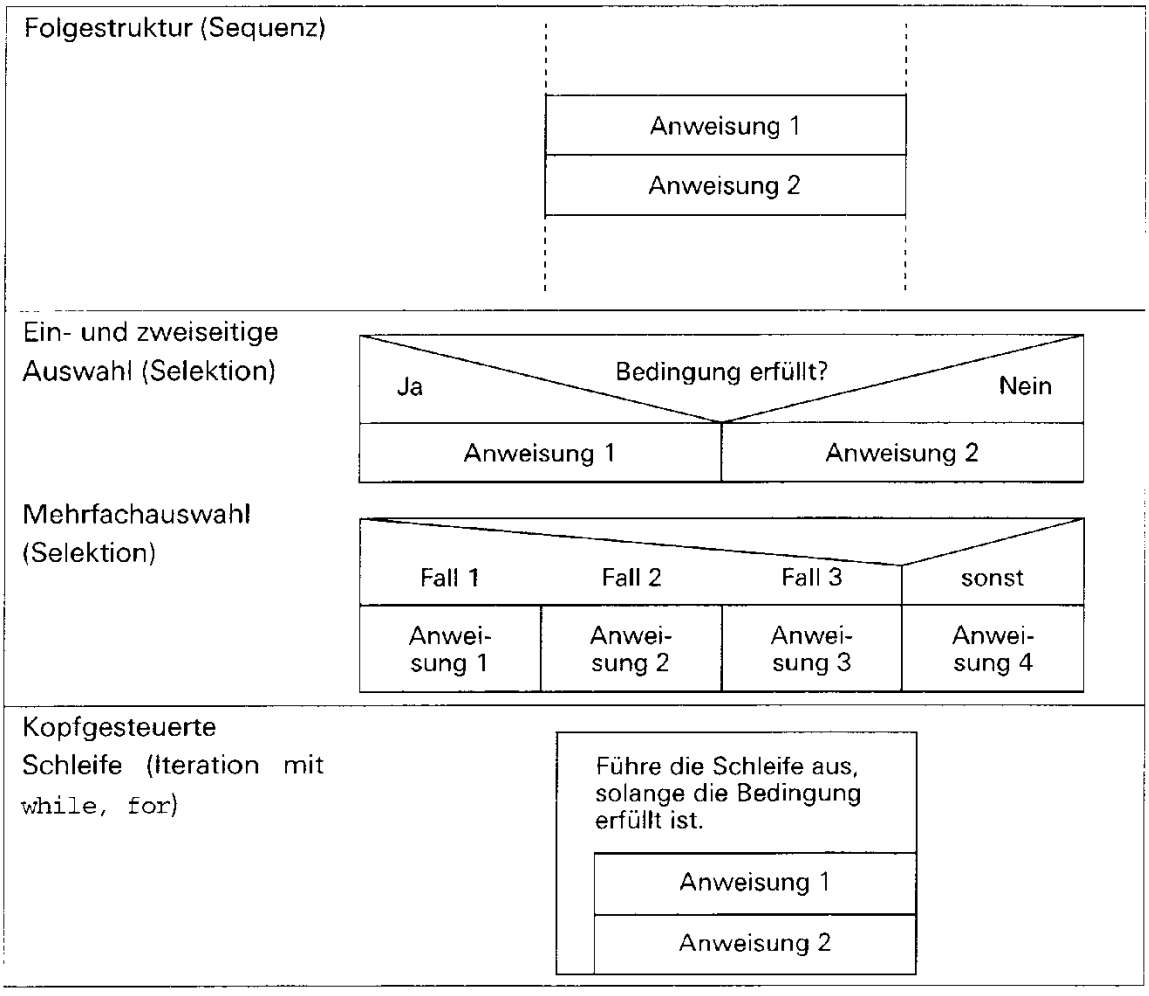
2 Struktogramme (Nassi-Shneiderman)

Struktogramme wurden von I. Nassi und B. Shneiderman als Darstellungsmittel für Algorithmen des strukturierten Programmentwurfs entwickelt.

Ein Struktogramm ist die grafische Darstellung eines Programmablaufs in Form eines geschlossenen Blocks, der entsprechend den einzelnen logischen Grundstrukturen in verschiedene unterordnete Blöcke aufgeteilt werden kann. Struktogramme setzen sich aus verschiedenen Symbolen für die verschiedenen Operationsarten zusammen, die von oben nach unten betrachtet werden (top-down).

Bevor man mit dem eigentlichen Programmieren beginnt, sollte man sich Gedanken machen, was das Programm denn überhaupt können soll. Hierzu wird heute überwiegend das Struktogramm mit Symbolen nach Nassi-Schneidermann eingesetzt. Mit diesem grafischen Hilfsmittel wird eine wesentliche Vereinfachung für die Planung, die Programmierung und den Programmtest (z.B. Fehlersuche) erreicht. Die Nassi-Schneidermann Darstellung ist stark verbreitet. Der Vorteil gegenüber dem Flussdiagramm ist, das die Darstellung dem zu programmierenden Algorithmus viel ähnlicher ist. Der Nachteil ist aber, dass diese Darstellung nur von Fachpersonen gelesen werden kann (für Kunden, Marketing etc. ungeeignet).





Fussgesteuerte Schleife (Iteration mit `do...while`)	Anweisung 1 Anweisung 2 Führe die Schleife aus, solange die Bedingung erfüllt ist.
Aufruf eines Unterprogramms im Hauptprogramm (Abstraktion)	Anweisung 1 Unterprogramm Anweisung 2
Funktionsvereinbarung des Unterprogramms	**Unterprogramm** Anweisung x Anweisung y Anweisung z

Aufgabe 03:
Stelle eine Software bereit, mit der du Nassi-Shneidermann Notation *darstellen kannst.*
Empfehlenswerte Freeware **PapDesigner:**

https://charyan.github.io/nsdmaker/ (kann ohne Installation als WebApp verwendet werden)
Die nebenstehende uralte Applikation (stgr32.zip) (Zip-File muss nur entpackt werden)

stgr32.zip

Aufgabe 04:
Bei Aufgabe 02 hast du den gegebenen Programmablaufplan „Primzahltester" in Python umgesetzt. Stelle nun den Algorithmus nach der Nassi-Shneidermann Notation grafisch dar.

3 Pseudocode

```
begin BetragPruefen
  Eingabe(a);
  Eingabe(b);
  if a >= b then
     Ausgabe(a);
  else
     Ausgabe(b);
  end if
end BetragPruefen
```

Bei dieser Darstellungsart wird die Programmlogik unter Verwendung von "normalen" Sätzen beschrieben.
Der Pseudocode ist eine halbformale, textuelle Beschreibung eines Programmablaufs, die sich am Aufbau der Programmiersprachen anlehnt. Der Pseudocode ist im Gegensatz zu den grafischen Struktogrammen nicht genormt. Für Kontrollstrukturen werden oft ähnliche Worte verwendet, wie sie in der Syntax einer Programmiersprache vorkommen. (if…..then….else…. while ……., etc.)
Für die Anweisungen werden verbale Beschreibungen verwendet (z.B. erhöhe i um 1) oder Anweisungen ähnlich einer Programmiersprache (z.B. i = i + 1). Auch Variablen- und Konstantendeklarationen können enthalten sein.

Ein Beispiel zum Thema Pseudocode:

Für jeden Angestellten:
    Wenn der Angestellte 0 bis 40 Std. gearbeitet hat, dann
        Nettolohn = Std.-Anzahl mal Std.-Lohn
    Andernfalls
        Wenn der Angestellte zwischen 40 und 50 Std. gearbeitet hat, dann
            Nettolohn = 40 mal Std.-Lohn
            Plus (Std.-Anzahl-40) mal Std.-Lohn mal 1,5
        Andernfalls
            Nettolohn = 40 mal Std.-Lohn
            Plus 10 mal Std.-Lohn mal 1,5
            Plus (Std.-Anzahl-50) mal doppeltem Std.-Lohn
    Steuern vom Gehalt abziehen
    Lohnblatt ausdrucken
Löhne in Fibu verbuchen
Aufgabe gelöst

4 Miniprojekte realisieren

Löst im Alleingang möglichst viele der untenstehenden Problemstellungen.

4.1 Projekt 01 (simple code):

📝 **Projekt 01: simple Code**		
Schwierigkeitsgrad: ⭐	⏳ Zeit: 20 Minuten	Arbeitsform: Alleine 👤

Betrachte das folgende Pythonprogramm. Stelle den einfachen Algorithmus als PAP und als Nassi-Shneiderman Applikation dar.

```
startwert = int(input("Startwert eingeben: "))
endwert = int(input("Endwert eingeben: "))
x = startwert
while x <= endwert:
    print(x)
    x = x + 1
```

4.2 Projekt 02 (Geldüberweisung via Twint):

📝 **Projekt 02: Geldüberweisung via Twint**		
Schwierigkeitsgrad: ⭐ ⭐	⏳ Zeit: 30 Minuten	Arbeitsform: Alleine 👤

Du möchtest einer Kollegin CHF 25.-- via Twint überweisen.
Erstelle dazu ein PAP, der Schritt für Schritt zeigt, was man tun muss um die Überweisung auszuführen.
Beachte, dass man das Mobile zuerst mit der Gesichtserkennung entsperren muss. Danach muss man die TwintApp starten, was nur mit dem korrekten Fingerabdruck geht. Kann der Fingerabdruck 3x nicht erkannt werden, muss zwingend das korrekte Twint-Passwort eingeben werden. Wenn das Twint-Passwort 3x falsch ist, wird die App für 1h gesperrt.
Hat das Login funktioniert, kannst du Mobilenummer, Betrag und Nachricht eingeben. Falls man nicht genügend Geld auf dem Konto hat oder die Mobilenummer ungültig ist, erscheinen entsprechende Meldungen.

4.3 Projekt 03 (Bananaway):

📝 **Projekt 03: Bananaway**		
Schwierigkeitsgrad: ⭐	⏳ Zeit: 30 Minuten	Arbeitsform: Alleine 👤

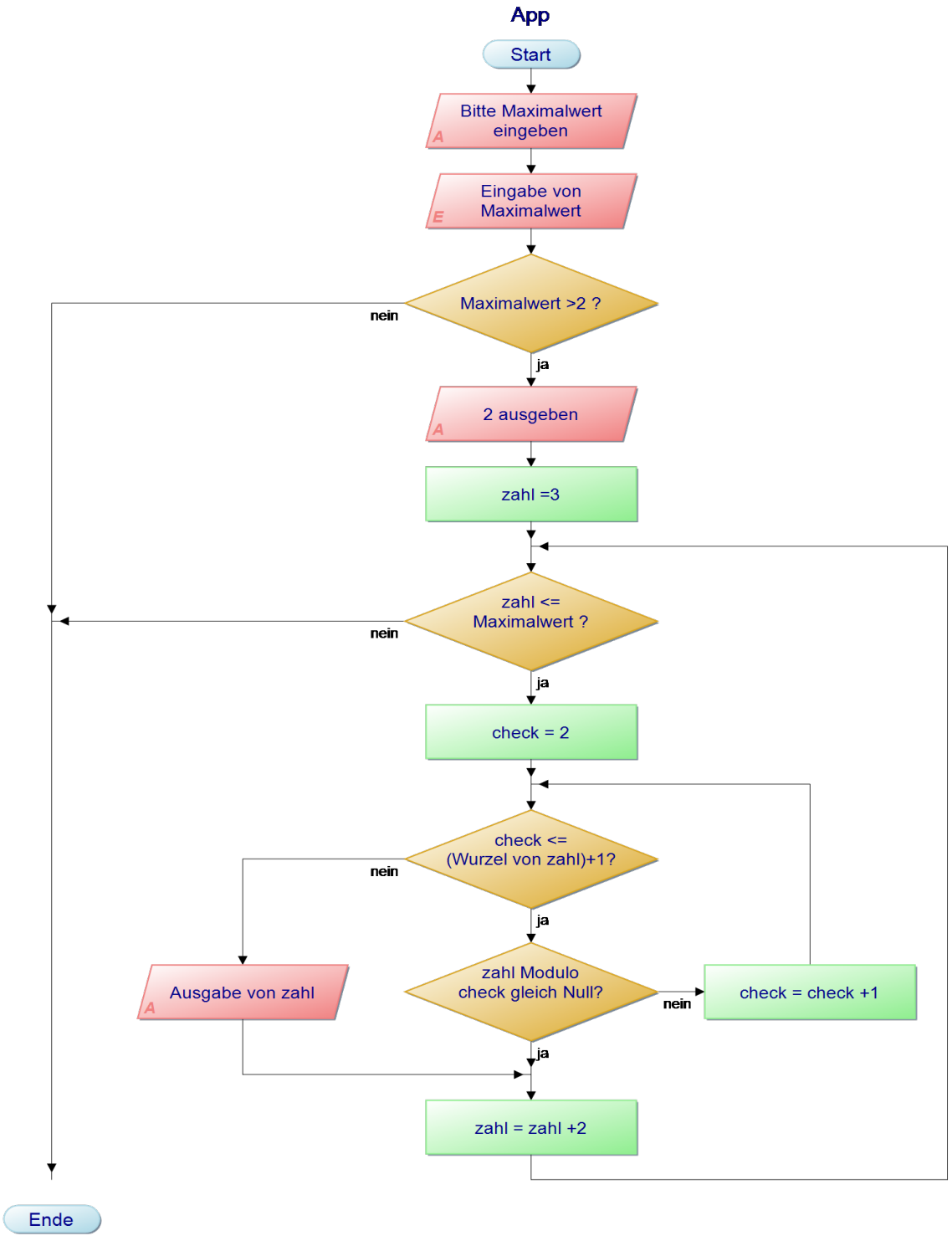
Für ein Spielzeugroboter musst du einen Bewegungsalgorithmus umsetzen. Der Roboter bewegt sich Schritt für Schritt. Nach jedem Schritt prüft er, ob eine Bananenschale auf dem Boden liegt, ob eine Wand vor ihm ist oder ob er das Ziel erreicht hat.
Falls eine Bananenschale auf dem Boden liegt, hüpft er darüber und läuft weiter geradeaus. Wenn eine Wand vor ihm erscheint, dreht er sich um 90 Grad nach links und läuft weiter geradeaus. Fertig ist er erst, wenn das Ziel erreicht ist.
Erstelle den visualisierten Alogrithmus nach PAP oder Nassi-Shneidermann.

4.4 Projekt 04 (PAP umsetzen):

Projekt 05: PAP umsetzen

Schwierigkeitsgrad: ⭐ ⭐	Zeit: 45 Minuten	Arbeitsform: in Zweierteam

Betrachte den folgenden PAP und versuche zu verstehen, was der Algorithmus macht.
01) Beschreibe den Algorithmus mit Pseudocode
02) Setze en Algorithmus in Python um und teste das Resultat (Person 01)
03) Beschreibe den Algorithmus als Nassi-Shneiderman-Diagramm (Person 02)



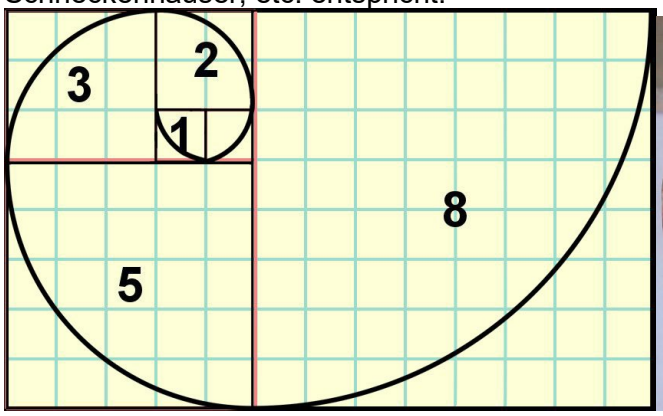
4.5 Projekt 05 (Fibonacci Folge):

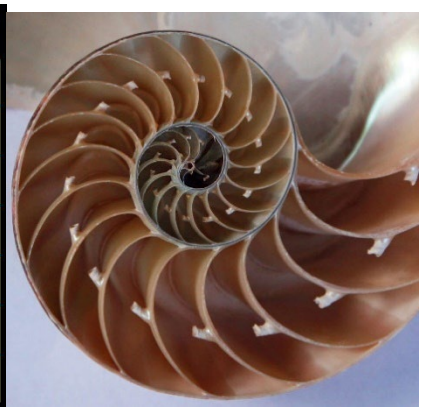
Projekt 05: Fibonacci Folge

Schwierigkeitsgrad: ⭐⭐⭐	Zeit: 45 Minuten	Arbeitsform: in Zweierteam

Kennst du die Fibonacci Folge? Sie liest sich folgendermassen:1, 1, 2, 3, 5, 8, 13, 21, 34, 55...
Man startet mit zwei Zahlen des Wertes Eins (1). Die nachfolgenden Zahlen ergeben sich jeweils aus der Summe der beiden Vorgänger: 1+1 ->2 1+2→3 2+3→ 5, etc.

Wo benötigt man dies? → https://youtu.be/kkGeOWYOFoA
Die Fibonacci Folge trifft man in der Physik, Biologie und Astronomie oft an. Man benutzt sie z.B. um eine Kaninchenpopulation (Anzahl Kaninchenpaare in jedem Monat) vorauszusagen.
Mit der Fibonacci Folge kann man auch die Fibonacci-Spirale herleiten, die z.B. der Form von Muscheln, Schneckenhäuser, etc. entspricht.





Das Video zeigt, wie man die Fibonacci-Spirale aufzeichnen kann: https://youtu.be/D-27wYXd5J0

Die Fibonacci-Folge steht in einem unmittelbaren Zusammenhang zum Goldenen Schnitt, der in der Fotografie und Kunst angewendet wird (siehe Fotos/Bilder am Schluss dieser Aufgabe)

Aufgabe: Erstelle ein Programm, das den Benutzer auffordert eine «Endzahl» einzugeben (z.B. 60). Das Programm soll danach die Fibonacci-Folge bis zur Endzahl ausgeben: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55...
Erstelle dazu den Programmablaufplan (PAP) und die Python-App